Developers Guide, API and Production Deployment

NetBurner 3.4.0f

Thu Oct 26 2023

Chapter 1

NetBurner 3.4.0f

1.1 Welcome to NetBurner Version 3!

This documentation package details the capabilities of your NetBurner Development Software and Tools. If you are viewing this document from your development tools installation, the latest version is available at www.netburner.com. There is also a pdf version of this document at PDF Version

Note

Version 3 is used as a development platform on all ARM based devices, as well as MOD5441x, NANO54415 and SB800EX devices. If you have an earlier ColdFire based device please refer to the version 2 development tools.

1.2 Who Is This Guide For?

NetBurner devices come in two forms:

- Devices pre-programmed with factory applications that can be used without any custom programming on the device itself. Serial to Ethernet devices are a good example of this.
- If you wish to customize any device, we provide development tools that enable you to create custom applications.

This guide is intended for software developers to create custom applications to run on NetBurner devices. If you are using a device that does not require custom programming, please refer to that device's product page for documentation and any appropriate software utilities.

1.3 Recommended Starting Point for New Developers

Your NetBurner development system is pre-programmed with a factory application. If connected to a network with Internet access and a DHCP server, you can open a web browser, go to discover.netburner.com, and access you device immediately; both the application web page and configuration interface.

If you are connected directly, such as a laptop with a direct Ethernet connection, or on a network without a DHCP server and Internet access, please see the procedure below:

- Configure the development kit device/hardware (if necessary) as described in the Quick Start Guide.
- Connect both your development computer and your NetBurner device to a network with Internet access. Internet access is not required for development, but will make the initial configuration of the device easier.
- If using a development kit with a USB interface, connect it from the development board to your computer. It will provide both power and a console/debug serial port. To view status and debug information, run a serial terminal program such as the included MTTTY, although any serial terminal application should work.
- Your NetBurner device will be running a pre-programmed factory application. Open a web browser and type "discover.netburner.com" to identify your device. Links on the web page are available to go directly to the device's web page and configuration page.

  Note

  If you do not have an Internet connection, you can use a utility such as localdiscover. Please refer to the Device Discovery and Configuration section of this guide.
- Refer to the NBEclipse Getting Started Guide section of this document to learn how to create, build and download applications. If you prefer to use command line tools, they are available as well.
- Select one of the examples from the "\nburn\examples" folder. Create a project and download per the NBEclipse Guide.
- Refer to the Programmers Guide and NetBurner API documents for additional information.

Note

NBEclipse provides auto-complete functionality by pressing cntl-space. This makes accessing the API functions much easier.

1.4 Documentation Overview

1.4.1 The NetBurner Application Programming Interface (API)

Description of the functions and capabilities in the NetBurner Library API. These functions include:

- Real-Time Operating System
- TCP/IP Stack
- Peripherals, such as SPI, I2C, 1-Wire, UARTs, ADC, DAC, Timers, etc
- Description of objects, classes and structures
- Function error codes

1.4.2 Device Discovery and Configuration

How to locate your NetBurner device as well as configure system parameters.

1.4.3 The NBEclipse Getting Started Guide

Describes how to create projects and build applications.

1.4.4 Migration Guides

Information for those moving from tools revision 2.x to 3.x.

1.4.5 Example Programs

Well over 100 example programs on everything from web servers, TCP, UDP, Configuration to hardware peripherals. The examples are described in this guide, and located in the "\nburn\examples" folder of your installation.

Note

NBEclipse provides an easy example import feature when you create a new project.

Interesting Starting Examples:

- ShowInterfaces: Displays all available network interface information on a web page and serial port
- SimpleHtml: Minimal web server
- IPADDR-IP Address: IPADDR object operations for assignment and display
- PlatformSpecific folder: Unique peripherals for each hardware platform, such as timers, ADC, DAC and unique I2C and SPI peripherals
- RTOS folder: How to use the various system RTOS capabilities

1.4.6 Programmers Guide

Textbook style guide on how to use the various features of your device and networking.

- Device configuration and the Configuration Server
- HTML Processing and Web Server
- NetBurner Real-Time Operating System (NBRTOS)
- Network Protocols such as TCP and UDP
- Security using SSL/TLS
- Embedded Flash File System (EFFS)

1.4.7 Platform References

The Platform References of this guide provide hardware information such as memory maps and device recovery should your application crash in a way that is not easily reset. Schematics for your NetBurner devices are located in the `\nburn\docs\NetBurner\platform` folder.

1.4.8 System Diagnostics

Information on how to enable and use built-in device diagnostics as well as customizing to add your own.

1.4.9 Utilities

Tools and Utilities that are run on a client computer, such as discover, configuration, and application updates.

1.4.10 Production and Deployment

Creating a great application and end product is just part of what needs to be done for a successful product. NetBurner provides deployment information and utilities that you can use in your production environment to program and configure modules in volume, as well as offer to your end customers.

1.4.11 Location of Original Factory Program

If at any time you wish to restore the original factory program that was running on your device, the pre-built application image is located in the `\nburn\platform\<your platform>\original` folder. For example, `\nburn\platform\MODM7AE70\original`. This path assumes a default install to a folder named `nburn`.

1.4.12 Experienced NetBurner Customers

A significant change you will notice is how the device configuration is accomplished using a web interface. Please refer to the Migration Guide for a detailed description of the differences between release version 3 and previous releases.

Chapter 2

Build System

2.1 Introduction

The Build System section contains information on subjects such as the GCC Compiler and Build Environment.

2.2 GCC Compiler Flags

Compiler flags can be added or modified in both NBEclipse and command line makefiles.

2.2.1 Generic

Language	Mode	Flag	Description
C/C++	R, D	`-gdwarf-2`	Adds dwarf-2 debug symbols to the binary. This makes it easier to resolve symbols in gdb.
C/C++	R, D	`-Wall`	Enables all warnings.
C/C++	R, D	`-Werror=return-type`	Treats warnings as errors for functions that are missing a return statement.
C/C++	R, D	`-Wno-unused`	Suppresses warnings for unused variables.
C/C++	R, D	`-falign-functions=4`	Aligns functions to 4-byte boundaries for improved performance.
C/C++	R	`-O2`	Enables level 2 optimization for improved performance.
C/C++	D	`-O0`	Disables optimization for debugging.

Language	Mode	Flag	Description
C/C++	R, D	'-D'PLATFORM_NAME="$(PLATFORM↩ _NAME)"'\ilinebr </td> <td class="markdown↩ TableBodyNone"> Defines thePLATFORM_NAMEmacro with the value of thePLATFORM↩ _NAMEenvironment variable. \ilinebr </td> </tr> <tr class="markdownTableRow↩ Odd"> <td class="markdown↩ TableBodyNone"> C/↩ C++ \ilinebr </td> <td class="markdownTableBody↩ None"> R, D \ilinebr </td> <td class="markdownTable↩ BodyNone">\ilinebr </td> <td class="markdownTable↩ BodyNone"> Includes the specified directories in the search path for header files. \ilinebr </td> </tr> <tr class="markdown↩ TableRowEven"> <td class="markdownTableBody↩ None"> C \ilinebr </td> <td class="markdown↩ TableBodyNone"> R, D \ilinebr </td> <td class="markdownTableBody↩ None">-std=gnu17\ilinebr </td> <td class="markdown↩ TableBodyNone"> Sets the C standard to GNU C17. \ilinebr </td> </tr> <tr class="markdownTableRow↩ Odd"> <td class="markdown↩ TableBodyNone"> C \ilinebr </td> <td class="markdown↩ TableBodyNone"> R, D \ilinebr </td> <td class="markdownTableBody↩ None">-ffunction-sections\ilinebr </td> <td class="markdown↩ TableBodyNone"> Places each function in a separate section to enable linker optimizations. \ilinebr </td> </tr> <tr class="markdown↩ TableRowEven"> <td class="markdownTableBody↩ None"> C \ilinebr </td> <td class="markdownTable↩ BodyNone"> R, D \ilinebr </td> <td class="markdown↩ TableBodyNone">-fdata-sections\ilinebr </td> <td class="markdown↩ TableBodyNone"> Places each data object in a separate section to enable linker optimizations. \ilinebr </td> </tr> <tr	Specifies the CPU architecture flag.

Language	Mode	Flag	Description

2.2.2 Architecture-Specific Compiler Flags

2.2.2.1 ARM Cortex-M7

Language	Mode	Flag	Description
C/C++	R, D	`-fasynchronous-unwind-tables`	Enables asynchronous unwind tables
C/C++	R, D	`-D_C11_SOURCE`	Defines the `_C11_SOURCE` macro.
C/C++	R, D	`-DCORTEX_M7`	Defines the `CORTEX_M7` macro.

2.2.2.2 ARM Cortex-M0+

Language	Mode	Flag	Description
C/C++	R, D	`-fasynchronous-unwind-tables`	Enables asynchronous unwind tables
C/C++	D	`-fomit-frame-pointer`	Omit frame pointer for improved performance.
C/C++	R, D	`-D_C11_SOURCE`	Defines the `_C11_SOURCE` macro.
C/C++	R, D	`-DCORTEX_M0PLUS`	Defines the `CORTEX_M0PLUS` macro.

2.2.2.3 ColdFire

Language	Mode	Flag	Description
C/C++	R, D	`-fno-omit-frame-pointer`	Do not omit frame pointer for improved debugging.
C/C++	R, D	`-fasynchronous-unwind-tables`	Enables asynchronous unwind tables
C/C++	R, D	`-DCOLDFIRE`	Defines the `COLDFIRE` macro.
Assembler	D	`--defsym _DEBUG=1`	Defines the `_DEBUG` symbol.

2.2.3 CPU-Specific Compiler Flags

2.2.3.1 SAME70

Language	Mode	Flag	Description
C/C++	R, D	`-mcpu=cortex-m7`	Specifies the CPU architecture.
C/C++	R, D	`-mthumb`	Specifies the Thumb instruction set.

Language	Mode	Flag	Description
C/C++	R, D	`-mfpu=fpv5-d16`	Specifies the floating-point unit.
C/C++	R, D	`-mfloat-abi=$(ABI_OPT)`	Specifies the floating-point ABI. Defaults to softfp
C/C++	R, D	`-D$(CPU)`	Defines the `$(CPU)` macro.
C++	R, D	`-fno-use-cxa-atexit`	Disables the use of `__cxa_atexit`
C++	R, D	`-falign-fuinctions=4`	Aligns functions to 4-byte boundaries for improved performance.

2.2.3.2 MIMXRT11xx, MIMXRT10xx

Language	Mode	Flag	Description
C/C++	R, D	`-mcpu=cortex-m7`	Specifies the CPU architecture.
C/C++	R, D	`-mthumb`	Specifies the Thumb instruction set.
C/C++	R, D	`-mfpu=fpv5-d16`	Specifies the floating-point unit.
C/C++	R, D	`-mfloat-abi=$(ABI_OPT)`	Specifies the floating-point ABI. Defaults to softfp
C/C++	R, D	`-D$(CPU)`	Defines the `$(CPU)` macro.
C++	R, D	`-fno-use-cxa-atexit`	Disables the use of `__cxa_atexit`
C++	R, D	`-falign-fuinctions=8`	Aligns functions to 8-byte boundaries for improved performance.

2.2.3.3 SAMD20

Language	Mode	Flag	Description
C/C++	R, D	`-mcpu=cortex-m0plus`	Specifies the CPU architecture.
C/C++	R, D	`-mthumb`	Specifies the Thumb instruction set.
C++	R	`-fno-use-cxa-atexit`	Disables the use of `__cxa_atexit`
C++	R	`-falign-fuinctions=4`	Aligns functions to 4-byte boundaries for improved performance.

2.2.3.4 MCF5441X

Language	Mode	Flag	Description
C/C++	R, D	`-D$(CPU)`	Defines the `$(CPU)` macro.
C/C++	R, D	`-mcpu=54415`	Specifies the CPU architecture.

2.2.4 Platform-Specific Compiler Flags

2.2.4.1 MODM7AE70

Language	Mode	Flag	Description
C/C++	R, D	`-D__SAME70Q21__`	Defines the `__SAME70Q21__` macro.
C/C++	R, D	`-DMODM7AE70`	Defines the `MODM7AE70` macro.

2.2.4.2 SBE70LC

Language	Mode	Flag	Description
C/C++	R, D	-D__SAME70Q21↩ __	Defines the __SAME70Q21__ macro.
C/C++	R, D	-DSBE70LC	Defines the SBE70LC macro.

2.2.4.3 SOMRT1061

Language	Mode	Flag	Description
C/C++	R, D	-D__MIMXRT106xx↩ __	Defines the __MIMXRT106xx__ macro.
C/C++	R, D	-D__MIMXRT1061_↩ _	Defines the __MIMXRT1061__ macro.
C/C++	R, D	-DSOMRT1061	Defines the SOMRT1061 macro.

2.2.4.4 MOD5441X

Language	Mode	Flag	Description
C/C++	R, D	-DMOD5441X	Defines the MOD5441X macro.

2.2.4.5 NANO54415

Language	Mode	Flag	Description
C/C++	R, D	-DNANO54415	Defines the NANO54415 macro.

2.2.4.6 SB800EX

Language	Mode	Flag	Description
C/C++	R, D	-DSB800EX	Defines the SB800EX macro.

2.3 Linker Flags

2.3.1 Generic

Flag	Description
$(CPUFLAG)	Specifies the CPU architecture flag.
-Wl,-n	Disables linking to dyn
-T$(NNDK_ROOT)/platform/$(PLATFORM)/lib/$(LINKER_SCRIPT)	Specifies the linker script.

Flag	Description
`-Wl,-R$(NNDK_ROOT)/platform/$(PLATFORM)/ldscripts/sysinfo.ld`	Specifies the linker script.
`-Wl,-Map=$(@:.elf=.map)`	Generates a map file.
`-Wl,--cref`	Generates a cross-reference table.
`-Wl,--gc-sections`	Removes unused sections to reduce the size of the binary.
`-Wl,--start-group,$(NBLIBS) $(XTRALIB) -lstdc++ -lc -Wl,--end-group`	Specifies the libraries to link.

2.3.2 Architecture-Specific Linker Flags

None

2.3.3 CPU-Specific Linker Flags

2.3.3.1 SAME70

Flag	Description
`-nostartfiles`	Disables linking to the default startup files.
`-mfloat-abi=$(ABI_OPT)`	Specifies the floating-point ABI. Defaults to softfp
`-Wl--thumb-entry=Reset_Handler`	Specifies the entry point.
`-Wl--thumb-entry=_start_`	Specifies the entry point. (SOMRT1061)

2.3.3.2 MIMXRT11xx, MIMXRT10xx

Flag	Description
`-nostartfiles`	Disables linking to the default startup files.
`-mfloat-abi=$(ABI_OPT)`	Specifies the floating-point ABI. Defaults to softfp
`-Wl--thumb-entry=Reset_Handler`	Specifies the entry point.

2.3.3.3 SAMD20

Flag	Description
`-nostartfiles`	Disables linking to the default startup files.
`-Wl--thumb-entry=Reset_Handler`	Specifies the entry point.

2.3.3.4 MCF5441X

None

2.3.4 Platform-Specific Linker Flags

None

2.4 Pack Binary

2.4.1 Generic

None

2.4.2 Architecture-Specific

None

2.4.3 CPU-Specific

2.4.3.1 MCF5441X

Executable	I/O	Flag	Description
m68k-elf-objcopy	.elf -> .s19	-O srec	Converts the elf file to s19
m68k-elf-objcopy	.elf -> .s19	-Wl,--gc-sections	Removes unused sections to reduce the size of the binary.
m68k-elf-objcopy	.elf -> .s19	--strip-all	Removes all symbols from the binary.
compcode	.s19 -> _APP.s19	-M $(^↩ :.s19=.map)	Specifies the map file.
compcode	.s19 -> _APP.s19	-R	Print report.
compcode	.s19 -> _APP.s19	-P$(PLATFORM)	Specifies the platform.
compcode	.s19 -> .bin	-O binary	Converts the s19 file to binary
compcode	.s19 -> .bin	-M $(^↩ :.s19=.map)	Specifies the map file.

2.4.3.2 SAMD20

Executable	I/O	Flag	Description
nbfspack	.elf -> .bin	-output-target bin	Converts the elf file to binary

2.4.3.3 SAME70

Executable	I/O	Flag	Description
arm-none-eabi-size	.elf -> .bin		Displays the size of the binary.
nbfspack	.elf -> .bin	-output-target bin	Converts the elf file to binary
nbfspack	.elf -> .bin	-S	Show build statistics.
nbfspack	.elf -> .bin	-P $(PLATFORM)	Specifies the platform.

2.4.3.4 MIMXRT11xx, MIMXRT10xx

Executable	I/O	Flag	Description
nbfspack	.elf -> .bin	-output-target bin	Converts the elf file to binary

2.4.4 Platform-Specific

2.4.4.1 MODM7AE70

Executable	I/O	Flag	Description
nbfspack	.elf -> .bin	-C	Compress program sections
nbfspack	.elf -> .bin	-cflag S:3	.
nbfspack	.elf -> .bin	-cflag S:0	.
nbfspack	.elf -> .bin	-cflag S:1	.

2.4.4.2 SBE70LC

Executable	I/O	Flag	Description
nbfspack	.elf -> .bin	-C	Compress program sections
nbfspack	.elf -> .bin	-cflag S:3	.
nbfspack	.elf -> .bin	-cflag S:0	.
nbfspack	.elf -> .bin	-cflag S:1	.

2.4.4.3 SOMRT1061

None

2.4.4.4 MOD5441X

Executable	I/O	Flag	Description
compcode	.s19 -> .bin	0xC0040000 0xC1FC0000	Application address range.

2.4.4.5 NANO54415

Executable	I/O	Flag	Description
compcode	.s19 -> .bin	0x04000 0x800000	Application address range.

2.4.4.6 SB800EX

Executable	I/O	Flag	Description
compcode	.s19 -> .bin	0x04000 0x800000	Application address range.

Chapter 3

Command Line Tools

3.1 Introduction

As an alternative to NBEclipse, you can use the command line tools to build your NetBurner projects. In fact, all the example programs are built and tested internally with command line tools. You will find a makefile in each example folder, so any of the examples can be used as a demonstration of how to use command line tools. Windows, Mac and Linux are supported. The command line tools can also be used to invoke the NetBurner tools from the development environment of your choice.

3.2 Command Line Summary

Command	Description
make	Build a project using the makefile in the project directory
make -j	Make using multiple cores to speed up the process
make clean	Delete a project's libraries and object files

3.3 Create a New Project From an Example

The easiest way to create a new project is to copy an existing one and modify it:

- Open a command prompt at the location where you keep your projects, such as `\NetBurner\Projects`
- Create a folder for your project, lets use HelloWorld as an example
- Move to the HelloWorld folder
- Copy an example, such as `\nburn\examples\web\SimpleHtml`

- At this point your project structure will look like:
```
\NetBurner\Projects
   \- HelloWorld
         \- html
             |- index.html
         \- src
             |- main.cpp
         | makefile
```
- Run "make -j" to build the project. The "-j" options uses multiple cores for the build
- When the build completes, an obj folder is created with the object files and .bin application image file: `\NetBurner\Projects\HelloWorld\obj\release\SimpleHtml.bin` (and also .s19 for SREC platforms).
```
\NetBurner\Projects
   \- HelloWorld
         \- html
             |- index.html
         \- obj
             \- release
                 |- SimpleHtml.bin
         \- src
             |- main.cpp
         | makefile
```
- To load the application into your device, use the "make -j load" command.

There are many files and folders under the obj folder, including the compiled libraries and object files.

Note

Each project has its own build of the libraries and system files. That way if any changes need to be made (which is rare), it can be done on a per-project basis. This will be discussed in the overload section of this document.

We named our project HelloWorld, but are running SimpleHtml at this point. To customize the project:

- Edit the makefile to change the application image file name to HelloWorld
- Edit main.cpp to change the application name to HelloWorld
- Run "make -j load" to build and load the application into the device

3.3.1 Edit the makefile

The example makefile is shown below:
```
NAME = SimpleHtml

CPP_SRC = src/main.cpp

CREATEDTARGS += src/htmldata.cpp
CPP_SRC += src/htmldata.cpp
src/htmldata.cpp : $(wildcard html/*.*)
    comphtml html -osrc/htmldata.cpp

include $(NNDK_ROOT)/make/boilerplate.mk
```

Changing the NAME from SimpleHtml to HelloWorld will create the image file as HelloWorld.bin (and .s19)

The CPP_SRC is a list of all .cpp files to build. In this case there is only one, but if there were more it would look like:
```
CPP_SRC  = src/main.cpp \
        += src/file2.cpp \
        += src/file3.cpp \
        += src/file4.cpp
```

If you have .c source files you can add them with C_SRC in the same manor.

The html related lines auto-generate a cpp file from the code in the project's html directory.

The final line calls the boilerplate makefile which includes the platform specific make instructions. While you can drill down into the details, this type of makefile structure makes creating projects much easier since you only need to list the source files of your project.

3.3.2 Edit main.cpp

A fully functional application with the web server enables is actually very few lines of code:

```
#include <init.h>
#include <nbrtos.h>
#include <system.h>

const char *AppName = "Simple HTML Example";

void UserMain(void *pd)
{
    init();                                     // Initialize network stack
    StartHttp();                                // Start web server, default port 80
    WaitForActiveNetwork();

    iprintf("Web Application: %s\r\nNNDK Revision: %s\r\n", AppName, GetReleaseTag());
    while (1)
    {
        OSTimeDly(TICKS_PER_SECOND);
    }
}
```

This example just initializes the system, starts the web server, prints out some information, and loops forever with a 1 second delay. Changing the AppName to HelloWorld will change the name that shows up in the find and configuration utilities.

3.4 Modifying System Files with Overload

There can be instances in which you need to modify a NetBurner system file. This is accomplished using the Oveload feature. NBEclipse creates an overload folder automatically, but with command line it must be created manually.

Note

The overload example is located in `\nburn\examples\Overload`

Adding an overload folder to our HelloWorld project:

```
\NetBurner\Projects
    \- HelloWorld
          \- html
              |- index.html
          \- obj
              \- release
                   |- SimpleHtml.bin
          \- overload
          \- src
              |- main.cpp
          | makefile
```

Once you know which file you want to overload, the path in the overload folder must exactly match the system file path. For example, if we want to overload the predef.h include file located in the `\nburn\nbrtos\include` folder:

```
\NetBurner\Projects
    \- HelloWorld
          \- html
              |- index.html
          \- obj
              \- release
                   |- SimpleHtml.bin
          \- overload
                \- nbrtos
                     \- include
                          |- predef.h
          \- src
              |- main.cpp
          | makefile
```

Now the project's predef.h will be used along with any changes made to it.

3.4.1 Important Overload Rules

- The first time a file is added to the overload folder, a "make clean" is required. From that point on, any modifications to that file will automatically handle building all necessary system files.

If a system include folder is overloaded, this folder should be added to your project include paths. Right click on the project and select project properties. Under C/C++ Build->Settings, select GNU C++ Compiler->Includes and add the overload include folder. If utilizing C code, then GNU C Compiler->Includes should also be added.

3.5 GNU Debugger (GDB)

To load an application and start GDB from the command line:

- Build and load a debug build of the application `make loaddebug -j`.
- Start GDB:
  - For ARM devices such as the MODM7AE70 and SBE70LC: `arm-eabi-gdb -se obj/debug/<elf.↩ file>`
  - For MODM7AE70, SBE70LC, and SOMRT1061 platforms: `m68k-elf-gdb -se obj/debug/<elf.↩ file>`
- Connect to the target device with: `target remote <ipaddress>:2159`

To end the GDB session and leave the target running use `detach`. To end the GDB session use `quit`. The GDB command reference is located here: http://www.gnu.org/software/gdb/documentation/.

3.5.1 Example GDB Session

An GDB session running the SimpleHtml example on a MOD54417 is shown below. The example has been modified to add some variables that do simple counting.

```
void UserMain(void *pd)
{
    init();                                     // Initialize network stack
    StartHttp();                                // Start web server, default port 80
    WaitForActiveNetwork(TICKS_PER_SECOND * 5);

    iprintf("Web Application: %s\r\nNNDK Revision: %s\r\n", AppName, GetReleaseTag());

    uint32_t i = 0;
    uint32_t j = 0;
    while (1)
    {
        OSTimeDly(TICKS_PER_SECOND);
        i++;
        j = j + 1;
        iprintf("i: %ld, j: %ld\r\n", i, j);
    }
}
```

Load the application. Note that the DEVIP environment variable is set to the device's IP address 10.1.1.169. The application will begin execution and wait for GDB to connect.

```
C:\NetBurner\projects\SimpleHtml>make loaddebug -j
C:/nburn/pcbin/nbupdate obj/debug/DBSimpleHtml.bin 10.1.1.169
uriReq: http://10.1.1.169:20034/appupdate.htm

urlStr: http://10.1.1.169:20034/appupdate.htm
200

C:\NetBurner\projects\SimpleHtml>
```

Start GDB:

```
C:\NetBurner\projects\SimpleHtml>m68k-elf-gdb -se obj/debug/DBSimpleHtml.elf
GNU gdb (GDB) 8.2
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=m68k-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from obj/debug/DBSimpleHtml.elf...done.
(gdb)
```

Connect to the MOD54417 target on port 2159. This will pause the application at whatever line of code it happens to be on. For simple examples that will typically be the NBRTOS idle task.

```
(gdb) target remote 10.1.1.169:2159
Remote debugging using 10.1.1.169:2159
OSTaskIdle (data=0x0) at C:/nburn/arch/coldfire/source/nbrtosmain.cpp:425
425             asm ( " NOP" );
(gdb)
```

At this point you can do whatever type of GDB commands you wish. To set a break point at the `i++;` at line 29 of main.cpp and use the `continue` command to executee until the break point is reached:

```
(gdb) break main.cpp:29
Breakpoint 1 at 0x40009282: file src/main.cpp, line 29.
(gdb) info break
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x40009282 in UserMain(void*) at src/main.cpp:29
(gdb) continue
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 1, UserMain (pd=0x0) at src/main.cpp:29
29                      i++;
(gdb)
```

The `list` command can be used to view the source code around the breakpoint:

```
Thread 2 hit Breakpoint 1, UserMain (pd=0x0) at src/main.cpp:29
29                      i++;
(gdb) list
24              uint32_t i = 0;
25              uint32_t j = 0;
26          while (1)
27          {
28              OSTimeDly(TICKS_PER_SECOND);
29                      i++;
30                      j = j + 1;
31                      iprintf("i: %ld, j: %ld\r\n", i, j);
32          }
33      }
(gdb)
\code

\latexonly \end{tcolorbox} \endlatexonly



\n
Now view the variables with print, as well as all local variables:

\latexonly \begin{tcolorbox} \endlatexonly

\code
(gdb) print i
$1 = 2
(gdb) info local
i = 2
j = 2
(gdb)
```

Use `next` to go to the next line without stepping into a function (the `step` command steps into a function). GDB commands can typically use just the first letter, in this case `n`:

```
(gdb) n
30                      j = j + 1;
(gdb) info local
i = 3
j = 2
(gdb)
```

To make changes to your code use the `detach` command do that the application can resume and will be ready for your next code download:

```
(gdb) detach
Detaching from program: C:\NetBurner\projects\SimpleHtml\obj\debug\DBSimpleHtml.elf, Remote target
Ending remote debugging.
[Inferior 1 (Remote target) detached]
(gdb)
```

To exit GDB, use the `quit` command:

```
(gdb) quit

C:\NetBurner\projects\SimpleHtml>
```

Chapter 4

Device Discovery and Configuration

4.1 Introduction

Device discovery and configuration can be accomplished in a number of ways, including methods that are platform independent, including Windows, macOS, Linux, tablets and mobile phones.

Note

Please refer to the Production & Deployment section of this document for methods and utilities designed to program and configure devices in a production environment.

4.1.1 Device Discovery Methods:

- If the device and computer have Internet access, open a web browser and go to 'discover.netburner.com'.
- Use the localdiscover utility located in `\nburn\pcbin\localdiscover`. This is a multi-platform utility written in Golang for Windows, macOS and Linux. The compiled version can be distributed to your customers, but not the Golang source code.
- Use the Python find utility, located in `\nburn\pctools\find`.
- Use the debug/console serial port.

4.1.2 Device Configuration methods:

- Interactively using a web browser to access the device's Configuration Server
- Sending a JSON object and a utility such as wget (See Production & Deployment)
- The device's serial interface to the Configuration Server

The primary configuration method is through the device's Configuration Web Server. By utilizing a web server, a device can be configured from any platform or operating system. The underlying configuration mechanism is handled through a JavaScript Object Notation (JSON) object. This will be described in more detail later, but the important point is that it provides a simple interface that can be used by non-embedded developers (e.g. web developers) to interact and configure the device. The implementation is also designed to enable you to add any application specific information to be stored and accessed in the same manor, such as calibration values, set points, etc.

You are also able to create your own configuration web interface that will appear in place of the default interface. In this way you can organize and present data in the appropriate way for your customers, as well as add your own custom logo, images and descriptions. Examples are provided.

4.2 Configuration Procedure

- Connect your NetBurner device to a network with Internet access.
- Open a web browser on your host computer and go to `discover.netburner.com`. You will see a list of devices on your network. (If the device does not have Internet access, use the localdiscover utility instead).

All system and application settings can now be viewed and/or modified by navigating the tree structure.

Using `discover.netburner.com`:

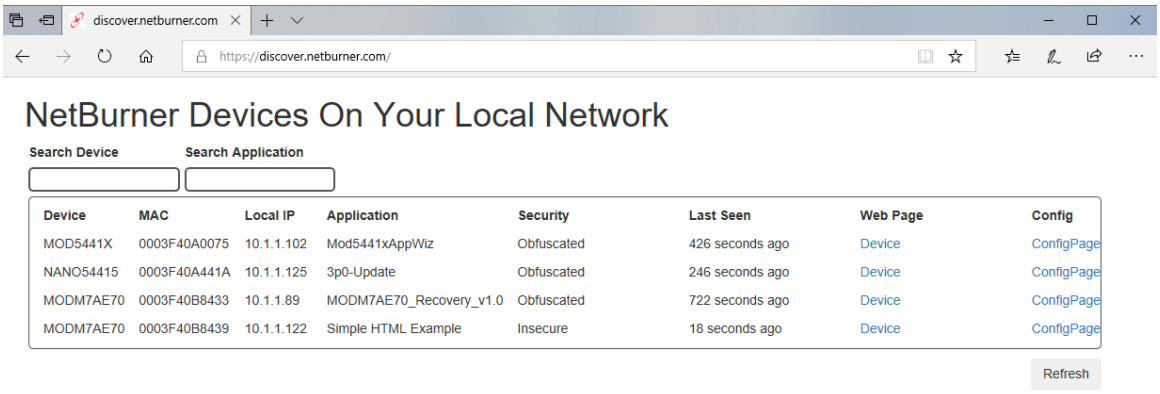


Figure 4.1 Discover using discover.netburner.com

Using `localdiscover` utility:

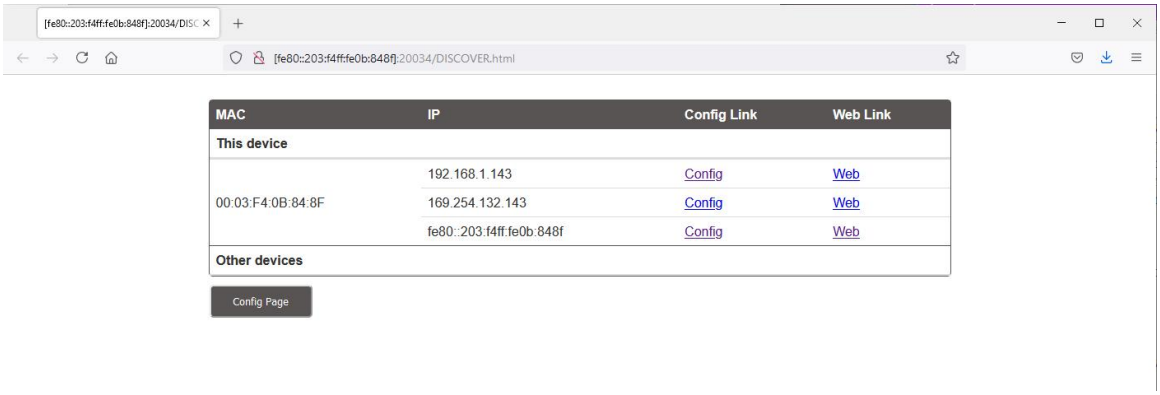


Figure 4.2 LocalDiscover Utility

The "Device" link will take you to the device application web page. The "ConfigPage" link will take you to the device configuration web page, as shown below:

Figure 4.3 Configuration Web Page

4.3 The JSON Object

It is not necessary to understand the underlying JSON implementation, but for those that are interested a sample JSON object is shown below. It identifies each configuration field, which can range from a string, boolean, or a selection of values. For developers, the programmers guide and examples provide the operational details.

Warning

If you do decided to use any of the JSON configuration methods, you must first download a JSON object from your specific device. Under no circumstances should you use the example below as you actual device will likely be different in some respects.

The root of the object is Config. The next levels are AppData and Sys. AppData is available for your application to store information. Sys is used for the system configuration such as IP settings.


```
{
    "Config":{
        "AppData":{},
        "Sys":{
            "Application":"Simple HTML Example",
            "Boot":{
                "Abort":"A",
                "BootBaud":115200,
                "BootDelay":2,
                "BootQuiet":false,
                "BootUart":0,
                "Password":"",
                "SerialConfig":{
                    "Choices":"DuringBoot, AlwaysEnabled, PauseAfterBoot, Disabled", "Value":"DuringBoot"
                },
                "User":""
            },
            "NetIf":{
                "Ethernet0":{
                    "DeviceName":"",
                    "DhcpDiscoverSec":1,
                    "DiscoveryObfuscate":true,
                    "DiscoveryReportInterval":900,
                    "DiscoveryReportUrl":"discover.netburner.com/DevicePost",
                    "IPv4":{
                        "ActiveAddr":"10.1.1.73",
                        "ActiveDNS1":"10.1.1.1",
                        "ActiveDNS2":"0.0.0.0",
                        "ActiveGate":"10.1.1.1",
                        "ActiveMask":"255.255.252.0",
                        "AutoIPAddr":"169.254.131.245",
                        "AutoIPEn":true,
                        "Mode":{
                            "Choices":"DHCP,DHCP w Fallback,Static,Disabled", "Value":"DHCP"
                        },
                        "StaticAddr":"0.0.0.0",
                        "StaticDNS1":"0.0.0.0",
                        "StaticDNS2":"0.0.0.0",
                        "StaticGate":"0.0.0.0",
                        "StaticMask":"0.0.0.0"
                    },
                    "IPv6":{
                        "ActiveAddr":["2602:306:b8e9:c83f::14c0","2602:306:b8e9:c83f:203:f4ff:fe0b:83f5","fe80::203:f4ff:fe0b
                        "ActiveDNS":["2602:306:b8e9:c83f:208:a2ff:fe0c:b081","2602:306:b8e9:c83f:208:a2ff:fe0c:b081"],
                        "ActiveRoute":["fe80::1:1"],
                        "Mode":{
                            "Choices":"DHCP,DHCP w Fallback,Static,Disabled", "Value":"DHCP"
                        },
                        "StaticAddr":"::",
                        "StaticDNS1":"::",
                        "StaticDNS2":"::"
                    },
                    "MAC":"00:03:F4:0B:83:F5"
                }
            },
            "Platform":"MODM7AE70"
        },
```

```
        "Version":8,
        "Reboot":false
    }
}
```

4.3.1 Developer Notes

If you were writing a web interface and need only the data under Ethernet in the tree, it can be accessed by: http://10.1.1.71:20034/UI.html?CONFIG/SYS/NETIF/Ethernet

To obtain just the IP address: http://10.1.1.71:20034/UI.html?CONFIG/SYS/NETIF/↩ Ethernet/IPv4/ActiveAddr

Recommended Examples: The following examples are recommended to begin evaluating the platform. They are located in the `\nburn\examples` directory.

- ShowInterfaces
- `\Configuration\Web\BasicWebConfig`

4.4 Configuration Security

One of the features of NetBurner 3.0 is that each device is configured though its own configuration web interface. Data sent and received through the network interface has three options for security:

- Open: No security, plain text
- Obfuscated: Encrypted, but without TLS and a secure certificate and private key
- Secure: Encrypted with TLS and a certificate and private key

4.5 Serial Port Configuration

If enabled, the Configuration Server can be accessed through the device serial port. The configuration tree is exactly the same as what you see on the web interface and in the JSON object. For example, the Ethernet0 settings are located at `Config > Sys > NetIf > Ethernet0`.

The serial port interface is located on UART0 by default. It can be modified and enabled/disabled in the `Config > Sys > Boot` level of the Configuration Server.

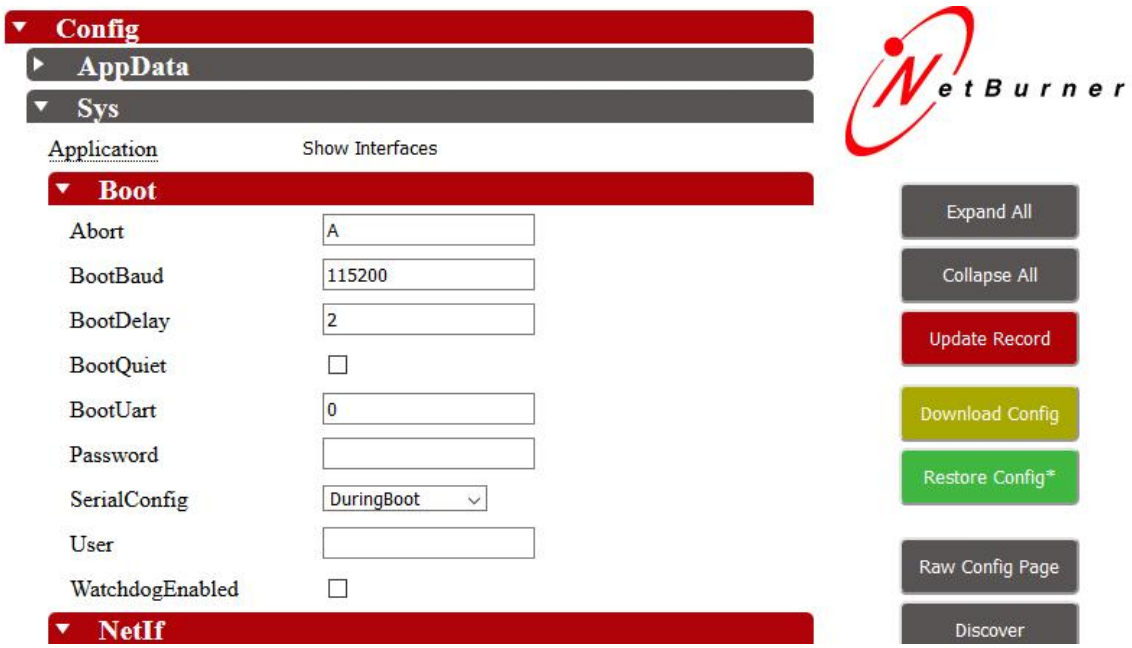


Figure 4.4 Config Boot Level

To access the serial port configuration:

- Connect a cable to the serial port selected as the BootUart
- Run a serial terminal program such as MTTTY
- Reset the device, and type a 'A' when prompted to Abort the boot sequence

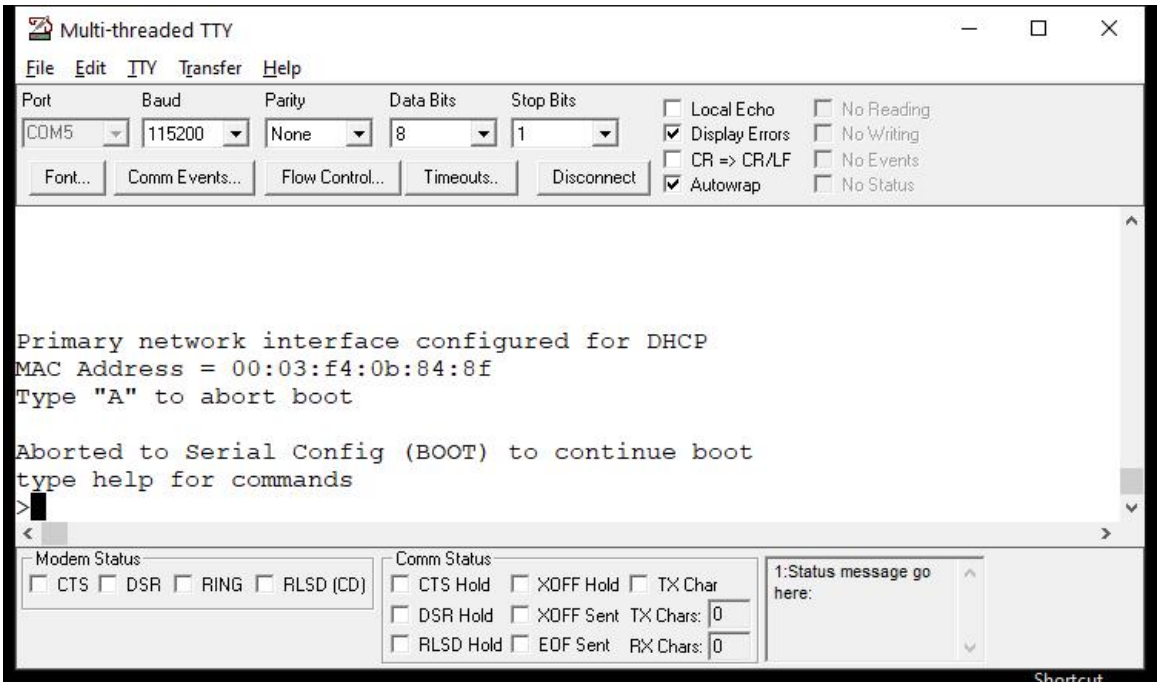


Figure 4.5 Abort Boot Sequence

You are now in the serial configuration mode as indicated by the '>' character. Typing "help" will display the help menu.

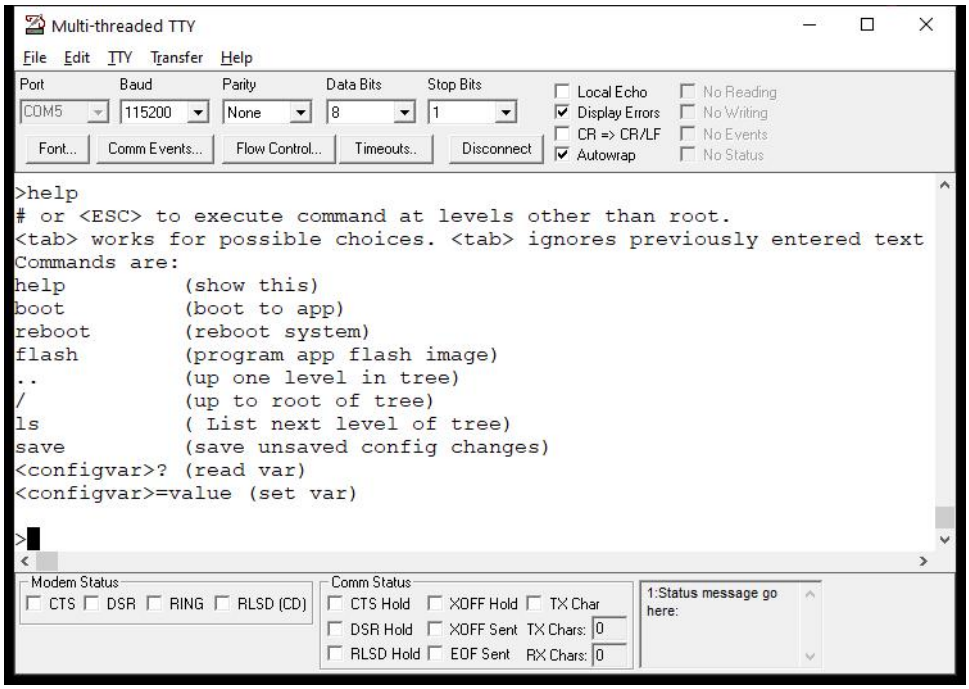


Figure 4.6 Serial Help Menu

If we type the "ls" command we can see the next level of the tree is `Config`, since we are at the root level. In this example we will set the Static IP values. Note that to change from DHCP to static, we must also change the `Mode`

option from DHCP to Static. To navigate to the next level you type in the name of that level. For example, to go to the `Config` level:

Figure 4.7 Config Level

The prompt will show the current location, in this case `Config>`. In the same manor to navigate to the IPv4 settings the level names are typed into the terminal. Note that tabbed auto-completion is also available.

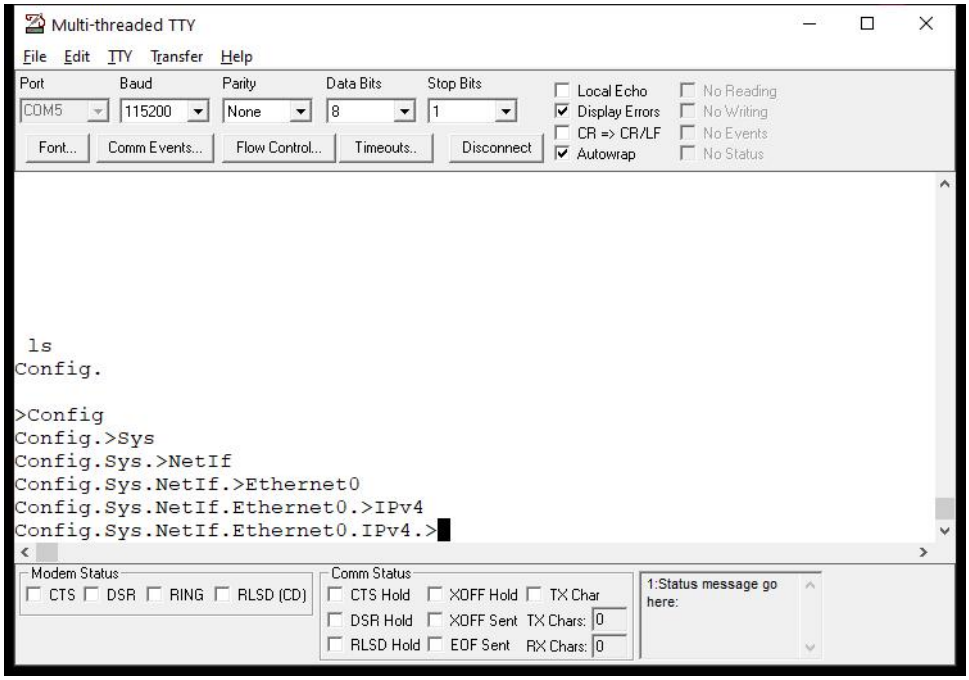


Figure 4.8 IPv4 Level

At any level, typing a "?" will display the current values of that level and all levels beneath in the JSON object format. For example:

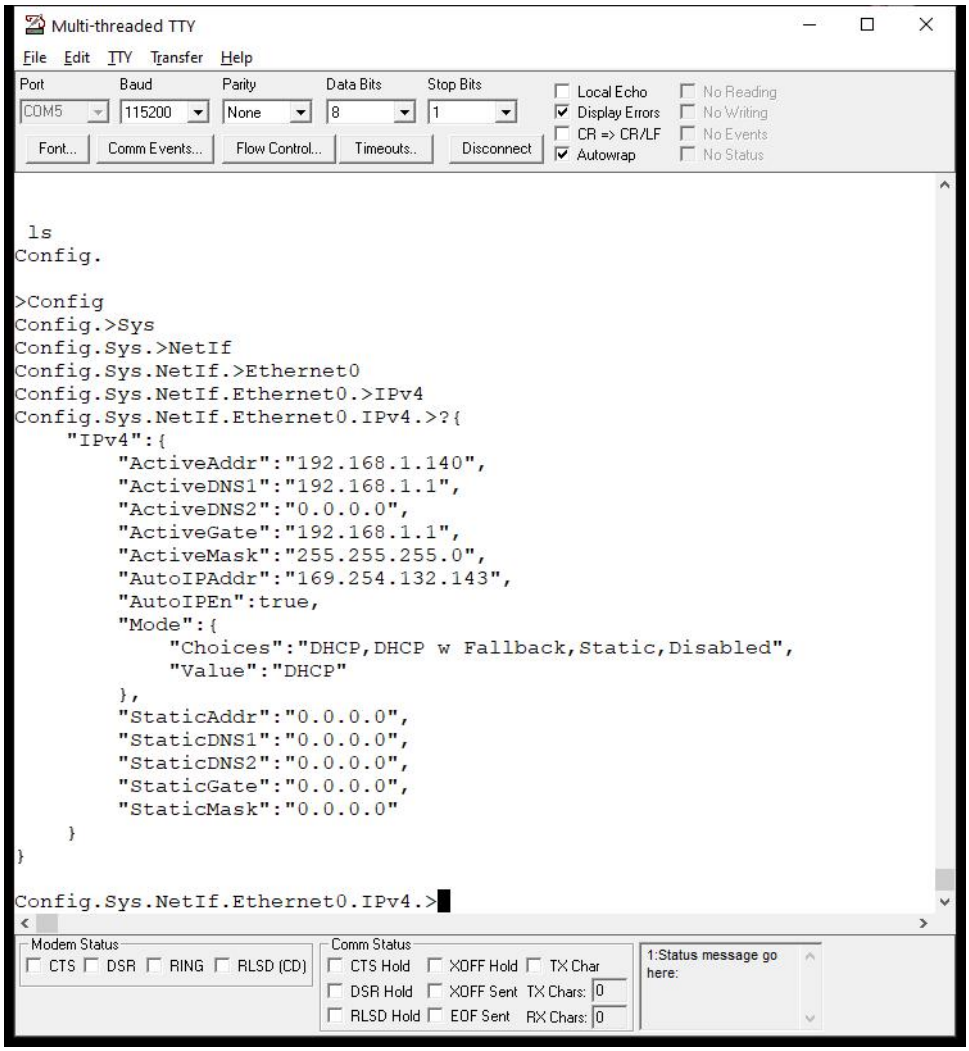


Figure 4.9 Display IPv4 Level

In this example we are interested in the static IP settings. There are two choices to set them: from the IPv4 level by including the name of the next level and the value, or by going down to each level individually and entering just the value. In this case we will stay at the IPv4 level and include the name of the static item we want to set. To set the `StaticAddr` type: `StaticAddr="10.1.1.99"`. We will also type a "?" to see the changed value:

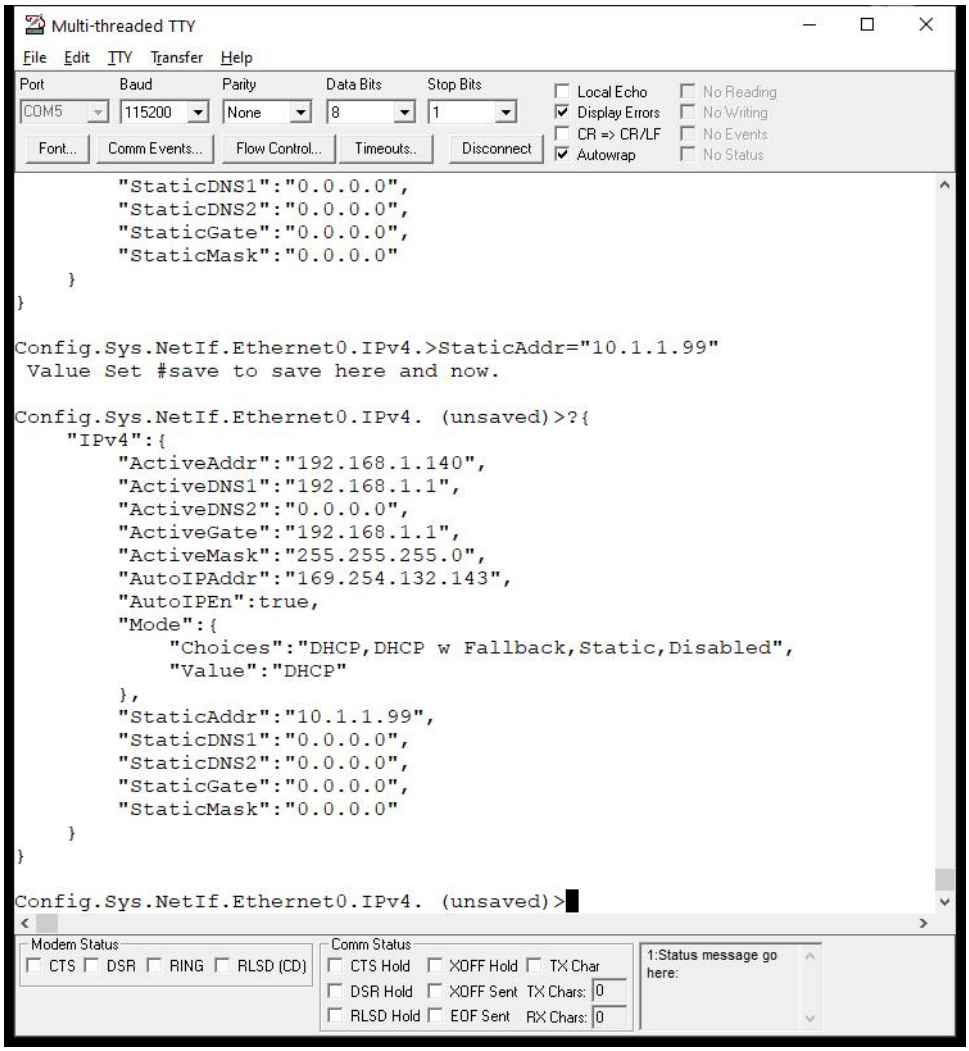


Figure 4.10 Set StaticAddr Value

In the same manor we will set the Mask, Gateway and DNS. To set the `Mode` we will need to either include the `Mode` level in the setting string or go to the `Mode` level, since the setting is a `Choices` selection and not a simple string like the static IP settings. To accomplish this from the IPv4 level type: `Mode.Choices="Static"`. If you navigate the to Mode level, the command would be: `Choices="Static"`. At this point the unsaved settings are:

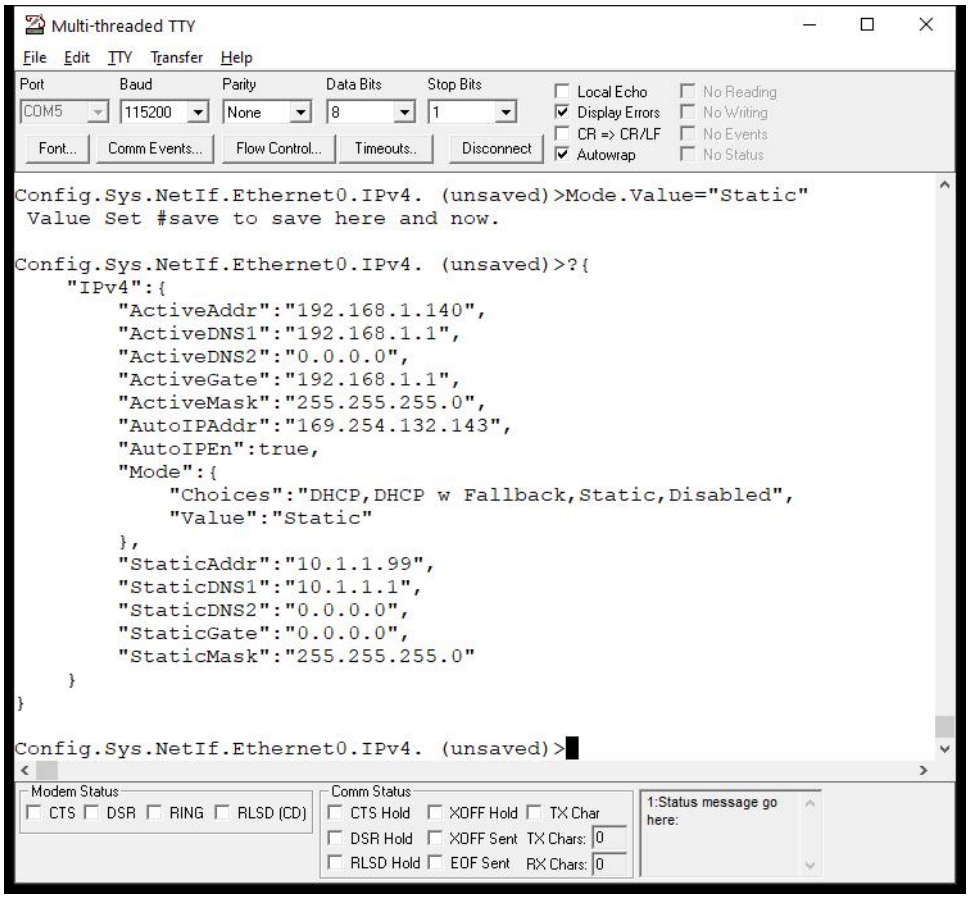


Figure 4.11 Static Summary

Warning

From the settings display you can see that the `Mode` selector control has two settings: `Choices` and `Value`. The `Choices` are values that appear in the web page selector control and you do not want to modify them. Only modify the `Value` to one of the `Choices` values.

To store the modified values in the Configuration Record in Flash memory, type `#save`. This will store the values, but the IP settings will not take effect until the device is rebooted.

Chapter 5

Migration Guides

There are currently three NetBurner development tool branches:

- 2.7.x, IPv4 only for all products except ARM.
- 2.9.x, IPv4/IPv6 dual stack for all products except ARM. This release will continue to be fully supported for ColdFire devices.
- 3.x, Latest release including IoT enhancements. Supports 5441x (MOD5441x, NANO54415, SB800EX) and ARM (MODM7AE70) platforms.

Note

If you are updating from a 2.7.x release to a 2.9.x release, please see the IPv6 guide in the `\nburn\docs` folder.

The following sections will describe the migration path in more detail.

5.1 Release 2.x to 3.x Overview

When developing NetBurner 3.x we incorporated customer requests to enable the use of the latest ARM base processors, security protocols, and IoT capabilities. For existing customers we have done our best to balance the additional capabilities with ease of migration.

5.1.1 Hardware

- Support for ARM processors, starting with the 300MHz M7.
- Support for NXP/Freescale 5441x based products: MOD54415, MOD54417, SB800EX, NANO54415. For other NetBurner platforms we will fully maintain the 2.9.x branch.
- Applications easily scale from single chip solutions to large memory platforms.

5.1.2 Configuration

A new configuration and update system has been created:

- Updates, discovery and configuration are now platform independent and do not require any computer OS specific utilities.
- Devices can be discovered by going to "discover.netburner.com" in a web browser.
- Applications can be updated from a web browser, the nbupdate utility, NBEclipse, WGET, or the serial interface.
- Configuration can now be done with a JSON object. These objects can be saved or uploaded, making configuration in a production environment much easier and faster.
- Developers can easily crease their own configuration objects.
- The IPSetup windows utility can be used to locate a device, but not configure it.

5.1.3 RTOS

The low level code of the RTOS has been modified to support new features including:

- Increased the number of tasks from 64 to a user defined value.
- OS functions are now managed as C++ objects.

5.1.4 TCP

- Increased the maximum number of sockets from 32 to a user defined value. A Maximum value over 256 may have an impact on performance.

5.1.5 Network Interfaces

- Many new interfaces supported
- Access to interfaces now requires looking at an Interface Object, rather than global variables such as ETHERNETIP. This was needed to support additional interfaces, a larger number of interfaces, and IPv6 capability.

5.1.6 SSL/TLS

- Updated to the latest protocols and ciphers (2.9.x also has the SSL/TLS updates).
- The process for generating compiled-in certificates has been made easier, but is slightly different.
- Devices can now generate certificates.

5.1.7 System Initialization

- A new function, init(), is now used to initialize stack, all interfaces, and network debugging.

5.1.8 NBEclipse

- Updated to Oxygen
- Can generate makefiles that can be used on command line builds.

5.1.9 GCC

- Updated to 8.1

5.2 Release 2.x to 3.x Porting Guide

5.2.1 Device Configuration

Previously device configuration was accomplished with IPSetup, a windows based program. In 3.x there are a number of ways to configure the device that are independent of the host computer type. The primary method is through the device's own configuration web server. In this way the device can be configured by a web browser on any platform. There is also a serial interface configuration method.

- Can be configured by any web browser, and applications can be programmed into the device. *Note:* Autoupdate and IPSetup are no longer used to download applications or configure the device. However, you can still use IPSetup to discover a device's IP address.
- Devices can be found by typing `discover.netburner.com` in the web browser URL field.
- The underlying configuration mechanism is handled through a JavaScript Object Notation (JSON) object. This will be described in more detail later, but the important point is that it provides a simple interface that can be used by non-embedded developers (e.g. web developers) to interact and configure the device. The implementation is also designed to enable you to add any application specific information to be stored and accessed in the same manor, such as calibration values, set points, etc.
- You are able to create your own configuration web interface that will appear in place of the default interface. In this way you can organize and present data in the appropriate way for your customers, as well as add your own custom logo, images and descriptions. Examples are provided.

5.2.2 Configuration Procedure

- Connect your NetBurner device to a network with Internet access.
- Open a web browser on your host computer and go to `discover.netburner.com`. You will see a list of devices on your network. 
- The "Device" link will take you to the device's web page. The "ConfigPage" link will take you to the device's configuration web page as shown below: 

A sample JSON object is shown below:

```
{
    "Config":{
        "AppData":{},
        "Sys":{
            "Application":"Simple HTML Example",
            "Boot":{
                "Abort":"A",
                "BootBaud":115200,
                "BootDelay":2,
                "BootQuiet":false,
                "BootUart":0,
                "Password":"",
                "SerialConfig":{
                    "Choices":"DuringBoot, AlwaysEnabled, PauseAfterBoot, Disabled", "Value":"DuringBoot"
                },
                "User":""
            },
            "NetIf":{
                "Ethernet0":{
```

```
                    "DeviceName":"",
                    "DhcpDiscoverSec":1,
                    "DiscoveryObfuscate":true,
                    "DiscoveryReportInterval":900,
                    "DiscoveryReportUrl":"discover.netburner.com/DevicePost",
                    "IPv4":{
                        "ActiveAddr":"10.1.1.73",
                        "ActiveDNS1":"10.1.1.1",
                        "ActiveDNS2":"0.0.0.0",
                        "ActiveGate":"10.1.1.1",
                        "ActiveMask":"255.255.252.0",
                        "AutoIPAddr":"169.254.131.245",
                        "AutoIPEn":true,
                        "Mode":{
                            "Choices":"DHCP,DHCP w Fallback,Static,Disabled", "Value":"DHCP"
                        },
                        "StaticAddr":"0.0.0.0",
                        "StaticDNS1":"0.0.0.0",
                        "StaticDNS2":"0.0.0.0",
                        "StaticGate":"0.0.0.0",
                        "StaticMask":"0.0.0.0"
                    },
                    "IPv6":{
                        "ActiveAddr":["2602:306:b8e9:c83f::14c0","2602:306:b8e9:c83f:203:f4ff:fe0b:83f5","fe80::203:f4ff:fe0b
                        "ActiveDNS":["2602:306:b8e9:c83f:208:a2ff:fe0c:b081","2602:306:b8e9:c83f:208:a2ff:fe0c:b081"],
                        "ActiveRoute":["fe80::1:1"],
                        "Mode":{
                            "Choices":"DHCP,DHCP w Fallback,Static,Disabled", "Value":"DHCP"
                        },
                        "StaticAddr":"::",
                        "StaticDNS1":"::",
                        "StaticDNS2":"::"
                    },
                    "MAC":"00:03:F4:0B:83:F5"
                }
            },
            "Platform":"MODM7AE70"
        },
        "Version":8,
        "Reboot":false
    }
}
```


If you were writing a web interface and need only the data under Ethernet in the tree, it can be accessed by: http://10.1.1.71:20034/UI.html?CONFIG/SYS/NETIF/Ethernet

To obtain just the IP address: http://10.1.1.71:20034/UI.html?CONFIG/SYS/NETIF/↩ Ethernet/IPv4/ActiveAddr

5.2.3 Recommended Examples

The following examples are recommended to begin evaluating the platform. They are located in the `\nburn\examples` directory.

- ShowInterfaces
- `\Configuration\Web\BasicWebConfig`

5.2.4 Configuration Security

One of the features of NetBurner 3.0 is that each device is configured though its own configuration web interface. Data sent and received through the network interface has three options for security:

- Open: No security, plain text
- Obfuscated: Encrypted, but without TLS and a secure certificate and private key
- Secure: Encrypted with TLS and a certificate and private key

Each NetBurner device generates a default certificate and private key, which is stored in its internal flash memory, which can be used to secure the connection as a self-signed certificate. You can also use the device's private key to create your own externally self-signed certificate, or have a certificate issued by a recognized third party. The externally signed certificate can then be installed by overwriting the device's default key.

5.2.4.1 Cautions When Using the Jumper Recover Procedure on a Device

Each device has a location to insert a shorting jumper to reset the module in the event of a catastrophic software problem, such as downloading an application image that corrupts flash memory or crashes. This recovery procedure will erase all of flash memory and download a recovery image. A new private key and certificate will then be generated. In such a case, any external self-signed or third party certificates will no longer be valid since the private key will have changed. A new certificate will have to be created from the new private key.

5.2.5 Device Application Updates

Note

The AutoUpdate utility is no longer used for download application to your device.

You will notice there is an Update Application on the web page. Application are now uploaded using the web interface. Alternatively, there is a windows executable that can be run from the command line in `\nburn\pcbin\nbupdate.exe`. Application images are now `.bin` files, they are no longer `.s19` files. If building in NBEclipse the `.bin` file will be located in your workspace project's release folder. The format using `nbupdate.exe` is as follows: `nbupdate <application .bin file> <device IP>`

5.2.6 Eclipse and Development Tools Updates

Summary of new and improved features:

- Eclipse has been updated to the Oxygen version. It also takes advantage of multiple processor cores on the host computer to greatly speed up compile times. *Note:* if using the command line to build you can use make `-j` to use multiple cores.
- GCC has been updated to version 8.1.
- Each project now has its own NetBurner library. Now any system changes you make will be specific to the project, and each project can have its own modifications.
- The makefile implementation is now identical for NBEclipse and command line builds.

5.2.7 SSL/TLS

Improvements include:

- Moving from NetBurnerSSL to wolfSSL
- Support for a much broader range of ciphers
- Support for DER encoded certificates
- Support for server-side peer verification
- Support for onboard certificate generation
- Substantial improvements to SSL stability
- Support for client side certificate authority lists
- A cleaner interface for using certificate authority lists, as well as the ability to dynamically alter them during runtime
- Faster key generation for onboard certificate generation
- Lays the groundwork for future improvements
  - TLS v1.3
  - FIPS 140 Ready
  - DTLS

5.2.8 Notes on Porting From Previous Revisions

- Big endian vs. little endian. The ColdFire processors have a bin endian architecture, while ARM processors are little endian. If you have code that manipulates 16-bit or 32-bit numbers that relay on a big endian format you will need to ensure it works for both types.
- The typical startup sequence of function calls: `IntializeStack()`, `GetDhcpIfNecessary`, `EnableAutoUpdate()`, etc have been replaced with a single call: `init()` which is declared in the header file `init.h`. You will need to remove header files such as:
```
#include <startnet.h>
#include <autoupdate.h>
#include <dhcpclient.h>
#include <NetworkDebug.h>
```
- The real-time operating system has been modified to increase performance and capabilities. The header file `ucos.h` is now `nbrtos.h`. Any RTOS function calls that had `UCOS` in related names have been modified to reflect this.
- `UCOS_ENTER_CRITICAL` has been replaced with `NBRTOS_ENTER_CRITICAL`.
- The `StartHTTP()` function is now `StartHttp()` and `StartHttps()`.
- Callbacks have been added to replace the MyDoGet() and MyDoPost() functions. See HTML Processing for examples on usage.
- A new function has been added to wait for an active network connection, meaning the device has link. A typical startup sequence will be:
```
init();
WaitForActiveNetwork(TICKS_PER_SECOND * 10);
StartHttp();  // or StartHttps();
```

- Types such as `WORD`, `DWORD`, etc have been replaced with `uint16_t`, `int16_t`, `uint32_t`, `int32_t`, etc.
- Registering a name with a DHCP server is no longer done by assigning a string to `extern const char pDHCPOfferName`. It is now set by assigning a name to the configuration tree JSON object element: `CONFIG:SYS:NETIF:Ethernet:DeviceName`.
- Runtime calls to the network interface such as `EthernetIP()` have been removed. Please refer to the ShowInterfaces example for methods to obtain runtime interface data.

5.2.9 Porting I2C From MCF52xx Based Products to MODM7AE70

Applicable Platforms:

- MOD5234
- MOD5270
- MOD5272
- MOD5282
- PK70EX
- SB70
- SB70LC
- SB72

MCF52xx platforms have a singe I2C peripheral, therefore the I2C driver API knew which signal pins to use and the set of functions to send/receive data only had to consider a single peripheral. ARM based platforms have many I2C peripherals, so a different type of driver/API is required. For example, with just a single I2C peripheral, an I2CSend() function is all that is needed. If there are many I2C peripherals, the driver must be able to send on any of them. This applies to all I2C functions: read, write, status, error detection, etc.

The most robust and scalable implementation is to use I2C peripheral C++ objects. You do not need to be a C++ programmer to use these objects in your application, other than to understand the format to call a member function of the object instead of a global C function call. For example, if in C you had a global function named GetI2CStatus(), it would now be a member function of a specific I2C object/peripheral and the call would be MyI2CObject.GetI2↩CStatus(). This method enables your application to handle multiple I2C peripherals, and the status function knows which peripheral to use because it operates on the object. Note that since the status function is called as a member function of the object, it is common to take that into consideration in the member function name, so it would likely be MyI2CObject.GetStatus().

In this way you can easily handle multiple I2C peripherals: MyI2CObject1, MyI2CObject2, MyI2CObject3, etc. Your application code does not need to manage and continuously check for which peripheral is being used, such as having a C style array of I2C peripherals.

The I2C driver for MCF52xx devices in the NNDK 2.x tools has the interface listed below. Note that there is a bit of C++ even in the old driver in terms of the function prototypes. You are likely calling some of these functions without any parameters, or with less than all the parameters listed. In C++ default parameters are specified in the function prototype with the '=' sign. So you can call I2CInit() with no parameters in your application, and the default parameters in the function prototype will be used.

```
void I2CInit( BYTE slave_Addr = 0x08, BYTE freqdiv = 0x3C )

BYTE I2CSend( BYTE val, DWORD ticks_to_wait = I2C_RX_TX_TIMEOUT )
BYTE I2CSendBuf(BYTE addr, PBYTE buf, int num, bool stop = true)

BYTE I2CRead( PBYTE val, DWORD ticks_to_wait = I2C_RX_TX_TIMEOUT )
BYTE I2CReadBuf( BYTE addr, PBYTE buf, int num, bool stop = true )

BYTE I2CStart( BYTE addr, bool Read_Not_Write, DWORD ticks_to_wait = I2C_RX_TX_TIMEOUT )
BYTE I2CStop( DWORD ticks_to_wait = I2C_RX_TX_TIMEOUT )
BYTE I2CRestart( BYTE addr, bool Read_Not_Write, DWORD ticks_to_wait = I2C_RX_TX_TIMEOUT )

void I2CResetPeripheral()

bool  I2CRXAvail()
DWORD I2CTXAvail()
BYTE  I2CGetByte()
```

The I2C implementation in the 3.x tools for ARM platform uses the I2CDevice Class. For example, on the MODM7↩AE70, an I2C object to select an I2C peripheral to interface with an EEPROM can be declared as (code taken from the I2C Pic Kit example):

```
#define I2C_MODULE_NUM      0           // use I2C0/TW0
#define I2C_EEPROM_ADDRESS  (0xA0 » 1) // Microchip 24LC0B EEPROM, 2Kbit (256 x 8)
I2CDevice EepromDevice(i2c[I2C_MODULE_NUM], I2C_EEPROM_ADDRESS);
```

The declaration of the I2CDevice creates an I2C object for I2C peripheral module 0 at the specified EEPROM I2C address. Creating the object also handles the initialization, so the equivalent of I2CInit() is not needed. Once we have created the object, all we need to do is call the member functions to control it in the format: `Eeprom↩Device.<member function>`. For example, to send/write a single byte value of 0x60 to address 0x01 of the EEPROM: EepromDevice.writeReg8(0x01, 0x60);. Note that in ARM terms, the I2C addresses are referred to as registers. writeReg8() refers to 8-bits.

You will also note that the return values use the C++ operator "::". Whereas the 2.x driver returned an unsigned 8-bit value (BYTE), the 3.x driver returns a typed value of Result_t. Using typed values will result in more robust application code since the compiler can detect when a return value is used incorrectly. the "I2C::" result types are (be sure to view the latest information in the 3.x NetBurner API manual):

```
I2C_RES_ACK     Acknowledged
I2C_RES_NACK    Not acknowledged
I2C_RES_ARB_LST Arbitration listening
I2C_RES_BUSY    Bus is busy
I2C_RES_ARG     Bad argument
```

The corresponding functions to the MCF52xx I2CDevice driver are below. This is the recommended driver.

```
// Constructor to create object, also initializes the I2C peripheral. The pInterface
// parameter will be i2c[0], i2c[1] or i2c[2], representing the 3 I2C peripherals.
// The optional 3rd parameter is normally left as a default. It can be used to set the
// number of register address byte to send, from 0 to 3.
// Example: I2Cdevice EepromDevice(i2c[0], address);
I2CDevice(I2C & pInterface, uint8_t deviceAddress, uint8_t numAddressBytes = 1)

Result_t writeReg8(uint32_t reg, uint8_t dat)
Result_t writeRegN(uint32_t reg, uint8_t *buf, uint32_t blen)

Result_t readReg8(uint32_t reg, uint8_t &dat)
Result_t readRegN(uint32_t reg, uint8_t *buf, uint32_t blen)

Start, Stop and Restart are taken care of in the writeRegN and readRegN functions.
In situations in which a restart was used for sending more than 1 byte as an address,
the numAddressBytes parameter in the I2CDevive constructor can be used.

void resetBus()
void setup(uint32_t busSpeed)  // change bus speed
```

If you wish to have more manual control over the I2C transactions, you can create an I2C object instead of an I2CDevice object:

```
// Create an I2C peripheral object. The module parameter for the MODM7AE70 can be 0, 1 or 2.
I2C(int module)

Result_t writeReg8(uint8_t devAddr, uint32_t reg, uint8_t data)
Result_t writeRegN(uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen)

Result_t readReg8(uint8_t devAddr, uint32_t reg, uint8_t &data)
Result_t readRegN(uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen)

void setup(uint32_t busSpeed)

Result_t DoTransaction(I2CTxn_t *pTransaction, bool bRepeatedStart = false);

void resetBus()
```

In NNDK 3.x there is also a third option. ARM refers to the I2C as a two-wire interface, or a wire interface in general. A WireIntf class is available for even more manual control. It was written for those familiar with Arduino drivers, but can be used for more detailed control in general. Please refer to the NNDK 3.x documentation for details.

5.3 Upgrade 5441x Platforms to 3.x

5.3.1 Introduction

The ColdFire 5441x platforms were created before NetBurner 3.x, but they can be easily upgraded. The primary difference is in the device configuration. For reverse compatibility with prior tool sets, all 5441x devices are shipped with a factory application built with the 2.x release so they work with the IPSetup and AutoUpdate utilities. Once you download an application created in NetBurner 3.x, the device will then have the advantages of the 3.x features. The upgrade procedure consists of first loading a conversion application as described below. Thereafter you can load 3.x applications directly.

5.3.2 5441x Product Models Numbers

- MOD54415
- MOD54417
- NANO54415
- SB800EX

5.3.3 3.x Update Procedures

There are 3 procedures that can be used depending on the production date of your 5441x device. For reverse compatibility all 5441x devices ship from the factory running a 2.x application. All are described in more detail in the following sections.

1. For devices manufactured January 2022 or later, the 2.x factory application can process 2.x or 3.x application images, so download your 3.x application using the AutoUpdate utility.
2. For devices manufactured prior to January 2022, use the AutoUpdate utility to download a 2.x to 3.x conversion application.
3. For devices manufactured January 2022 or later, abort to the Alternate Monitor and download either a 2.x or 3.x application. Note that this procedure is also used to revert back to a 2.x compatible device.

Attention

Once a device is running a 3.x application, you will no longer use the 2.x utilities IPSetup and AutoUpdate. Instead, you will use the methods described in the Device Discovery and Configuration section of this manual.

Attention

The NetBurner development tools for 5441x platforms produce both S-Record (.s19) and binary (.bin) application images. The 2.x compatible AutoUpdate utility mentioned in the migration section only uses .s19 files. The preferred file format going forward for all 3.x applications is the .bin format.

5.3.3.1 Procedure For a New 5441x Device Running The NetBurner Factory App

New devices from the factory are programmed with a 2.x factory application to be compatible with older designs. The application can process both 2.x and 3.x applications. The procedure is to identify the device and download a 3.x application.

- Connect the NetBurner device to the network and verify you can see the device's IP address with the AutoUpdate utility.
- Select your 3.x application and update.

5.3.3.2 Procedure Using the 2.x to 3.x Conversion Application: xxxx-3p0-Update_APP.s19

This procedure can be used for a 5441x device that is currently running any 2.x application. The procedure is to identify the device, download a 2.x to 3.x conversion application, then download any 3.x application.

- Connect the NetBurner device to the network and verify you can see the device's IP address with the AutoUpdate utility.
- Use the Autoupdate utility to download the appropriate conversion application image for your particular platform, located in the folder: `\nburn\platform\<platform name>\original\`. For example, a MOD5441x application name would be: "MOD5441X-3p0-Update_APP.s19".
- You will now be able to create and download 3.x applications on the device.

5.3.3.3 Procedure Using Alternate Boot Monitor

As of April 2021, the Alternate Boot Monitor has the ability to process a 3.x image file (older revisions must use the previous procedure by loading the conversion application). Alternate Boot Monitor revisions must be the revision shown below or later:

Platform	Revision
MOD5441x	1.11
NANO54415	1.05
SB800EX	1.05

- Connect to the debug/console port of the device with a serial terminal such as MTTTY
- Reset or power cycle the device (or use the recovery jumper)
- Press the 'A' key to abort when prompted (or use the recovery jumper)
- Use the AutoUpdate utility to download your 3.x application image

Note

If you have a device with an earlier Alternate Boot Monitor revision and wish to update it (the Alternate Boot Monitor), please contact NetBurner Support at https://support.netburner.com.

5.3.4 Procedure to Revert Back to 2.x

If you wish to revert back to 2.x, there are two methods to abort the boot sequence and enter the Alternate Boot Monitor:

1. Using the serial port interface
2. Using the hardware reset jumper

5.3.4.1 Serial Port Procedure

- Connect to the RS-232 debug/console port of the device with a serial terminal such as MTTTY
- Reset or power cycle the device
- Press the 'A' key to abort when prompted to abort to the Alternate Boot Monitor
- Use the AutoUpdate utility to download your 2.x application image

5.3.4.2 Hardware Reset Jumper Procedure

- Insert the hardware reset jumper
- Reset or power cycle the device to abort to the Alternate Boot Monitor
- Use the AutoUpdate utility to download your 2.x application image

5.3.5 Note on IPSetup utility and 3.x Devices

The windows IPSetup utility will still identify the IP address of a 3.x device. However, it cannot configure the device. The IPSetup device window designates these devices with a prefix of NEWCONFIG.

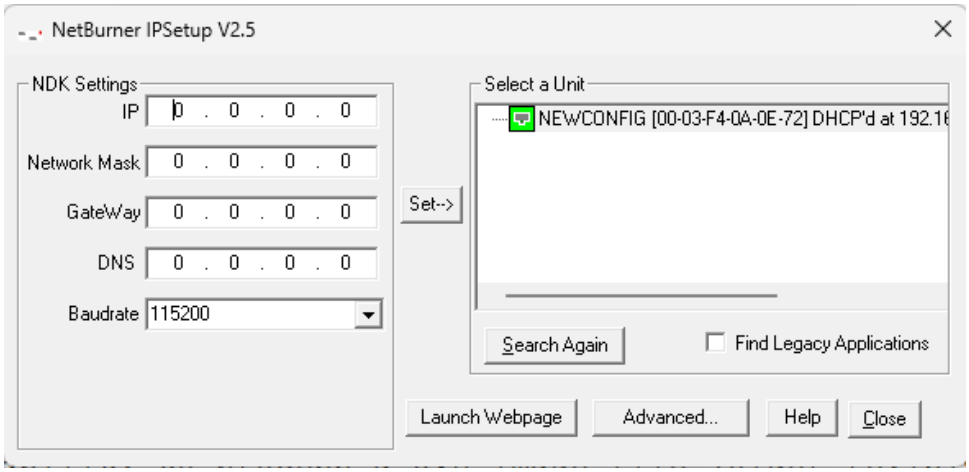


Figure 5.1 IPSetup with 3.x Application

Chapter 6

NBEclipse

There are three different ways to develop with the NetBurner Tools:

1. Install NBEclipse, which provides project management, a make system, and an editing environment.
2. From the command line using the make utility. Each example in the `\nburn\examples` folder provides and example makefile.
3. Use your own editing environment, and have that environment call the makefile for the project.

The documents in this section will describe how to use the NBEclipse environment and command line.

6.1 Getting Started Guide

6.1.1 Introduction

6.1.1.1 How to Use This Guide

The goal of the NetBurner Eclipse Getting Started Guide is to familiarize new users with the NetBurner Eclipse (NBEclipse) interface and basic operation. NBEclipse is based on the standard Eclipse release with additional features to support development with NetBurner hardware, such as the ability to locate your NetBurner device on the network, and download an application to the device's flash memory. Topics include:

- How to create a new project
- Import, compile, and execute example programs
- Debug an example program
- Use the application wizard to create a new program
- Various tips on useful features of the NBEclipse environment

6.1.1.2 Source Code for Example Programs

Source code for the examples in this document are located in the \nburn\examples directory of your NetBurner tools installation.

6.1.1.3 Hardware Setup

This document assumes you have a working hardware platform on which you can run the examples. Your NetBurner development kit hardware is pre-programmed with an example application at the factory. Before executing the examples in this document, it would be a good idea to connect your NetBurner device to your computer and verify you have serial and network communication with the device as described in the NNDK Quick Start Guide.

6.1.1.4 Software Installation

NetBurner software tools run on Microsoft Windows and OSX. When you receive your development kit there will be a bright red card with a keycode. Register this keycode at https://support.netburner.com to download the development tools.

6.1.1.5 Upgrading From a Previous Installation

Do not install over an existing version when upgrading from a previous version of the NetBurner tools. The upgrade requires that you either rename or uninstall the existing installation directory. The recommended procedure is as follows:

1. Backup any existing projects you want to save, especially if they are in the NetBurner tools installation directory.
2. Rename the existing tools installation folder. For example, if upgrading from revision 2.8.8, rename \nburn to \nburn2p8p8, then install the new version to \nburn. This way you can always get back to the previous installation and environment by renaming \nburn2p8p8 to \nburn.

6.1.1.6 Java Installation

NBEclipse requires the 64-bit Java Runtime Environment (JRE) revision 1.8 or higher. If there is no JRE installed, then you will receive an error message when starting NBEclipse stating that it cannot find the JRE executable. The JRE is free from http://java.sun.com. Download either the Java SE (Standard Edition) Development Kit (JDK) or Runtime Environment and follow the installation prompts. You may need to restart your computer after installation is complete.

Note

NBEclipse tools version prior to 3.0 require the 32-bit JRE.

6.1.1.7 Debug Port

Throughout this guide, we will refer to the "serial debug port". The serial debug port is one of the RS-232 ports that can be used to interact with your NetBurner device in the example programs. Stdout, stdin, and stderr are mapped to the debug port by default, so when you use functions like printf(), scanf(), gets(), they read and write to the serial debug port. Mapping of the stdio is configurable, as you can disable the serial debug port and use it as a general purpose UART, or reassign the stdio file descriptors to use other serial or network interfaces. Debugging in NBEclipse on network-enabled platforms will occur through the Ethernet connection, while debugging on non-network platforms will use the serial port.

6.1.2 Platform Overview

6.1.2.1 NetBurner Development Platform Choices

Your NetBurner Network Development Kit provides two options for development:

1. NBEclipse Integrated Development Environment (IDE)
2. Command line tools using the GNU make utility.

The choice of what platform you wish to use depends on personal preferences. The command line tools use makefiles, which are included for every example program and can be used as a starting point for your own makefiles. The make utility can also be invoked from within other IDEs and code editors such as Microsoft Visual Studio.

6.1.3 Project Management

The following points are useful to users who have not used a similar type of project manager. There are three primary methods to maintain and build a project:

1. Use NBEclipse to manage the project with the project files located in the NBEclipse workspace. This is the recommended method.
2. Use NBEclipse to manage the project with the project files located in an existing directory outside the NBEclipse workspace.
3. Configure NBEclipse to use an external makefile that you maintain.

6.1.3.1 NBEclipse Key Points

1. NBEclipse uses a "workspace" in which projects are created and source files are copied.
2. All source code files must reside in the project's "src" folder.
3. If using the web server, all HTML and associated files must reside in the project's "html" folder.
4. All recognized files in a project directory will be compiled. Recognized files have extensions of .c, .cpp, and .s (.s is an assembly language file).
5. The Project Explorer tab on the left side of the NBEclipse IDE is a navigation window like a file browser, it is not a file project manager. All recognized files will be built as part of the application. If you have non-project related files in your project directory and wish to see only the project files in the tab, you can select a filter to display only source code files.
6. To download an application to you NetBurner device, select "Run as NetBurner Application" from the project's Build Targets folder.
7. Application build output files such as .o, .elf, .map,.bin and .s19, and .bin will be written to project sub-directories named "Release" and "Debug" that correspond to the type of build executed. The .bin and .s19 files are the application image files.
8. The keyboard shortcut `<cntl> b` will build the active project.
9. The keyboard combination of `<cntl> <space>` invokes function auto-complete.

6.1.4 Creating Projects and Importing Files

This is the most common type of project creation. It will be used to build and run example programs, and is a good starting point for your own projects. By "empty project" we are referring to creating a project without running the Application Wizard to auto-generate a minimal application. After we create the project we will use the project import feature to import source code files from an example into the project.

1. Create an empty project.
2. Import source files by right-clicking on the project name and using the Import feature from the drop down menu.
3. Build the application and download to your NetBurner device.

6.1.4.1 Create a New Project

From the main menu, select `File->New->Project`:

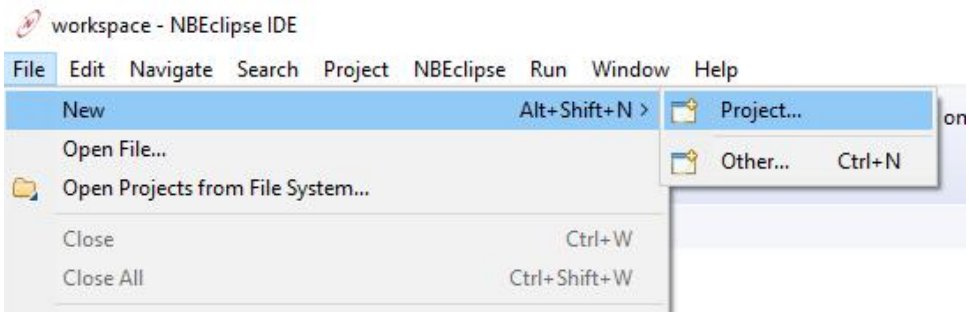


Figure 6.1 Create a New Project

6.1.4.2 Select the NetBurner Project Type

Select the NetBurner project type, then select Next:

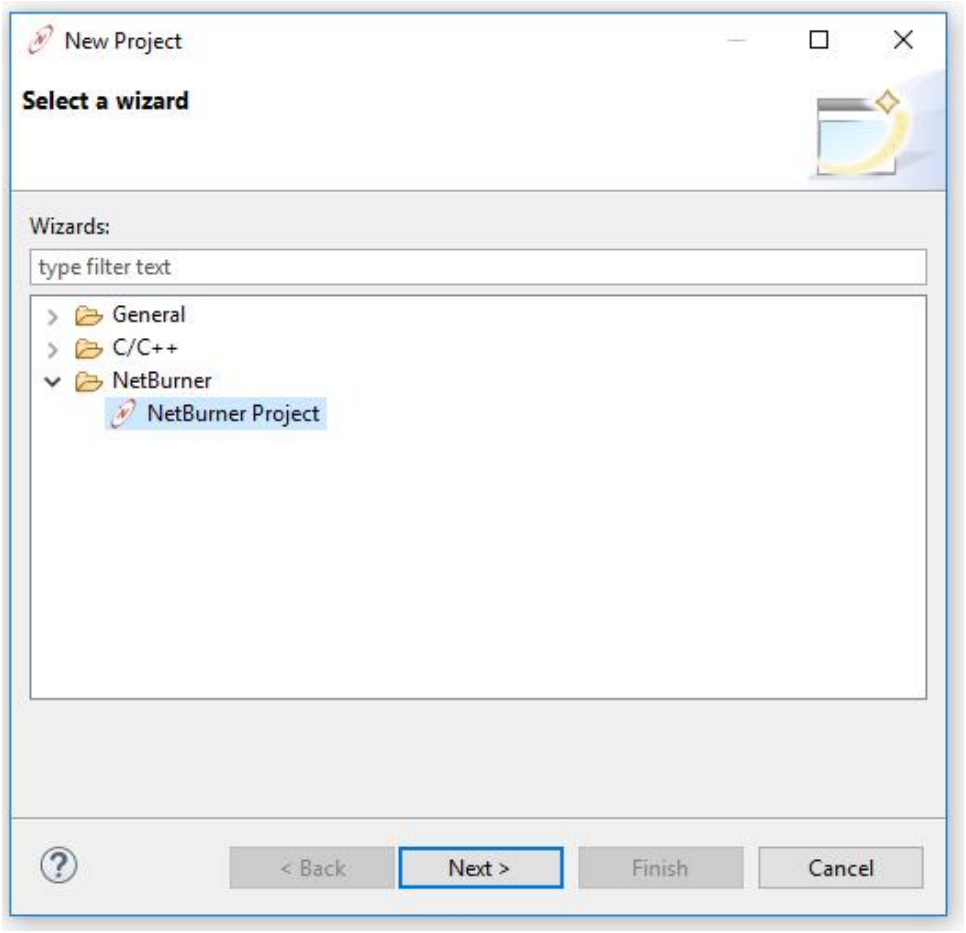


Figure 6.2 Select Product Type

6.1.4.3 Specify Project Name and Executable Option

Enter your project name in the Project Name field (it is best not to use spaces with gcc tool chains). In this example we entered "SimplHtml", since that is the example we will import. Make sure the Executable is set to: "NetBurner C++ Project".

Warning

Be sure to select NetBurner Device Executable, do not select the C/C++ option.

Note

You can create a project in a directory outside of your workspace by unchecking the Use Default Location checkbox, and using the Browse button to select the directory containing the source files.

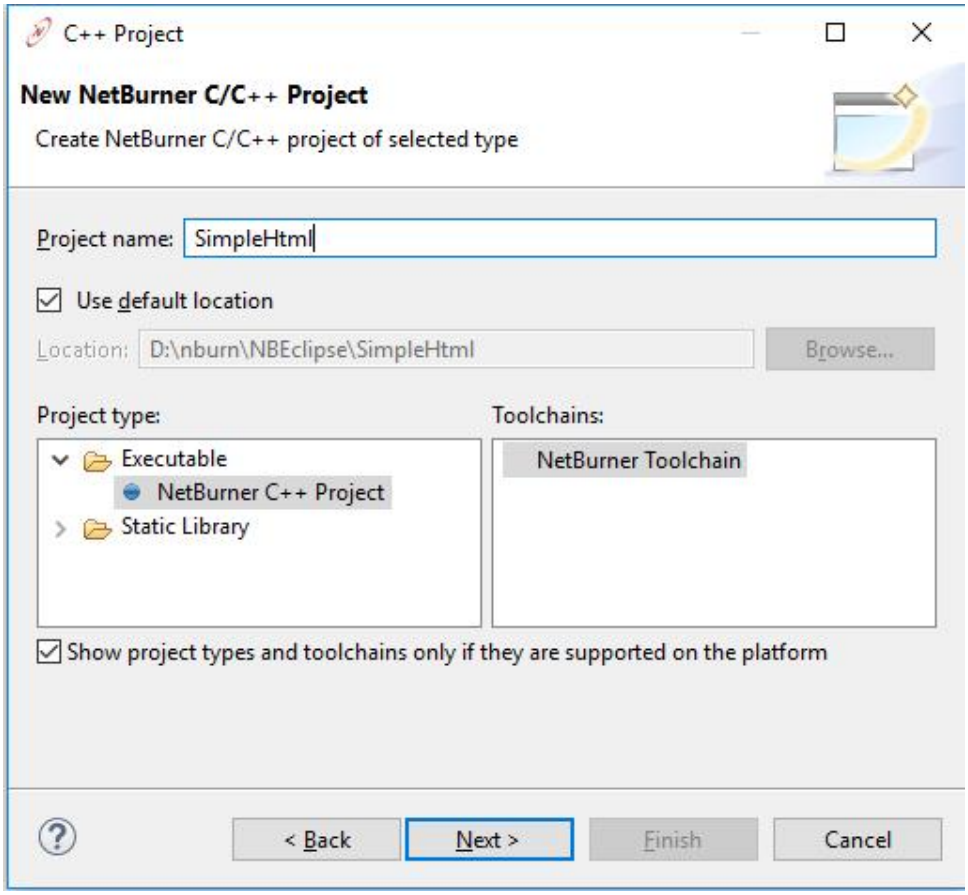


Figure 6.3 Select NetBurner Device Executable

6.1.4.4 Project Configurations

The default options for the project configurations are both Release and Debug. Leave the defaults and select Next.

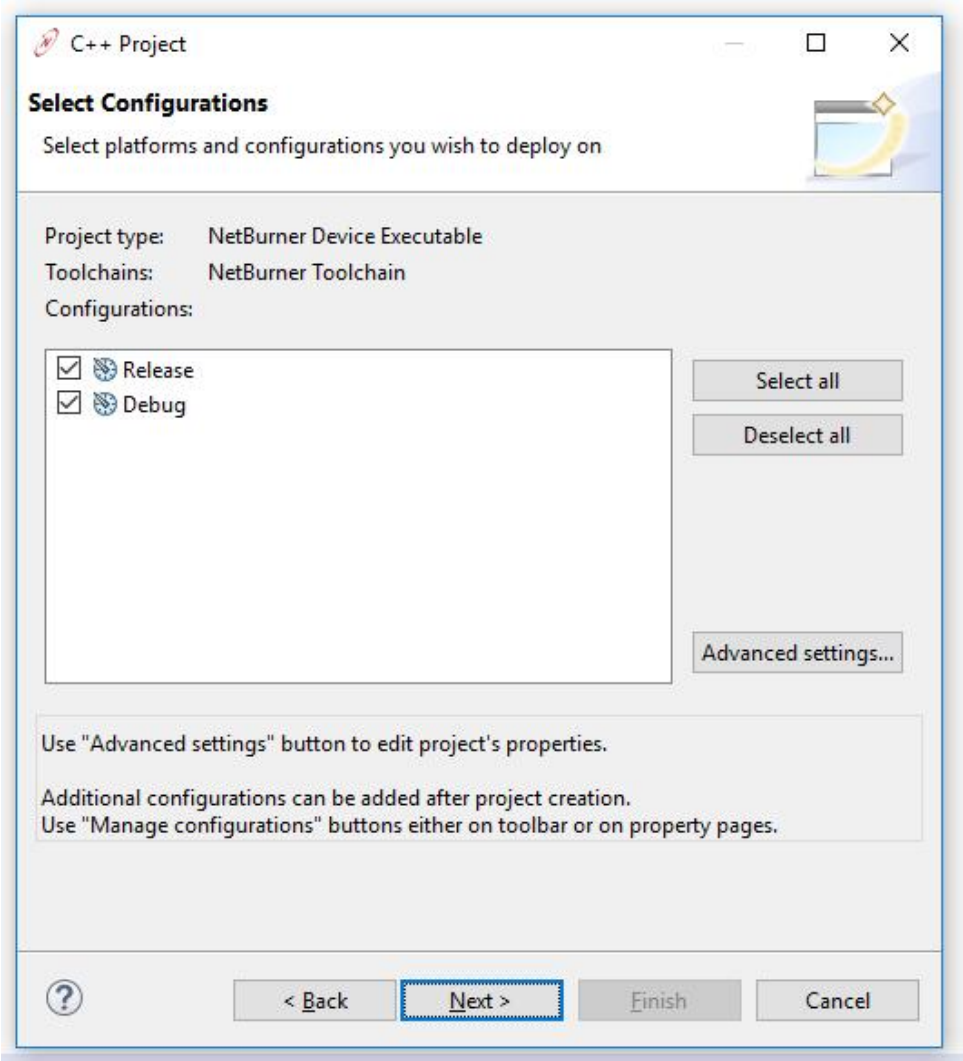


Figure 6.4 Project Configurations

6.1.4.5 Select Your NetBurner Target Device

There are two ways to select your target NetBurner device:

1. Click on the Search button to bring up a list of NetBurner devices on your network (device must be operational).
2. Manually: Click on the Target Platform drop down box to select your platform, then type in the IP address.

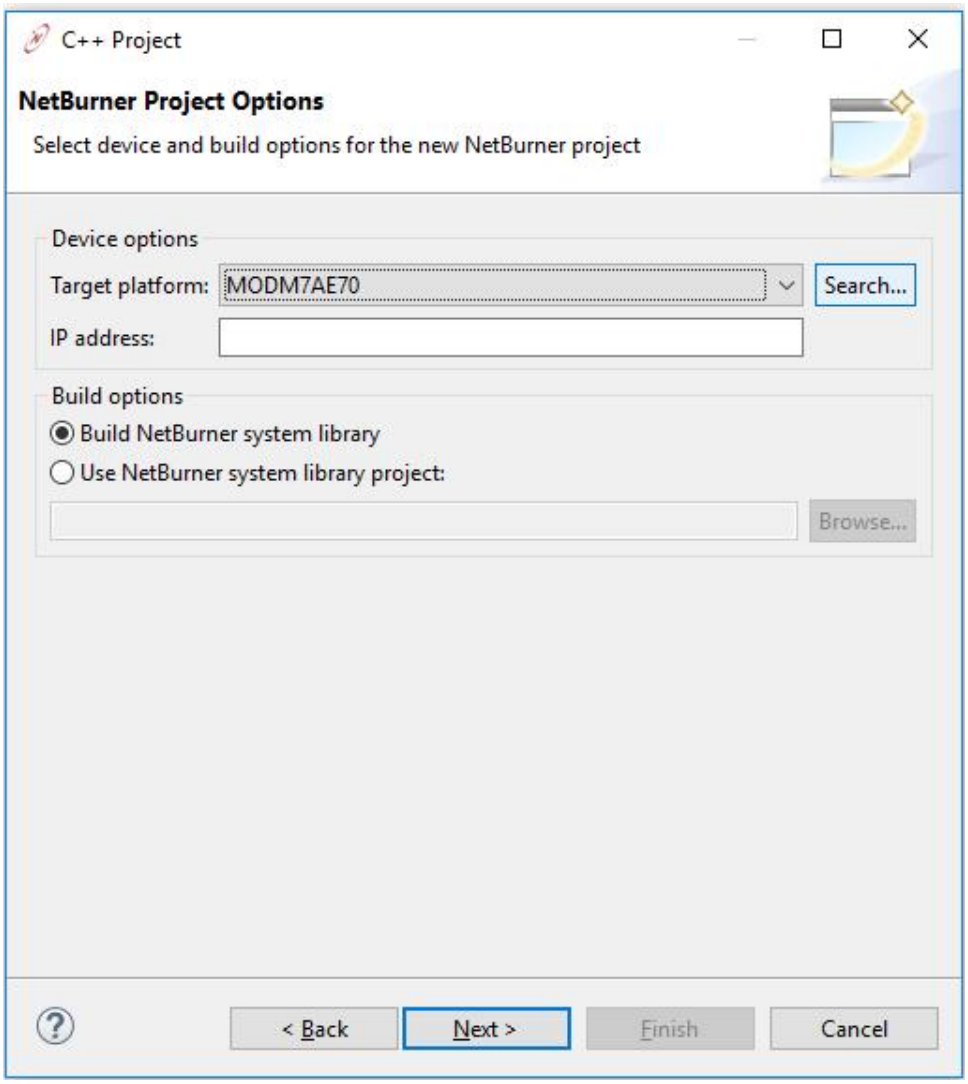


Figure 6.5 Search for devices

When using the Search feature, select the device in the list that corresponds to the MAC address printed on the label of your NetBurner device. In the image below we can see the device type, MODM7AE70, it's MAC address, and it's IP address. It will then fill in the platform and IP address for you. When done select Next.

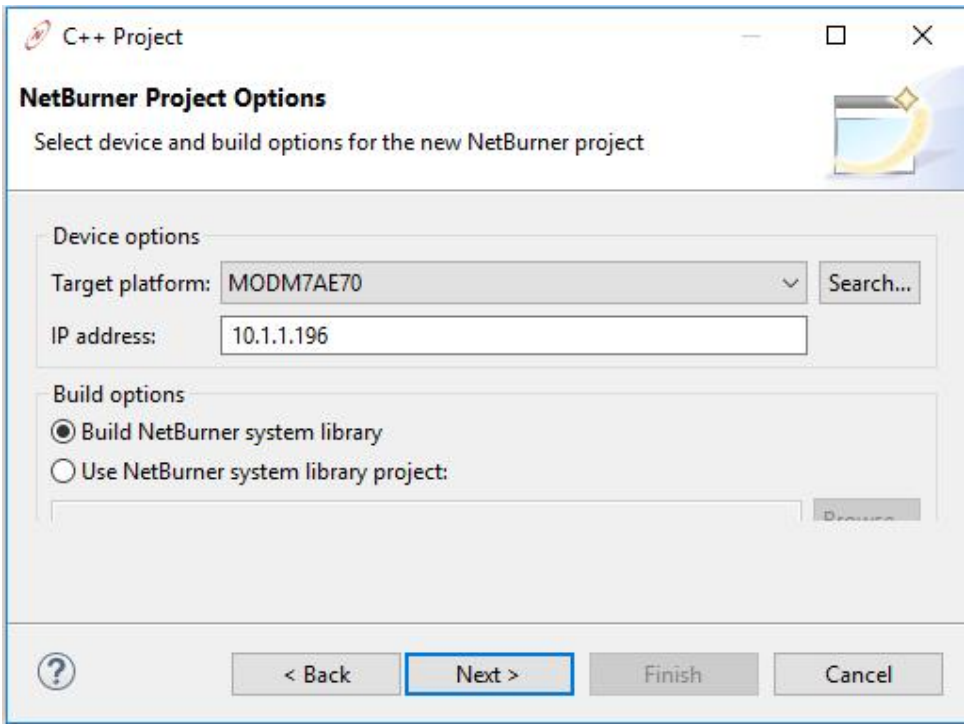


Figure 6.6 Successful search entry

6.1.4.6 Application Wizard

Since we are creating an empty project and will be importing our own source files we do not want to select any items here. However, if you wish to generate a minimal application using the Application Wizard, select both Standard Initialization and Web Server. This will generate a main.cpp source file to use as a starting point for your application.

Select Finish to complete the project creation.

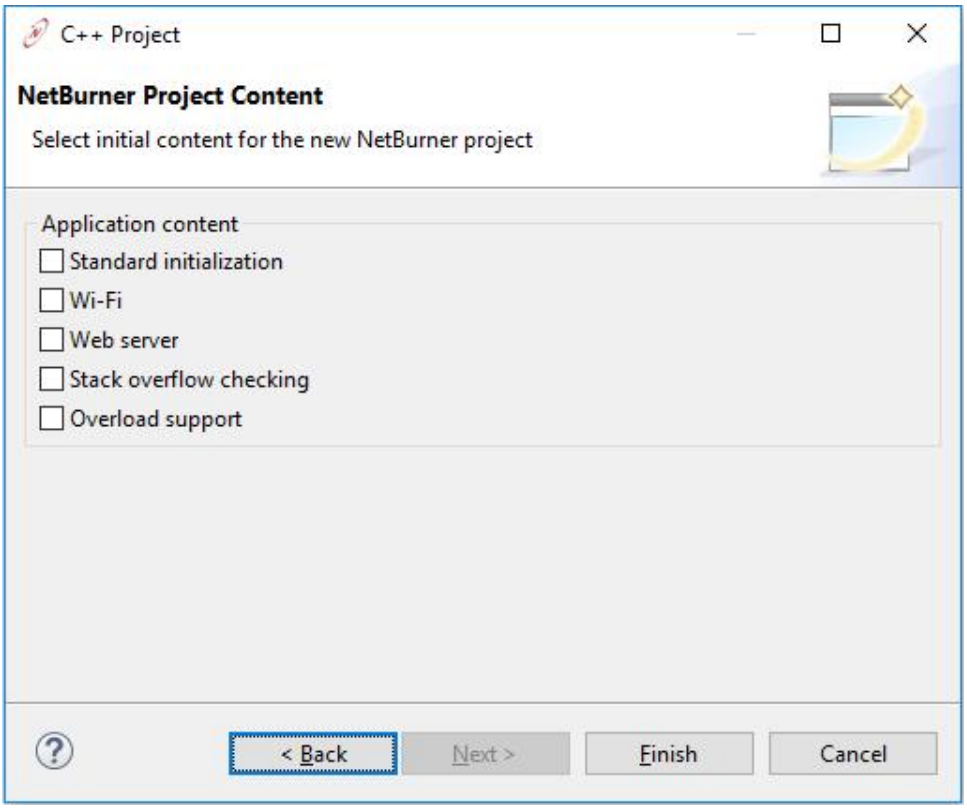


Figure 6.7 Skip Application Wizard code generation

6.1.4.7 New Project Created

You will now see your new project in the Project Explorer pane. At this point we have a project, but still need to import our source files (if you did not use the Application Wizard).

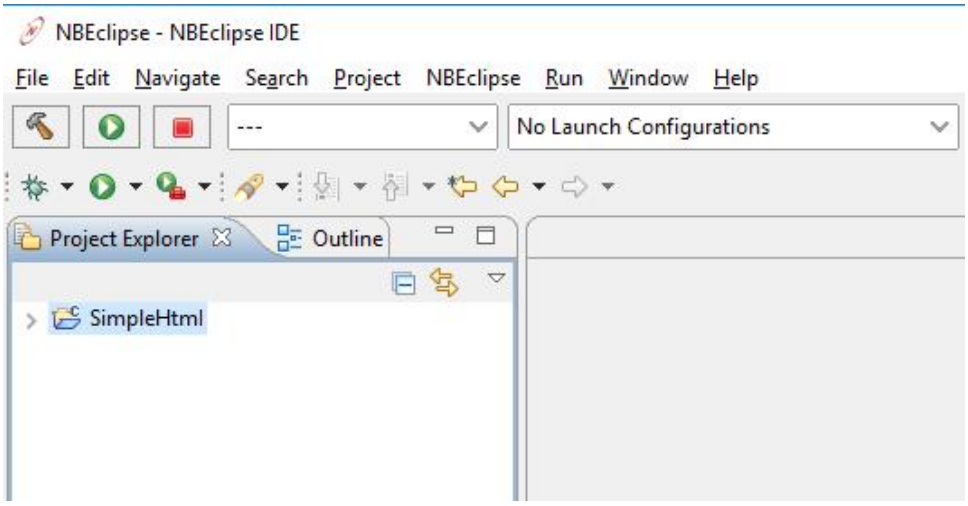


Figure 6.8 A new project has been created

6.1.4.8 Import Example Source Files to the Project SRC Folder

The next step is to import the source files of an example. NBEclipse uses a make utility to manage projects, and will attempt to build all source files in your project directory. Do not import non-source code files, such as .o, .elf, .map, .mk, auto-generated htmldata.cpp files, etc. NBEclipse will store all output files similar to these in project subdirectories named "Release" and "Debug" corresponding to the build type preformed.

For this demonstration we will use "\nburn\examples\Web\SimpleHtml". The first step is to right-click on your project in the NBEclipse Project Explorer pane:

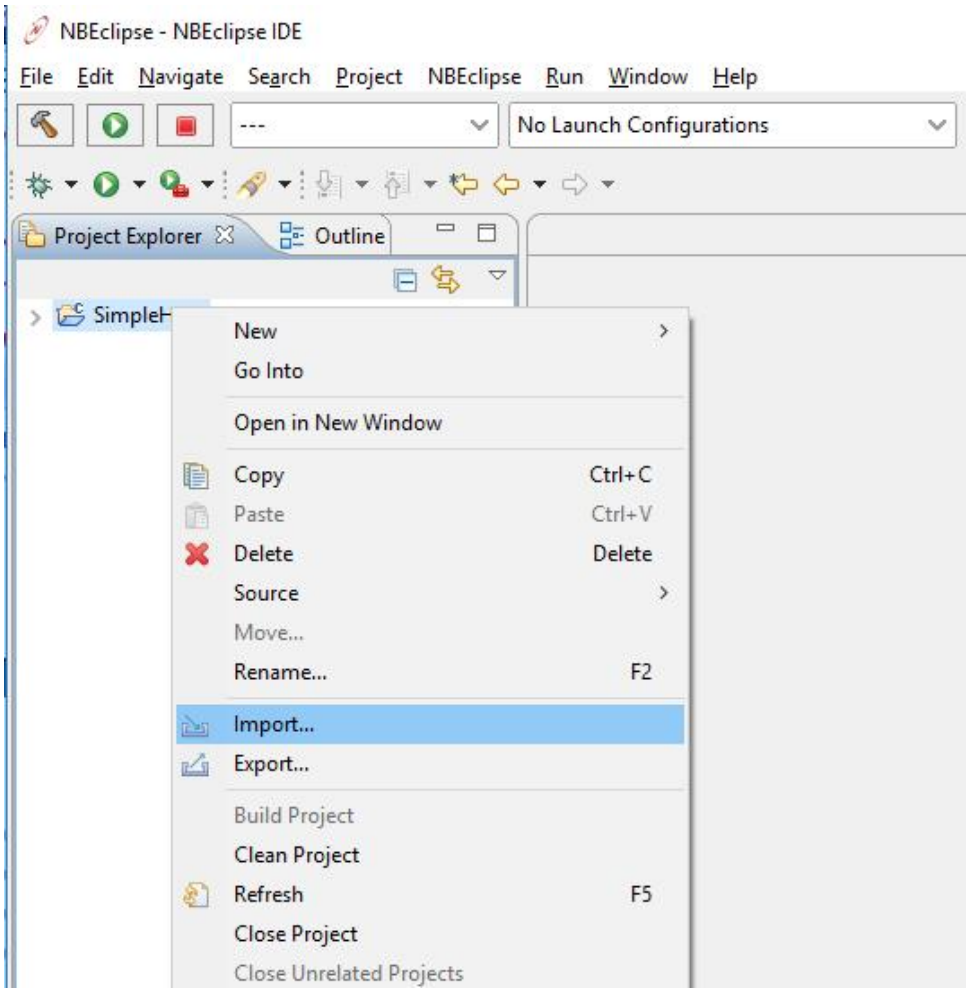


Figure 6.9 Select Import

Next select the General->File System, then select Next:

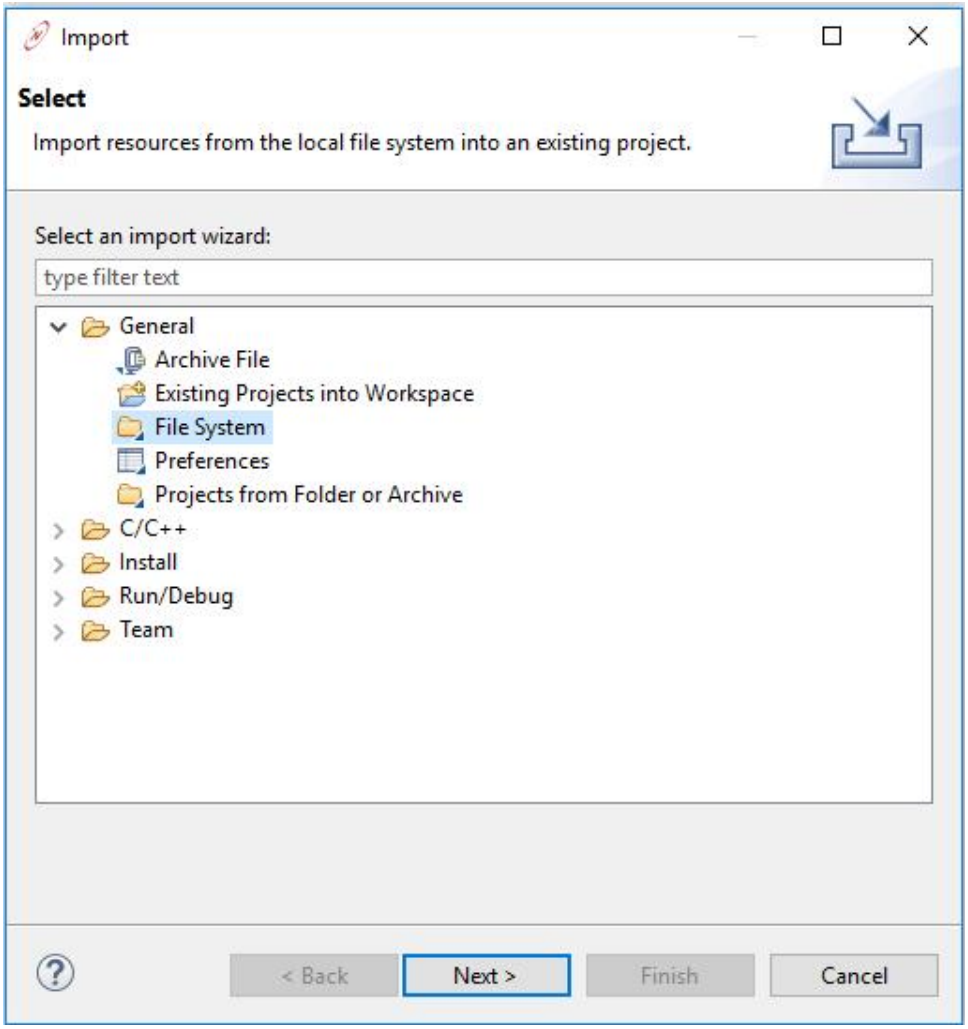


Figure 6.10 Select File System

Select the Browse button to bring up a file selection window, navigate to `\nburn\examples\Web\Simple↩Html`, and select the OK button:

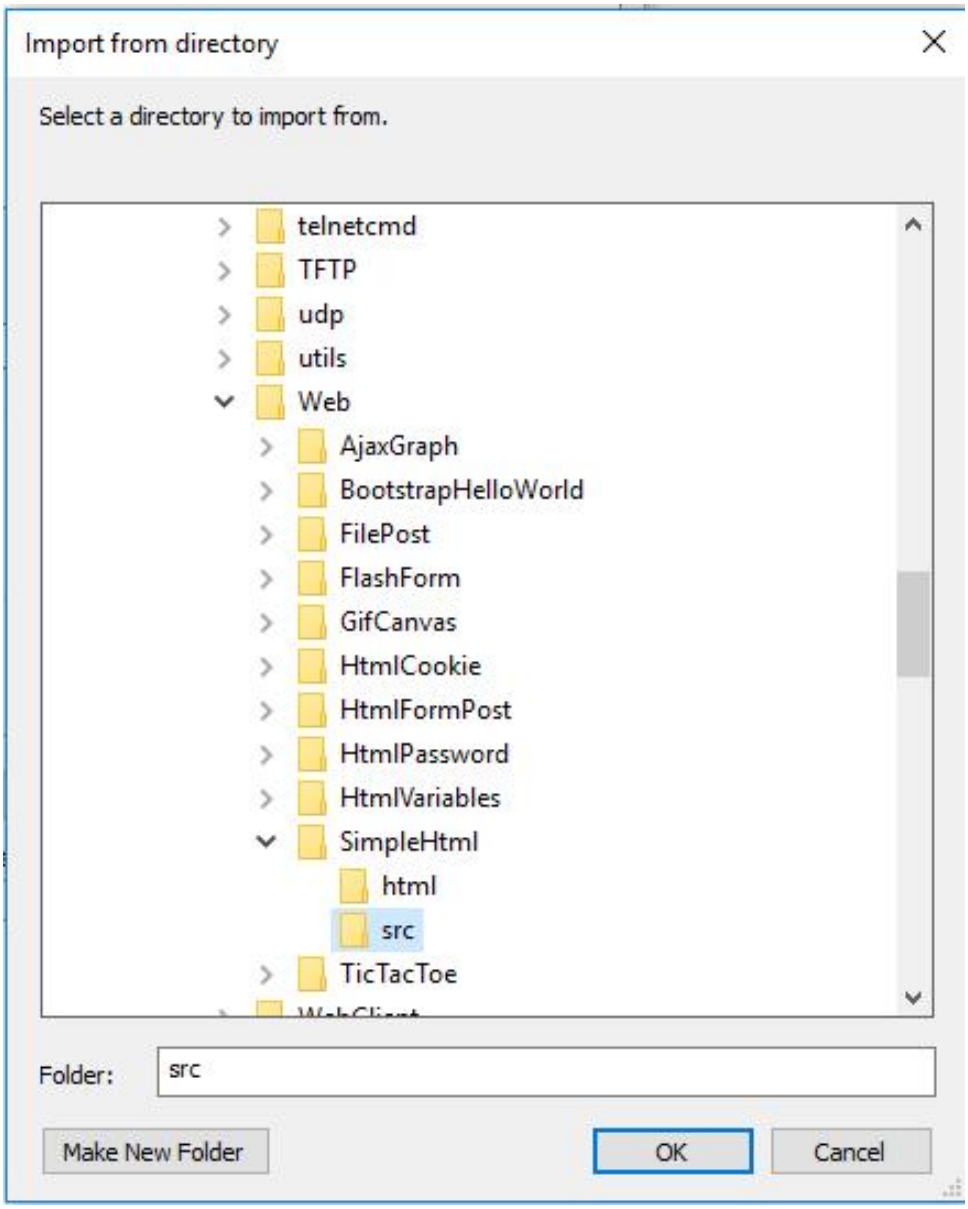


Figure 6.11 Select the Simple HTML Example

Once back at the File System dialog box, click on the ">" symbol next to the SimpleHtml name to open the tree structure. You can see that there is a folder named src and a folder named html. We want the files in both folders, so select the checkbox next to the name "SimpleHtml", which will select all folders underneath it. Note: the makefile you see is only used for command line builds, but it does no harm to import it as well.

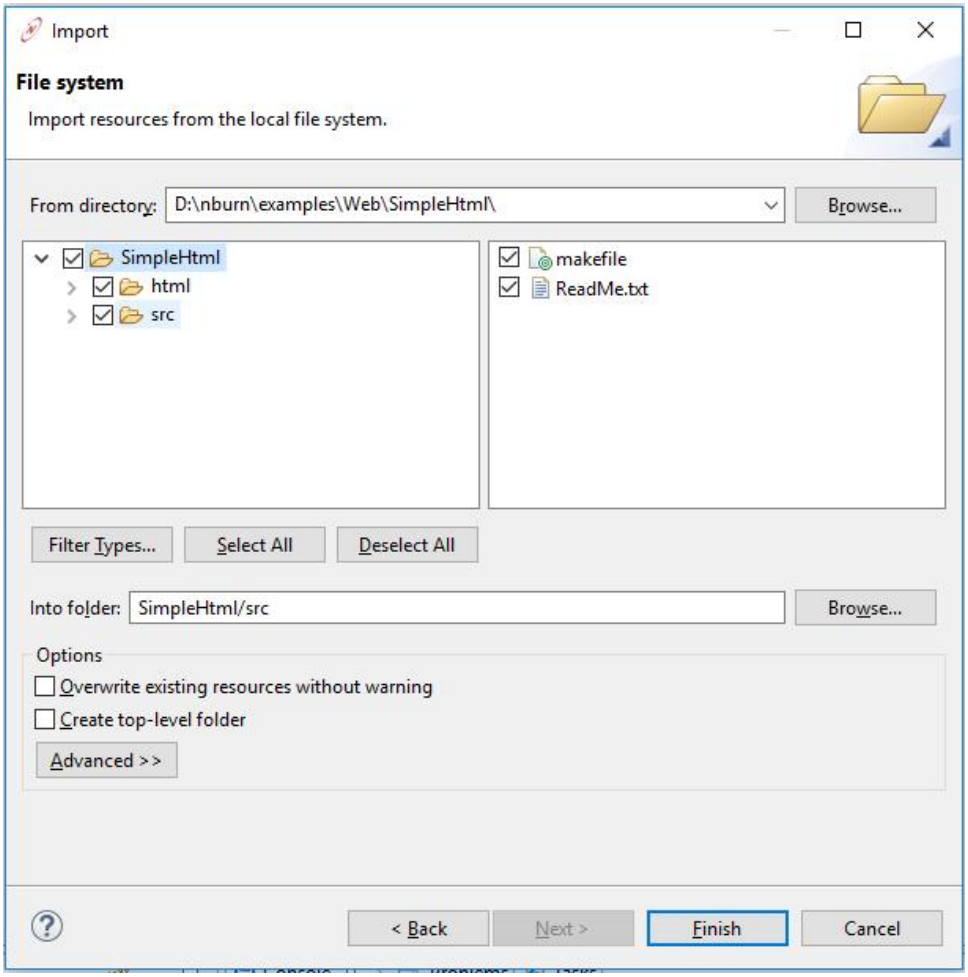


Figure 6.12 Select all SimpleHtml files for import

Select Finish, which will import the files and automatically build the project. You can view the status of the build process in the console window, located in the lower right corner of your screen. In the Project Explorer, click on the ">" symbol next to SimpleHtml. Some important items of note:

- Build Targets has two functions: downloading code to your device, and rebuilding the system libraries if needed.
- The src folder contains all your source code files, such as main.cpp.
- The html folder contains all your web based code.
- The overload directory is used to override any system library files. See the example in `\nburn\examples\Overload↩Directory` for more details.

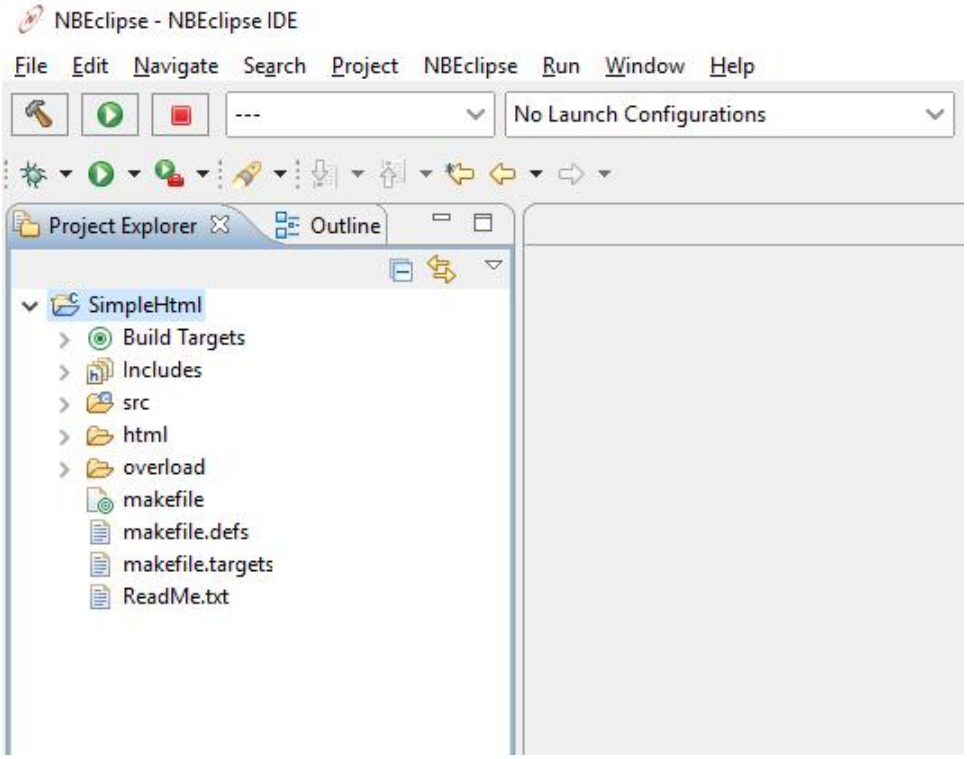


Figure 6.13 Import complete

6.1.4.9 Open your main.cpp source file

Clicking on the ">" symbol next to the src folder, you can open main.cpp in the editor

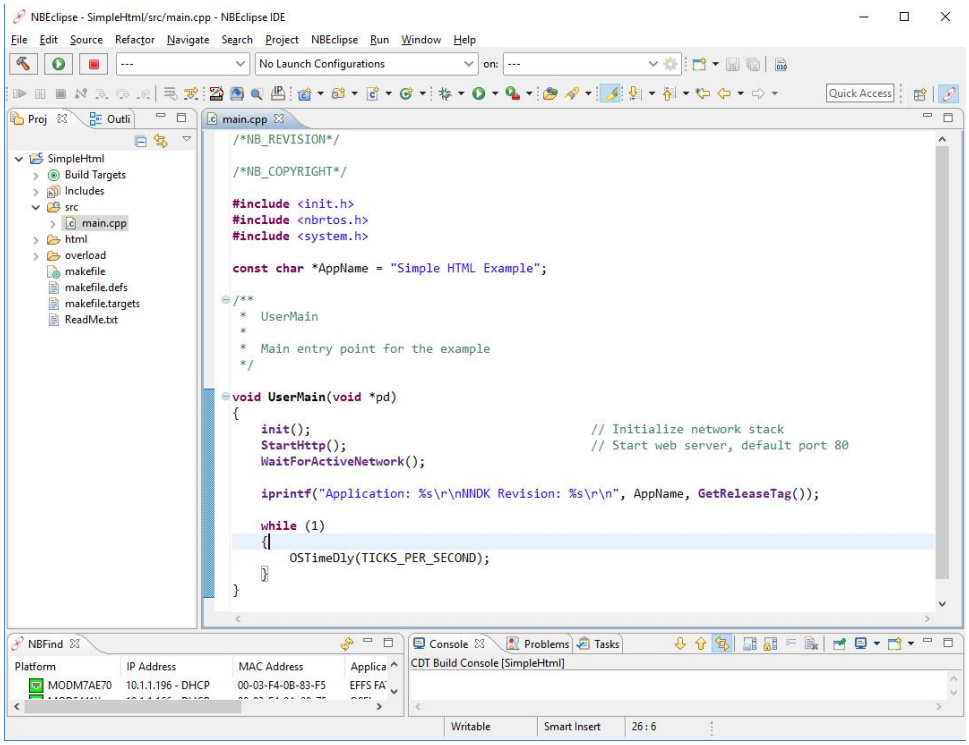


Figure 6.14 Open main.cpp

6.1.4.10 Project Import Complete

At this point you have a project complete with source code files. You can use the editor to modify both src and html files. If you wish to open a html file in text mode to do things like add dynamic content tags such as CPPCALL, VARIABLE, etc, right-click on the html file and use the Open With option to select the text editor:

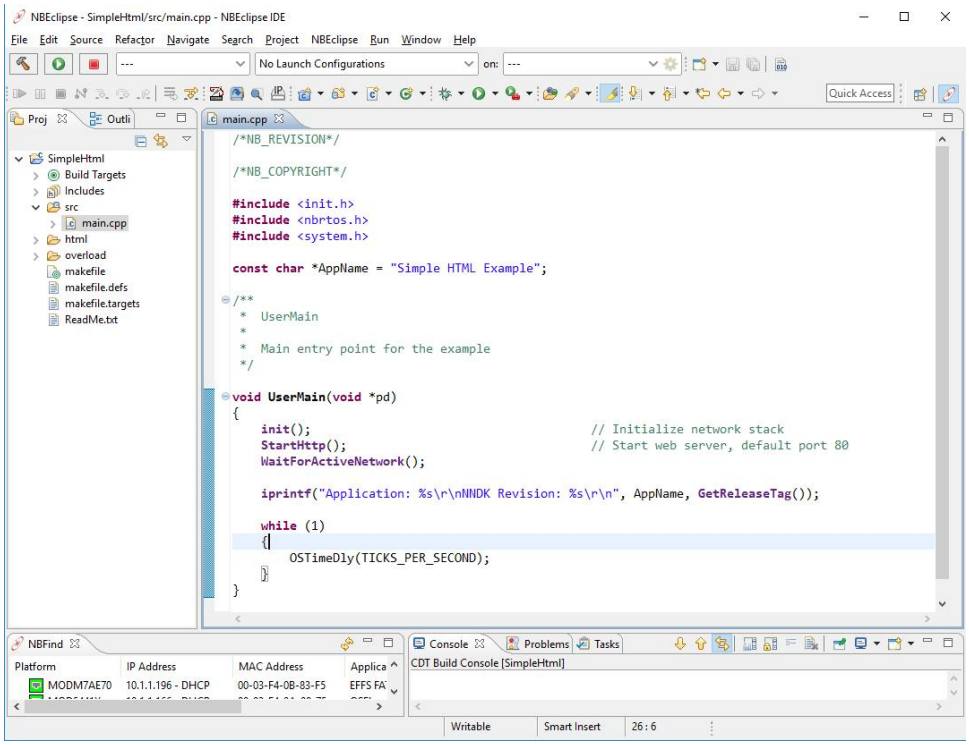


Figure 6.15 Open HTML files in text mode

By default, any time you modify and save a source file, the system will automatically build the application. If you prefer to turn off the auto-build option, to build the project: from the main menu select Project->Build Project, or use the <cntl> b keyboard shortcut.

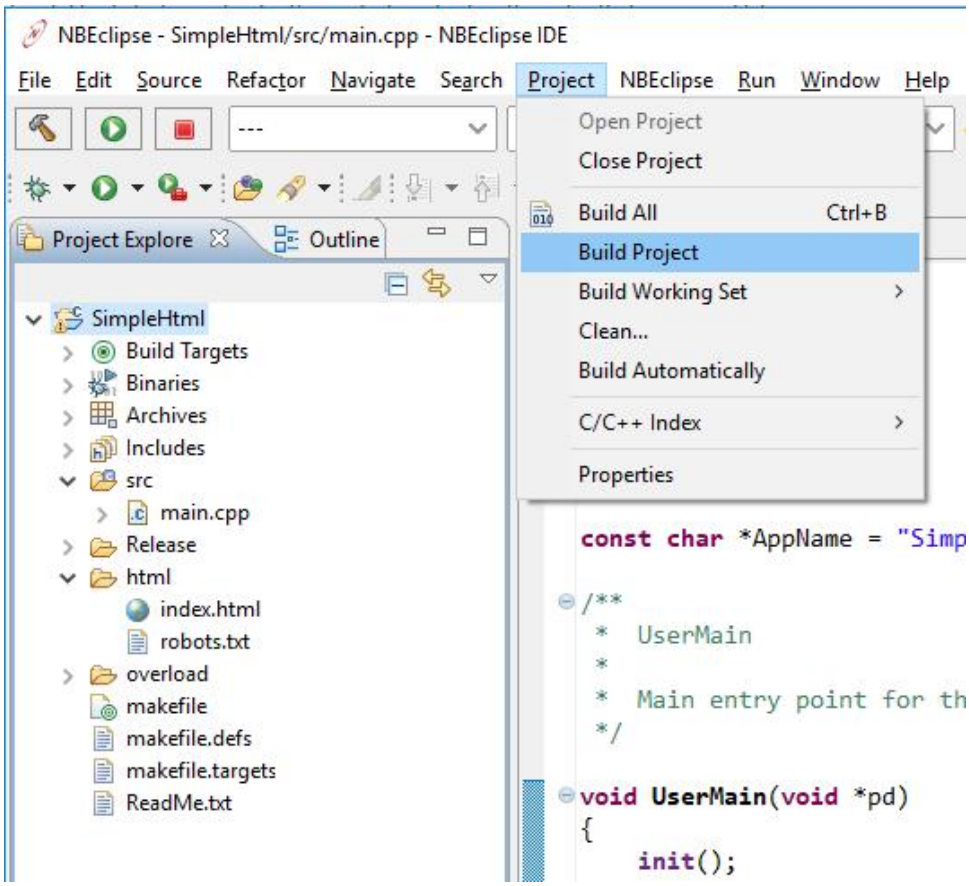


Figure 6.16 Build the Project

6.1.4.11 Download the Application to Your Device

Now that an application image has been built, the next step is to download the binary image to your NetBurner device. NBEclipse can simultaneously handle multiple NetBurner hardware platforms of the same or different types. For example, multiple MODM7AE70 core modules, or a mix with some MOD54415 modules. Targeting a project to a specific hardware device, is accomplished by creating a Run Configuration for release mode project builds, or a Debug Configuration for project debug builds. An important part of such a configuration is associating the device's IP address with the project so NBEclipse knows where to download the project's binary image.

When you create a new project and select the download option for the first time, NBEclipse will create a Run Configuration for you (Run and Debug Configurations can also be created manually). First select your project by left clicking on it in the Project Window:

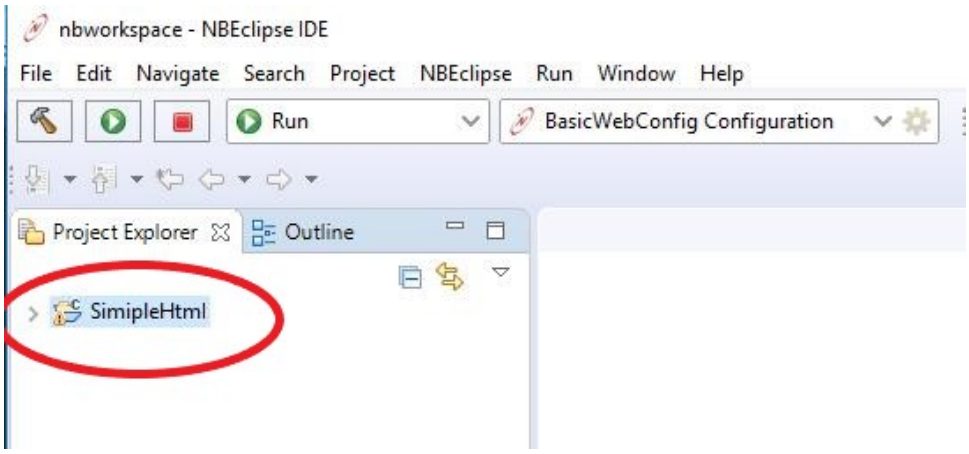


Figure 6.17 Select Project

Then click on the Run button in the upper right corner of NBEclipse:

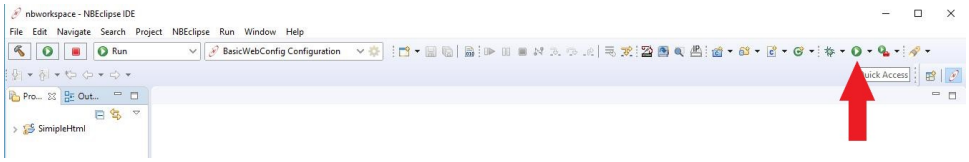


Figure 6.18 Select Run Button

The Run As dialog box will appear. Select NetBurner Application, which will run the application on the remote system: When you click the OK button, NBEclipse will download the binary image to the remote device and reboot it.

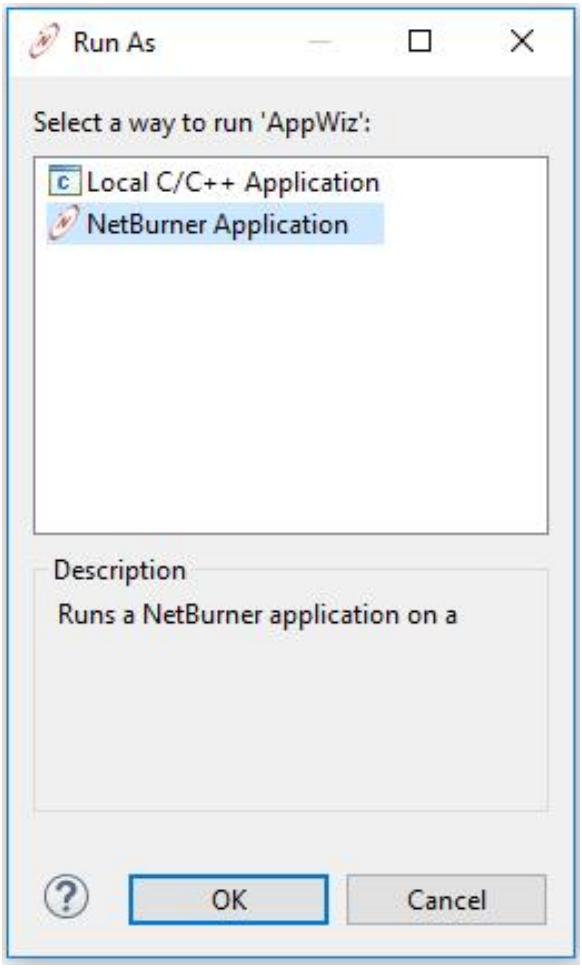


Figure 6.19 Select Run Application

The application should now running on the device. If you do not remember the IP address of the device, you can find it using the "Find" window in the lower left of NBEclipse. In the screen shot below there are 3 devices. The device running SimpleHtml has IP address 10.1.1.196.

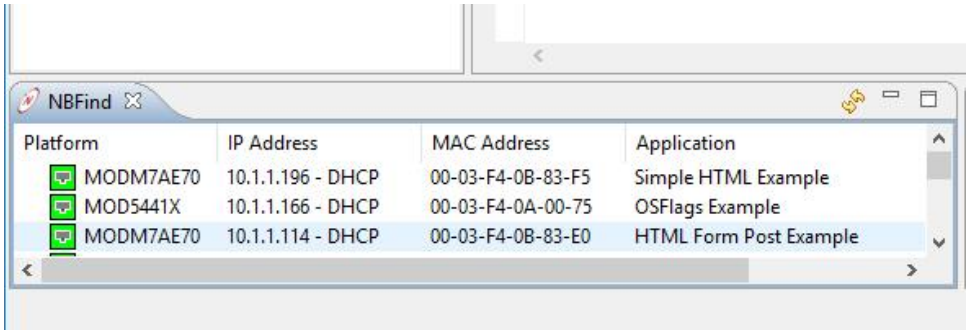


Figure 6.20 Find Device IP

6.1.4.12 View the Device Web Page

Opening a web browser and typing 10.1.1.196 in the URL field:

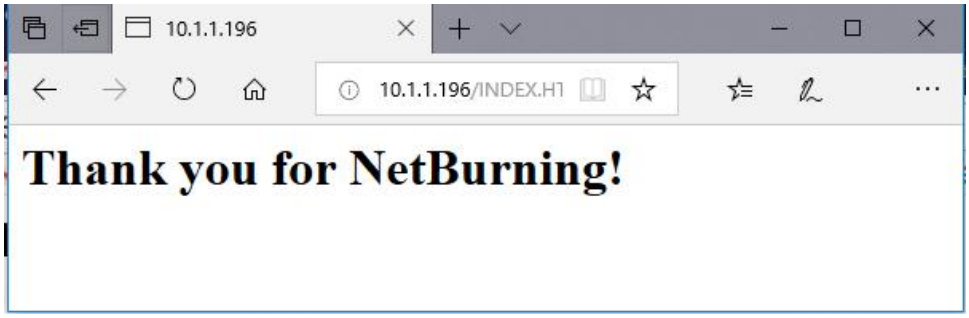


Figure 6.21 HTML Page

If you now look at the upper left side of NBEclipse, you will see the configuration set to Run, and the project set to SimpleHtml for the configuration.

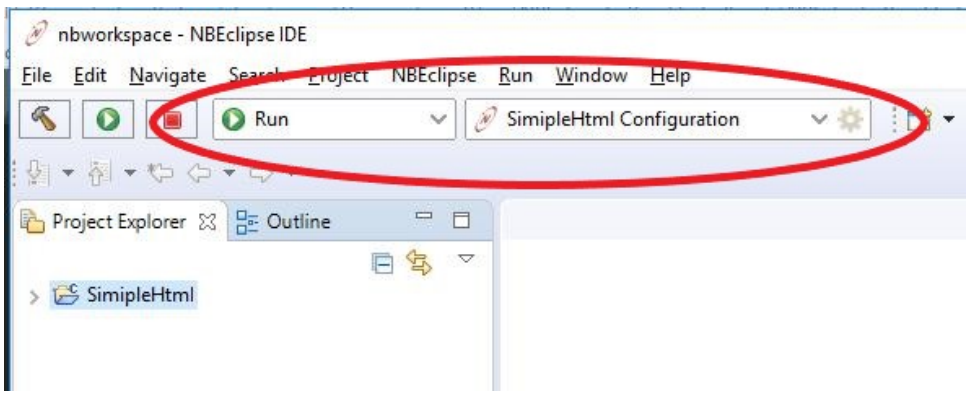


Figure 6.22 Run Configuration

6.1.4.13 View Serial Port Status Messages with MTTTY

All NetBurner examples make use of the debug/console serial port to provide status messages. You will need to have a serial connection to your device either through USB or the DB9 on the development board, depending on your hardware configuration. You can then use the MTTTY serial terminal application to view the serial output. To open MTTTY:

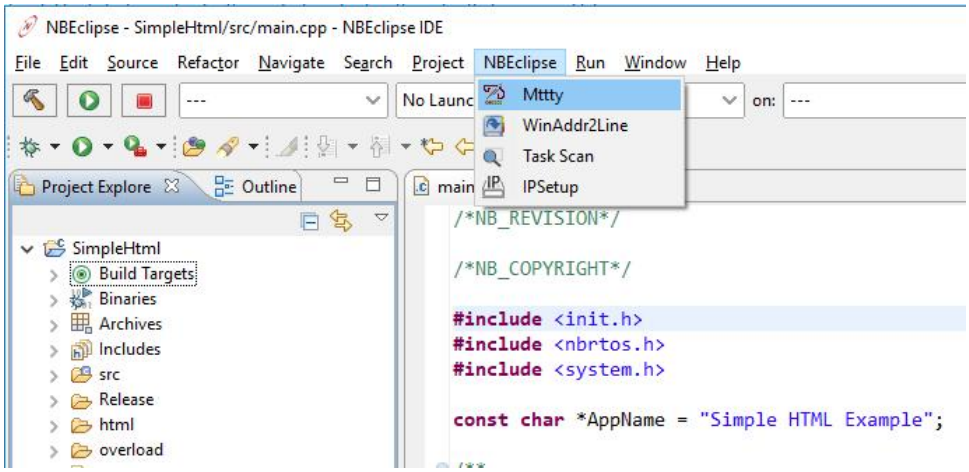


Figure 6.23 Open MTTTY

When the device boots the SimpleHtml application you will see:

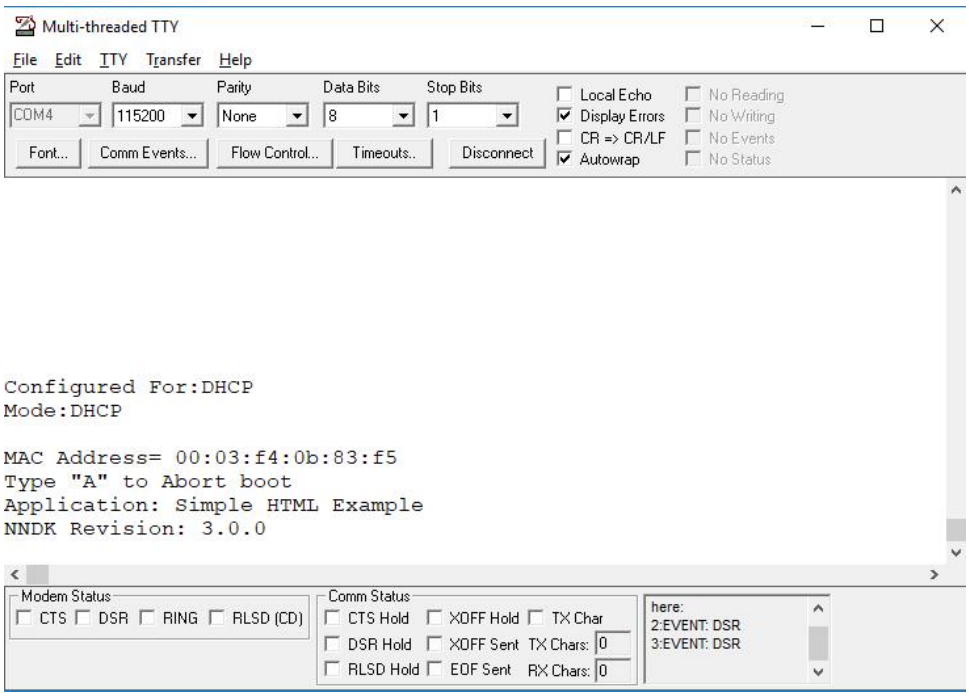


Figure 6.24 MTTTY Output

6.1.5 Debugger

In the previous section we created and ran a project in release mode. To use a debugger, the application must be built in debug mode. All debuggers require the code be built in a different manor than a release build:

- Code optimization is disabled
- If the platform uses external SDRAM with the internal SRAM acceleration for stacks, buffers and variables, it is disabled
  NBEclipse uses a network debugger. This means that an application must be able to at least boot through network initialization so the NBEclipse agent can connect to the GDB stub on the target device.

6.1.5.1 Starting a Debug Session

We will pick up right where we left off for the SimpleHtml example project in the previous section. First, lets open main.cpp and add a variable. UserMain() is shown below:

```
void UserMain(void *pd)
{
    init();                                     // Initialize network stack
    StartHttp();                                // Start web server, default port 80
    WaitForActiveNetwork();

    iprintf("Application: %s\r\nNNDK Revision: %s\r\n", AppName, GetReleaseTag());

    while (1)
    {
        OSTimeDly(TICKS_PER_SECOND);
    }
}
```

Add two lines of code to create an integer and increment it:

```
int i = 0; and i++;
```

So UserMain() becomes:

```
void UserMain(void *pd)
{
    init();                                     // Initialize network stack
    StartHttp();                                // Start web server, default port 80
    WaitForActiveNetwork();

    iprintf("Application: %s\r\nNNDK Revision: %s\r\n", AppName, GetReleaseTag());

    int i = 0;      // <--- Declare variable
    while (1)
    {
        OSTimeDly(TICKS_PER_SECOND);
        i++;        // <--- Increment variable
    }
}
```

Previously for the Run Configuration and download we clicked on the green play button in the upper right corner of NBEclipse. This time we will click on the debug button, which is next to it on the left:

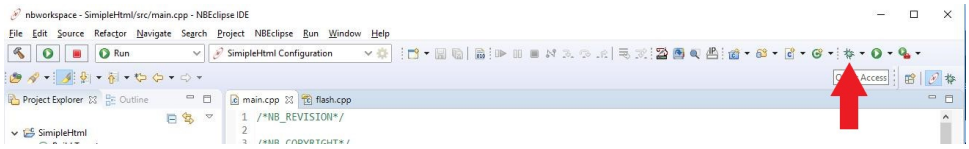


Figure 6.25 Select Debug Configuration

It will take a few seconds for the debugger to connect, and the NBEclipse Perspective will change from the NetBurner perspective to the Debug perspective:

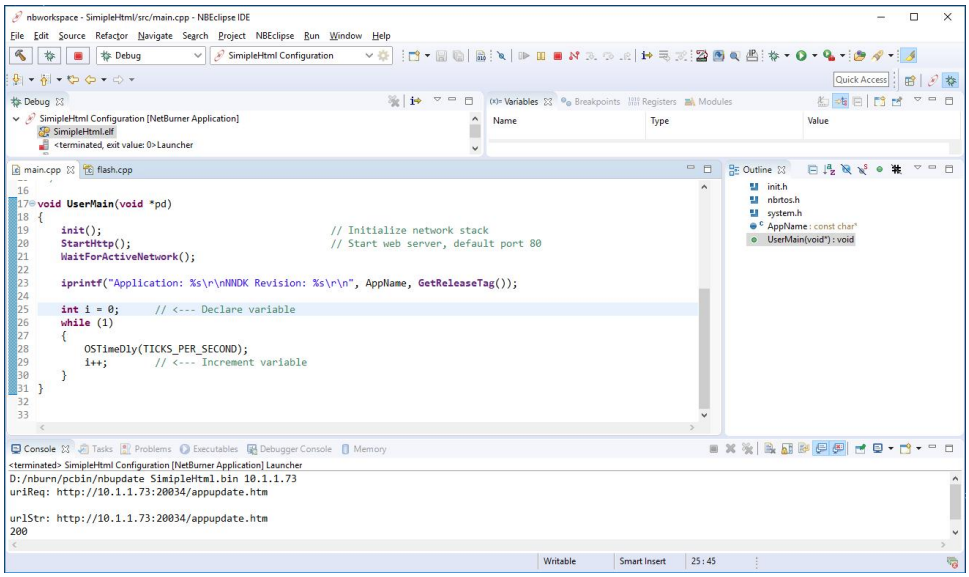


Figure 6.26 Debug Perspective

At this point the code is running. To set a breakpoint on our new variable, double click to the left of the line of code. In this example it is just to the left of line 29:

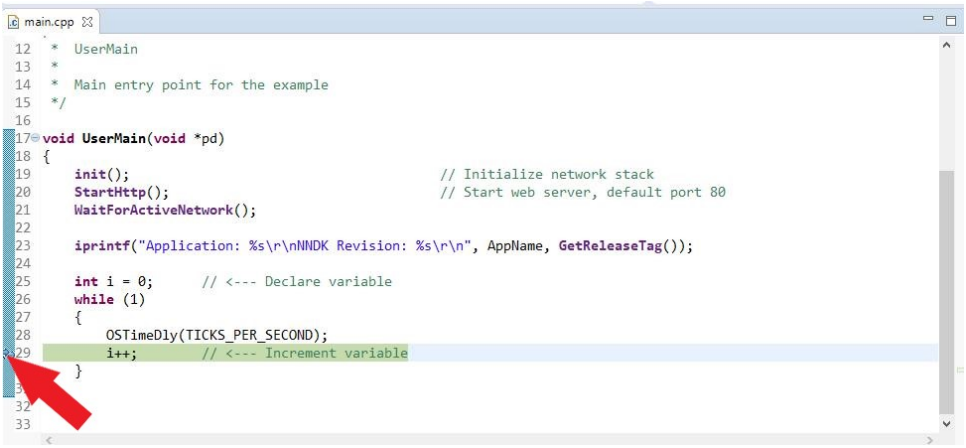


Figure 6.27 Set Breakpoint

The screen shot below shows the program stopped at line 29. The Variables window in the upper right shows the variable 'i', which has incremented 7 times so far.

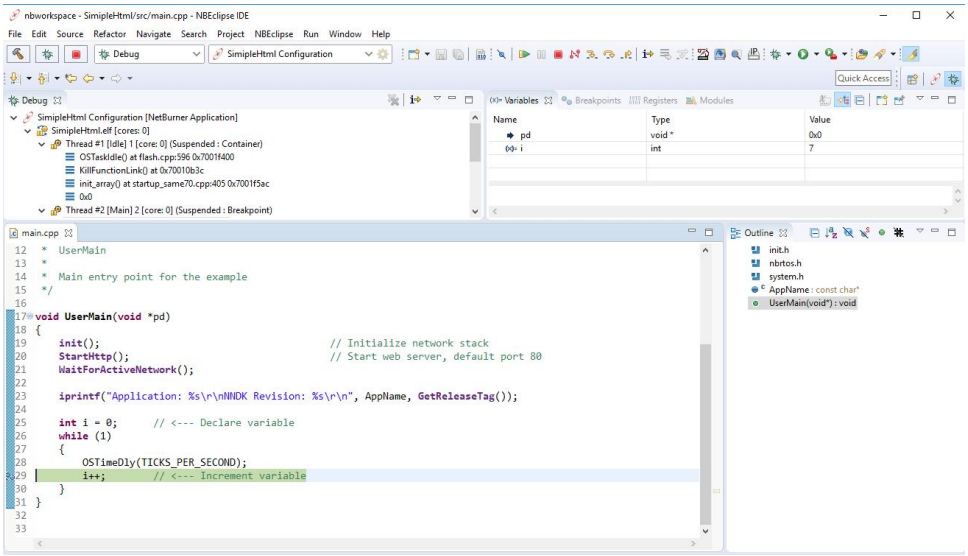


Figure 6.28 Stopped at Breakpoint

The debugger operations are controlled with the tool bar at the top of the screen. If you hover the cursor over the icon the tool tip will describe what it does. Common features are:

- Resume: Available when stopped at a breakpoint. Selecting it will resume the application and stop at the next breakpoint.
- Terminate: Disconnects the debugger from the target device. The device will resume execution without any breakpoints.

- Disconnect: Identical to Terminate.
- Step Into: Single step into a function call.
- Step Over: Single step to the next line of code in the current function.
- Step Return: Execute until current function returns.
  Notes on debugging:
- You can switch between the Debug perspective and NetBurner perspective using the NetBurner and Debug icons in the upper right corner of the screen.
- You can edit and build your application from either perspective.
- Hovering the cursor over variables, macros and functions in the source code window will display information on that item.
- The upper left window displays the RTOS task stack.
- It is recommended to terminate a debug session before starting a new one.
- The three icons at the top left of the screen: build, debug and stop, are not used by either perspective.

6.1.6 Rebuilding Projects & Libraries

6.1.6.1 Rebuild Project Source Files

A project "clean" will rebuild all of the project's source code files. It does not affect the library system files associated with the project. From the Project Explorer pane, right-click on your project, then select Clean Project:

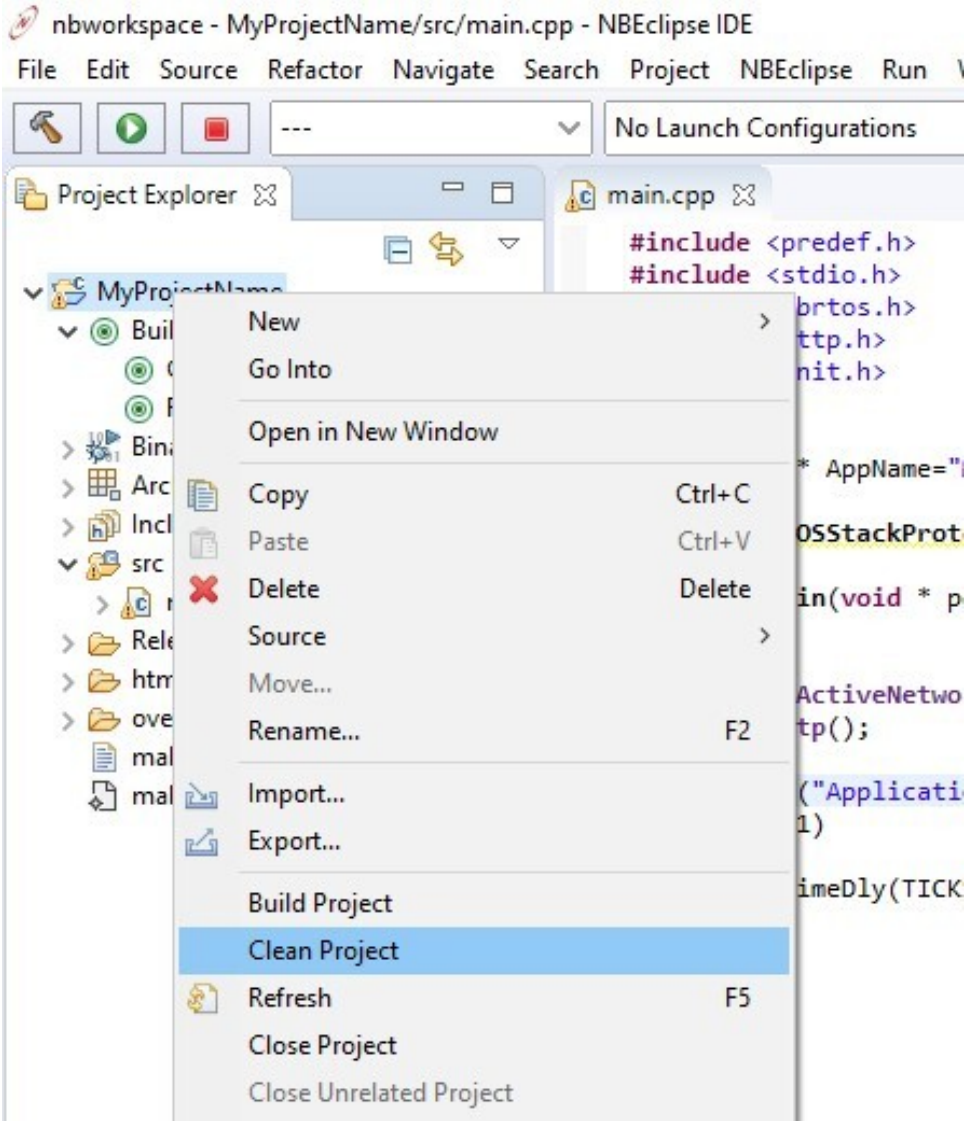


Figure 6.29 Rebuild project files

6.1.6.2 Rebuild Library Files

This is normally not necessary, but there are cases in which the user may want to rebuild the NetBurner system library files. For example, if the user made changes to system source files or added a system source file to a project's overload directory. In these cases, it's recommended to rebuild the NetBurner system library files followed by "cleaning" the project's source files.

To force a rebuild of the NetBurner library files associated with a project, double-click on Clean Project Library in the Build Targets tree:

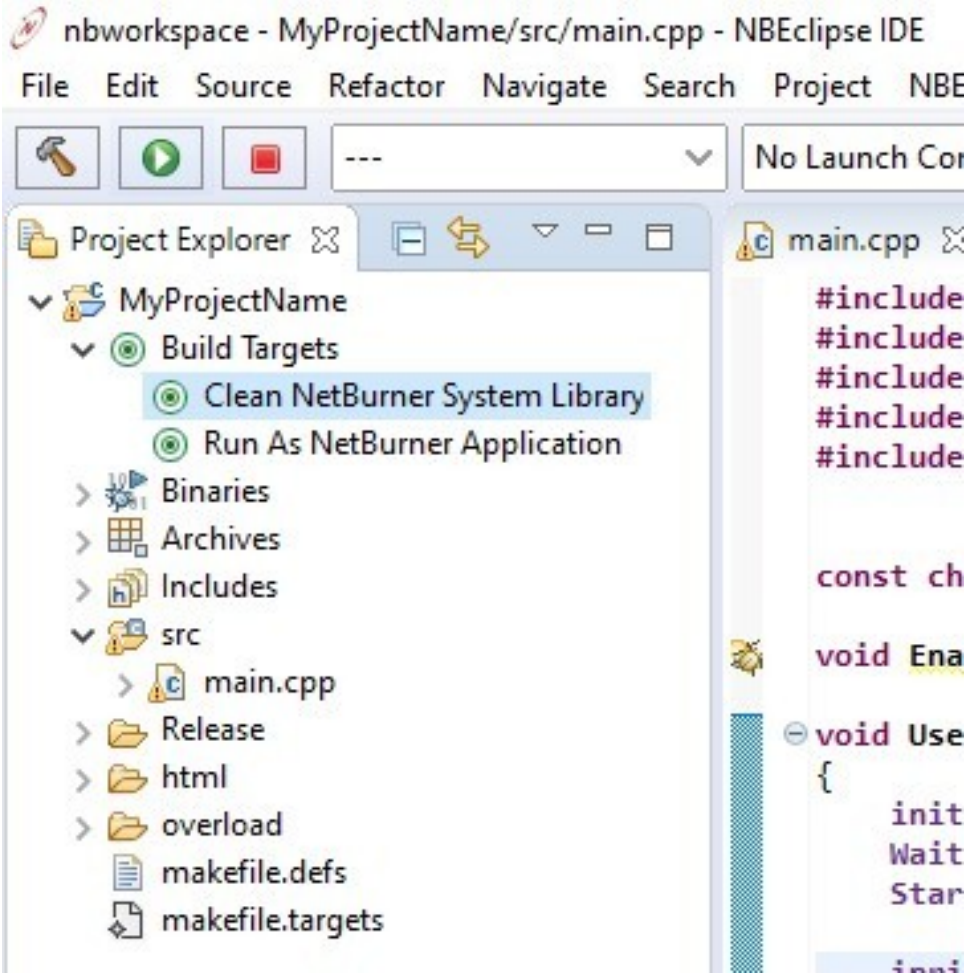


Figure 6.30 Rebuild a project library

6.1.7 Overload Directory

The overload directory, located inside a project, is used to override any system file by including local copies of system source or header files. This allows the user to isolate system files changes to a specific project. The overload directory provides an alternative to modifying the original system files, which would affect all projects.

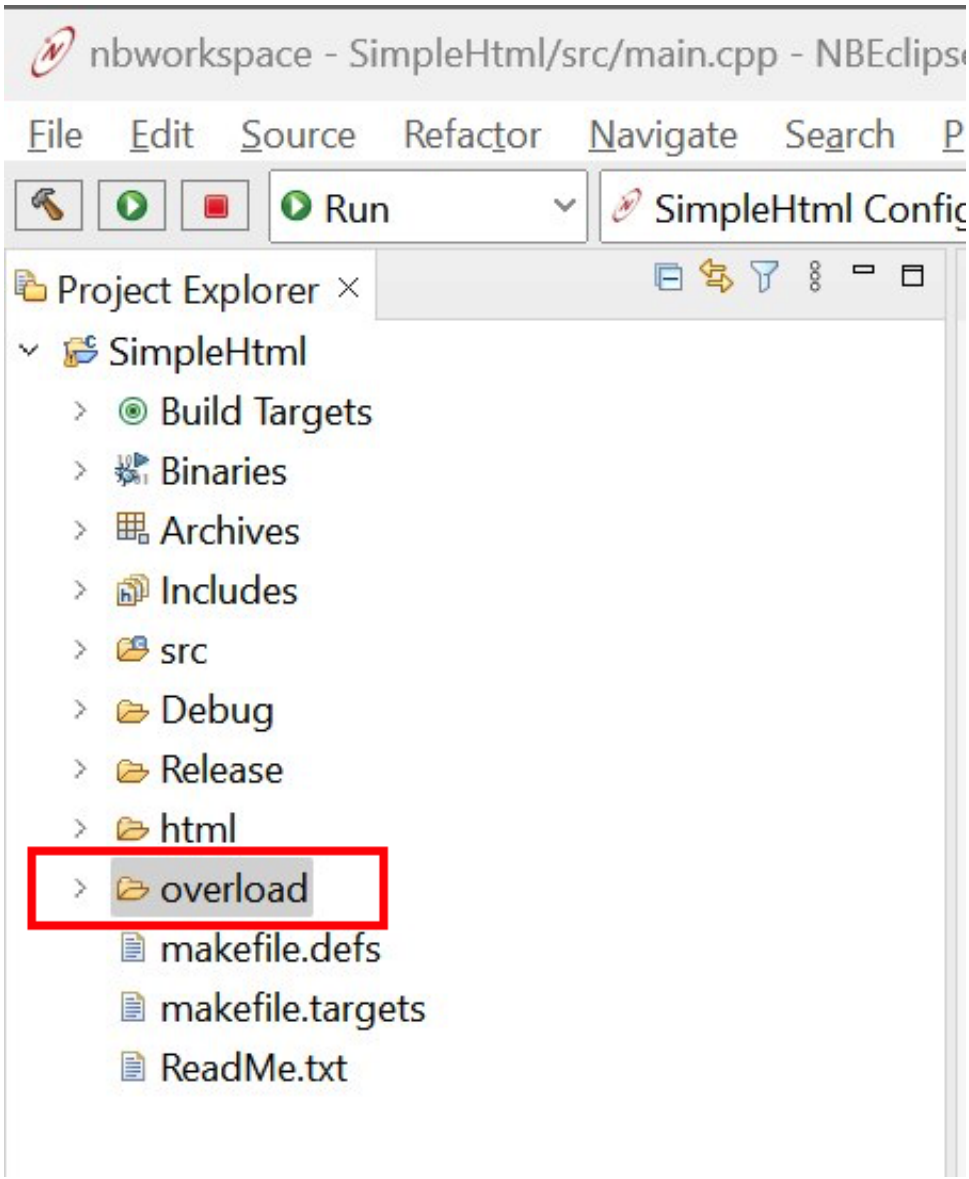


Figure 6.31 Project's Overload Directory

To use this feature, the file to be overloaded should be placed in the overload directory under a directory structure that matches the file's location relative to the NNDK install directory.

To demonstrate the usage and configuration of the overload directory, we will show an example. Let's say the user wants to overload the system's timezone structure so that they can change the name of a timezone. Perhaps the user is deploying their device in a non-English speaking country and would like to change the language used for timezone names. The timezone names are defined in a system file located in the following directory:

```
/nburn/nbrtos/source/timezones.cpp
```

Therefore, the directory structure is replicated inside the overload directory with the following structure:

```
<project root>/overload/nbrtos/source/timezones.cpp
```

The directory structure can be created by right-clicking the overload directory and selecting New->Folder. The overload directory must have a matching directory structure to the path of the system file to be overloaded. For this example, the overload directory should look like the following:

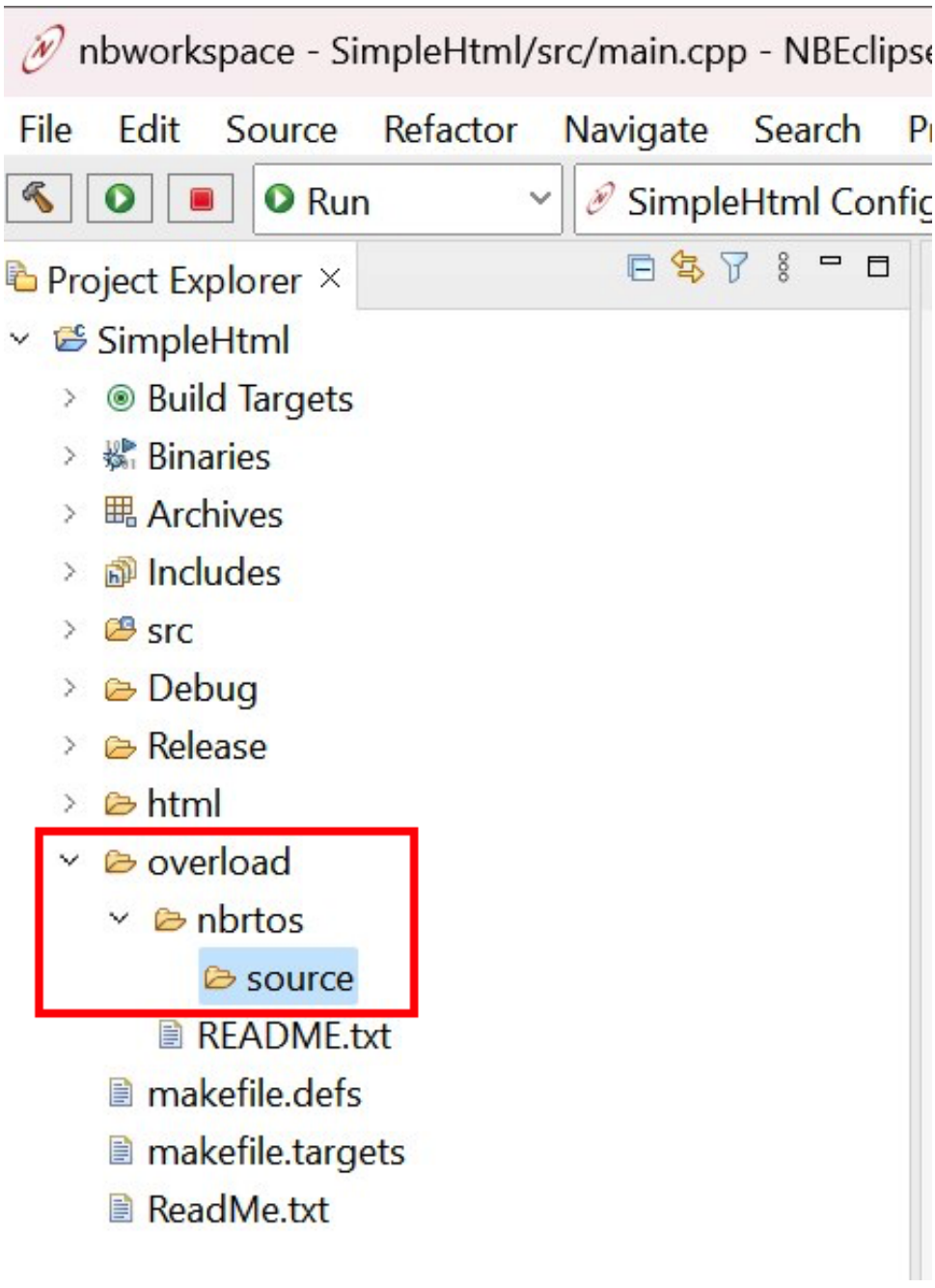


Figure 6.32 Project's Overload Directory Structure

Next, the modified source file must be placed in the correct path in the overload directory. The file can be drag and dropped in place. Alternatively, the file can be imported into the overload directory by right-clicking the overload directory and navigating to Import->General->File System and navigating to the source file. After adding timezone.cpp, our overload example would look like the following:

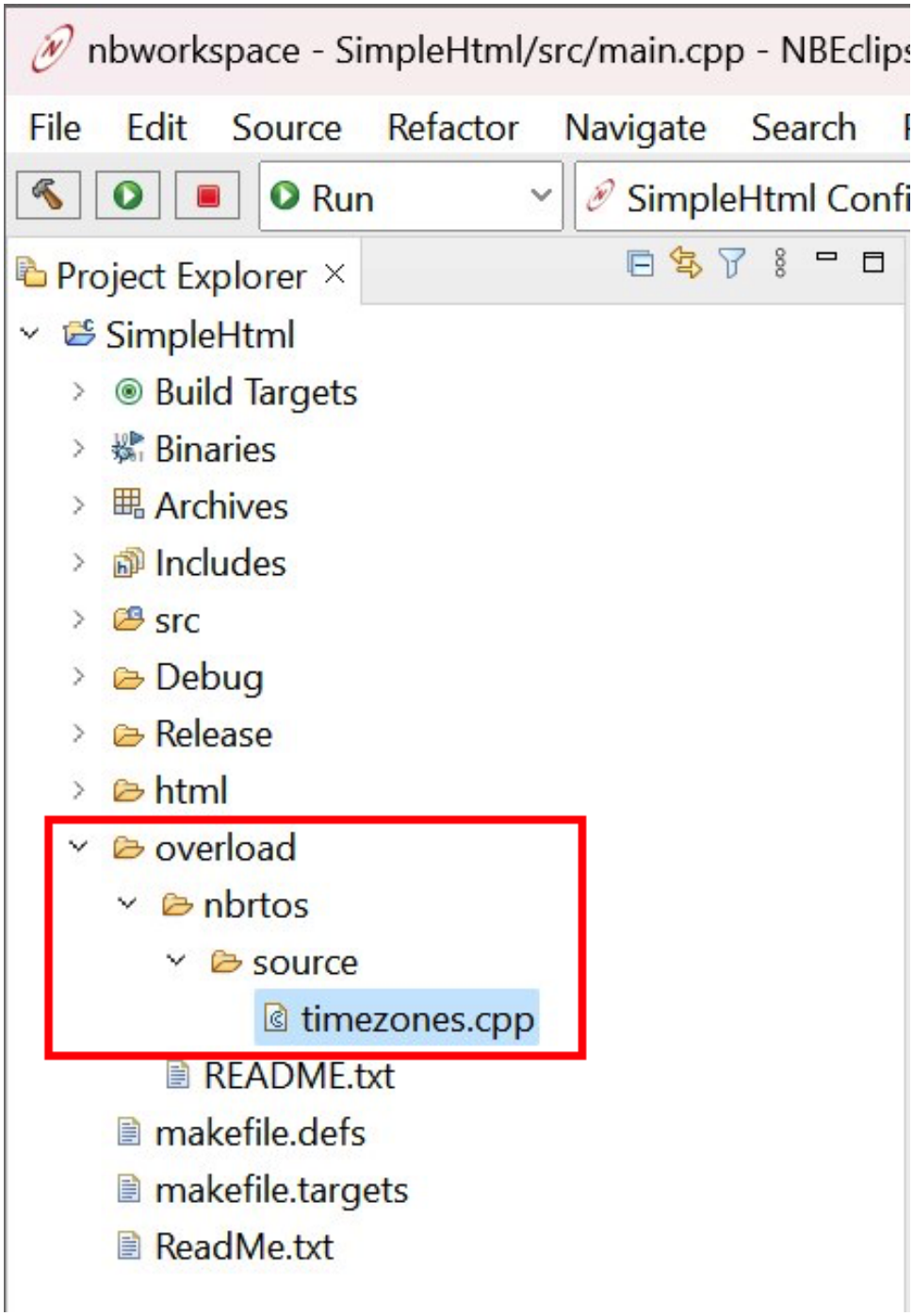


Figure 6.33 Overload Source File Added

Next, the project properties must be configured so that the C++ compiler knows to include the path to the overloaded source file. This can be done by right-clicking the project and selecting "Properties" as shown in the image below.

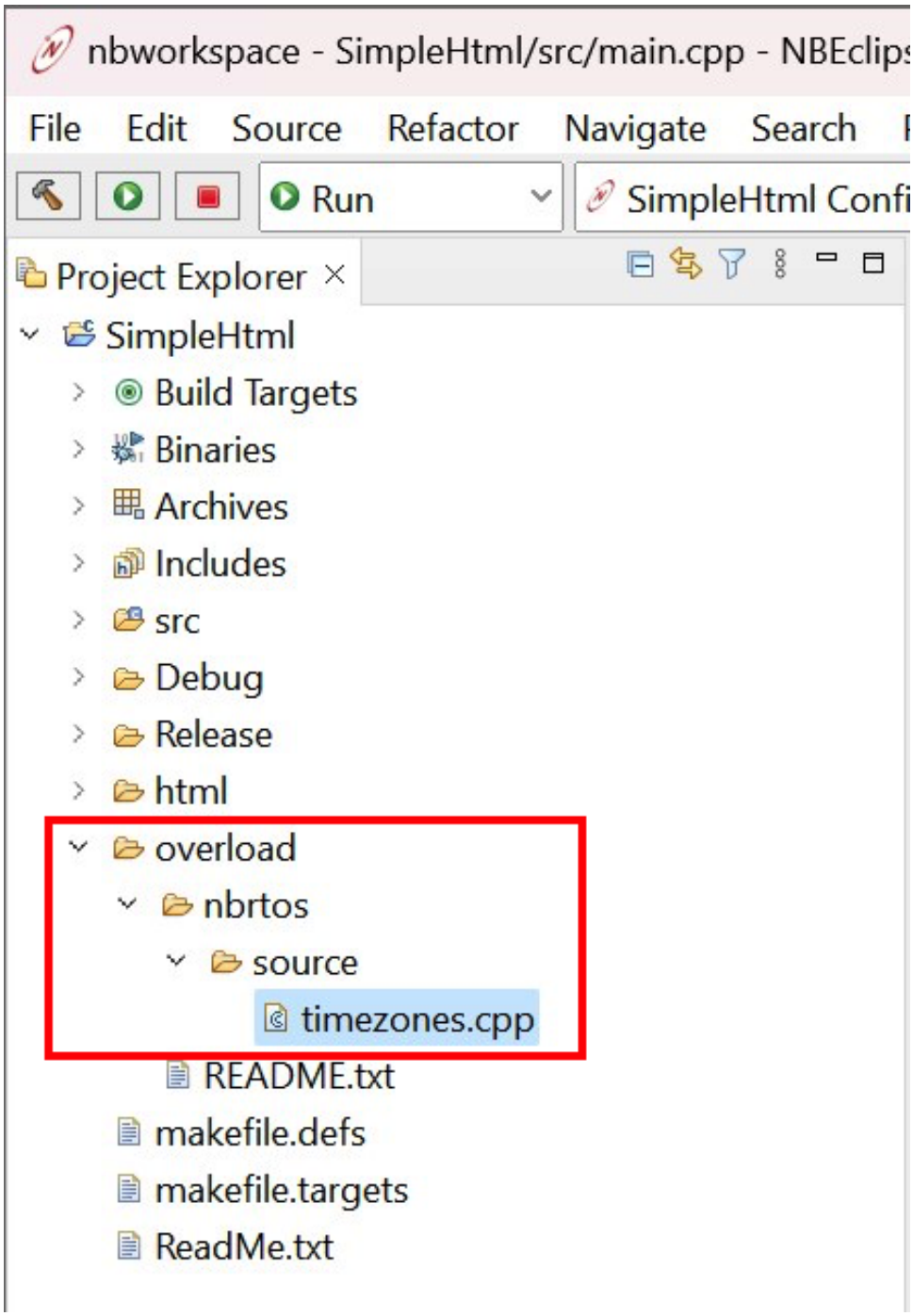


Figure 6.34 Project Properties

Navigate to C/C++ Build->Settings, select GNU C++ Compiler->Includes.

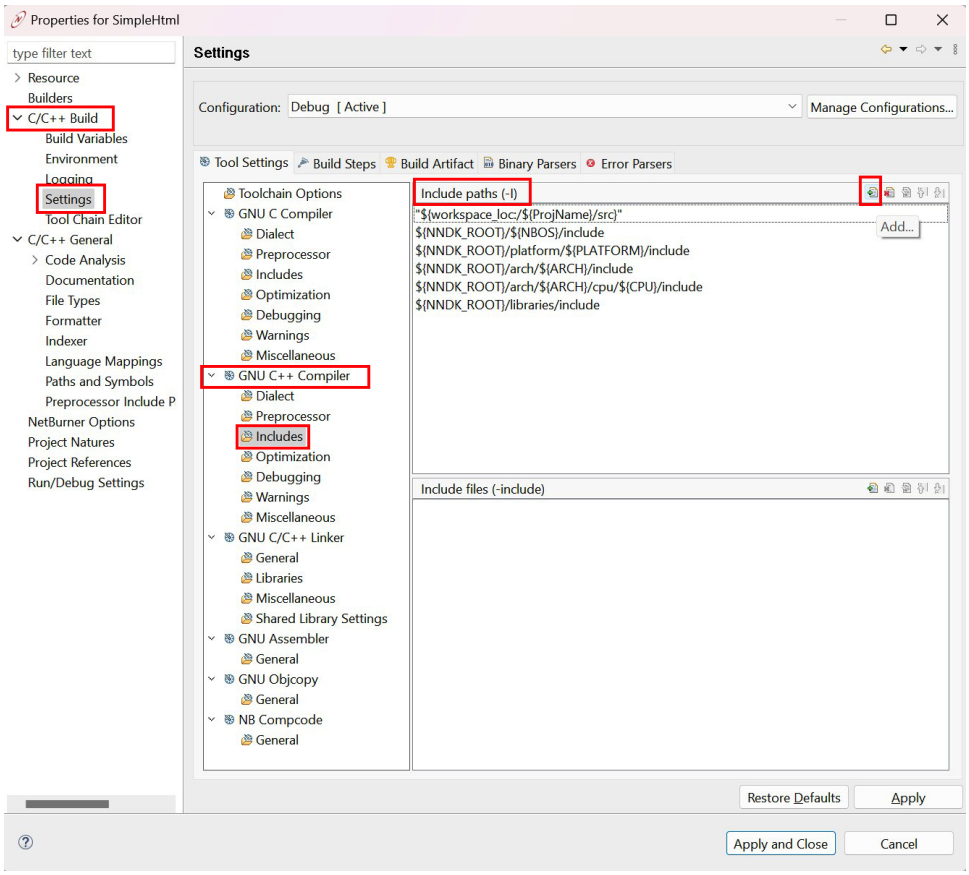


Figure 6.35 Overload Include Paths

Add the overload directory path to the "Include Paths" list by selecting the "+" logo. If utilizing C code, then the path to the overload directory should be added to GNU C Compiler->Includes as well.

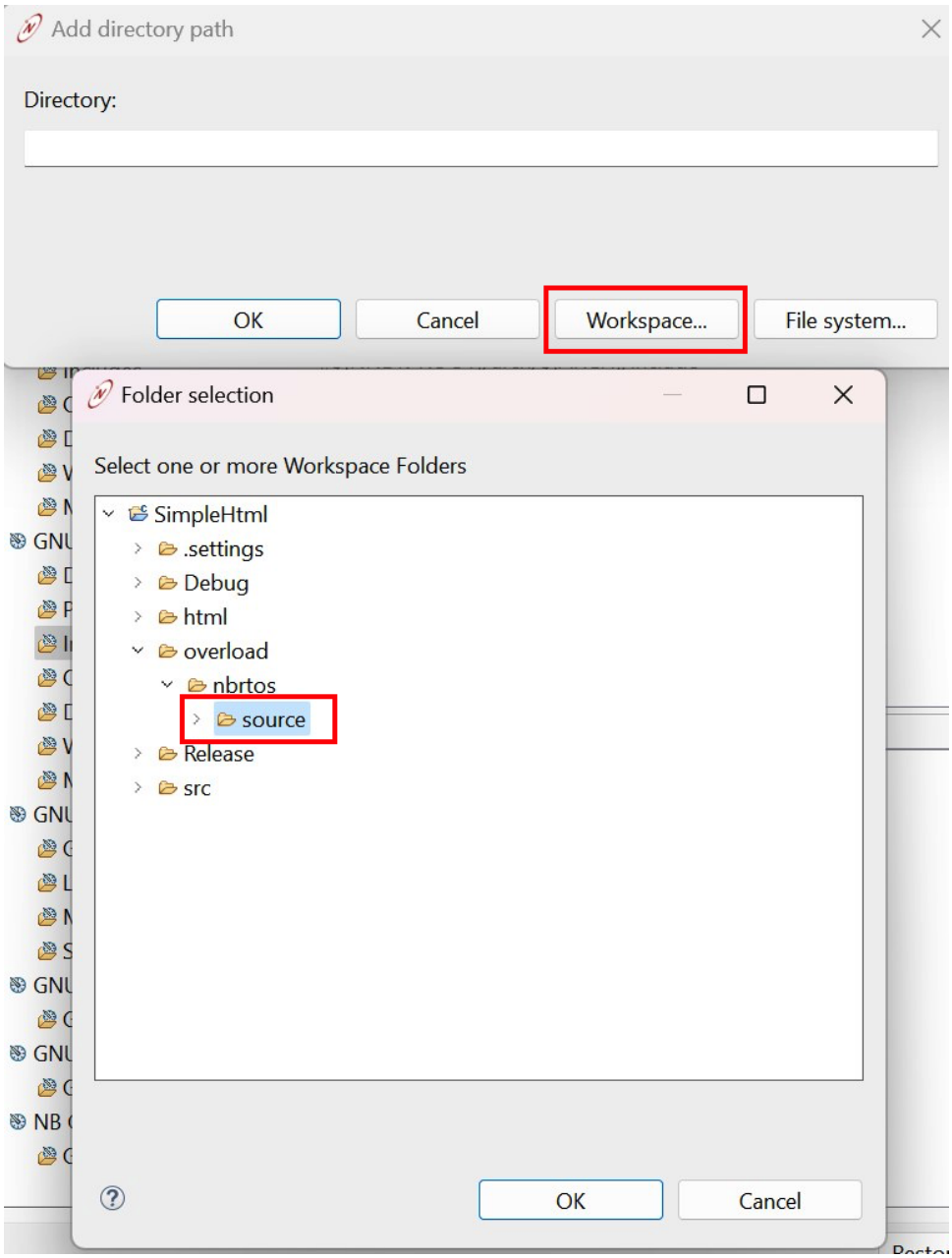


Figure 6.36 Overload Include Path Selection

After adding the overload directory path to the list of "Include Paths," the list should look like the following↩ :

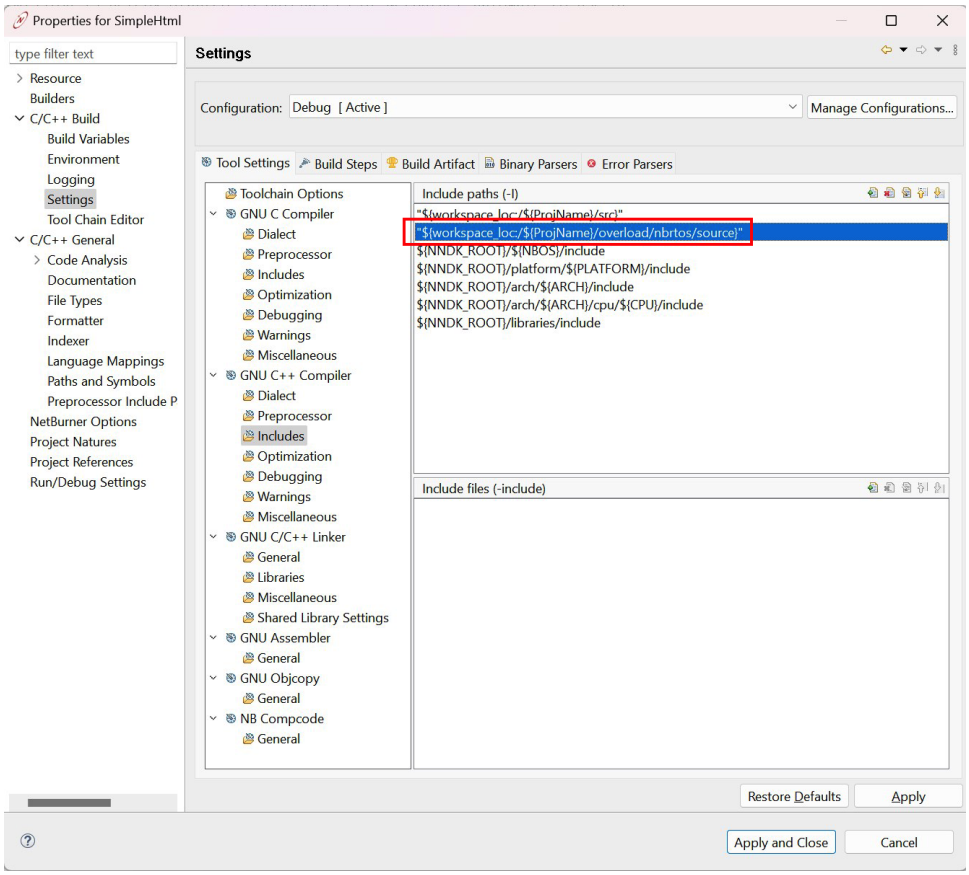


Figure 6.37 Overload Directory Path Added to Include Paths List

Select "Apply and Close" to apply the modifications.

Last, we need to rebuild the NetBurner system files then clean the project files (the order of operations matters). For instructions on doing so, see Rebuilding Projects & Libraries. From this point on, modifying the overloaded source file, timezone.cpp, will automatically re-build the necessary files in the NetBurner system files when the project builds.

6.2 Add a Library to a Project

NBEclipse includes the most common libraries by default. However, there are times you will need to tell the build system to include additional libraries, such as if your application uses the Embedded Flash File System (EFFS) for flash memory cards, in which case you need to add the FatFile library.

Note: If you are building a project that uses EFFS functions, and have not added the FatFile library, you will see many link errors for functions that start with f_ and fm_.

To add a library:

- In NBEclipse, right-click on your project, and select "Properties".
- Select "C/C++ Builds -> Settings" on the left-hand side.
- Select "GNU C/C++ Linker -> Libraries" under the "Tool Settings".

- In the "Libraries" list box, add "FatFile" by using the add icon in top-right corner of the list.
- In the "Libraries Search Path" list box, add the path to the library. For example, for MOD5441x device, the path is: `\nburn\platform\MOD5441X\original\lib`. Note: the naming convention for gcc is as follows: the name of the library is actually libFatFile.a, but in the Libraries entry box you only use what comes after "lib" and do not use the .a suffix. In this case, we just use "FatFile".

Note

The other common library to add is the EFFS-STD library: StdfFile. EFFS-FAT is a FAT32 library for flash memory cards. EFFS-STD is used to create a file system in the flash memory chip on the device.

6.3 Change The IP Targeted By A Project

If the IP of your device changes (for example, if you unplug it and it is assigned a different IP when you plug it back in), you will need to tell the build system to target a different IP.

To change the targeted IP:

- In NBEclipse, click on "Project" from the top menu, then click "Properties".
- In the left-hand panel of the Properties menu, click "NetBurner Options".
- Change the targeted IP address in the right-hand side of the panel.

6.4 Change CompCode Settings for EFFS

The CompCode utility is used in the final stage of a build to compress an application image. The COMPCODE memory address range defines the amount of space available as application space so programming an application in flash memory does not overwrite any of the space allocated for the EFFS-STD file system. Documentation on the memory map for your particular device can be found in the Platform Reference section of this document.

To change the CompCode memory address range for the application in NBEclipse:

1. Right-click on your project, and select "Properties".

2. Select "C/C++ Builds $>$ Settings" on the left-hand side.

3. Select "NB Compcode". In the All Options section the current value of the application memory range is displayed. In this example it is a NANO54415 platform. To make the next step easier, highlight and copy the memory address range.

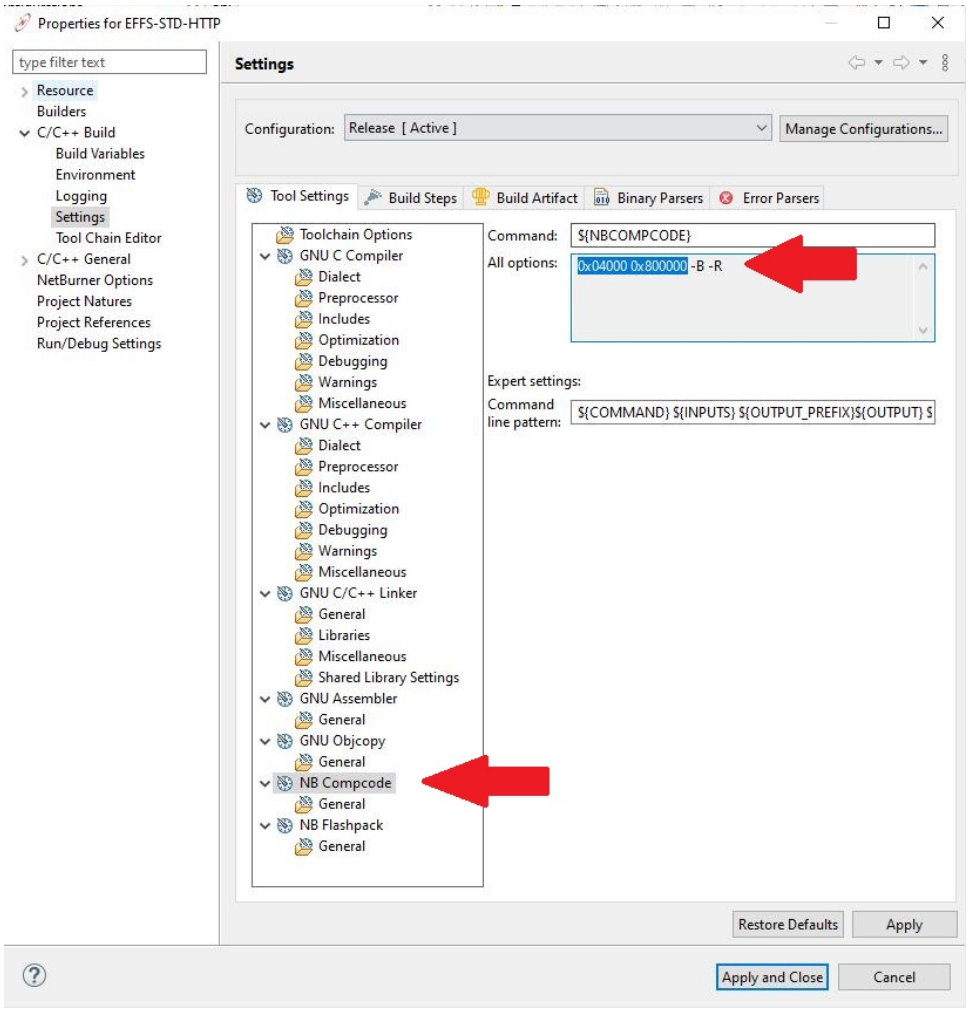


Figure 6.38 Compcode default values

4. Select the NB Compcode > General tab. If the current value is the factory default, you will see the macro ${COMPCODEFLAGS} displayed. This is the value that is read from the factory default file.

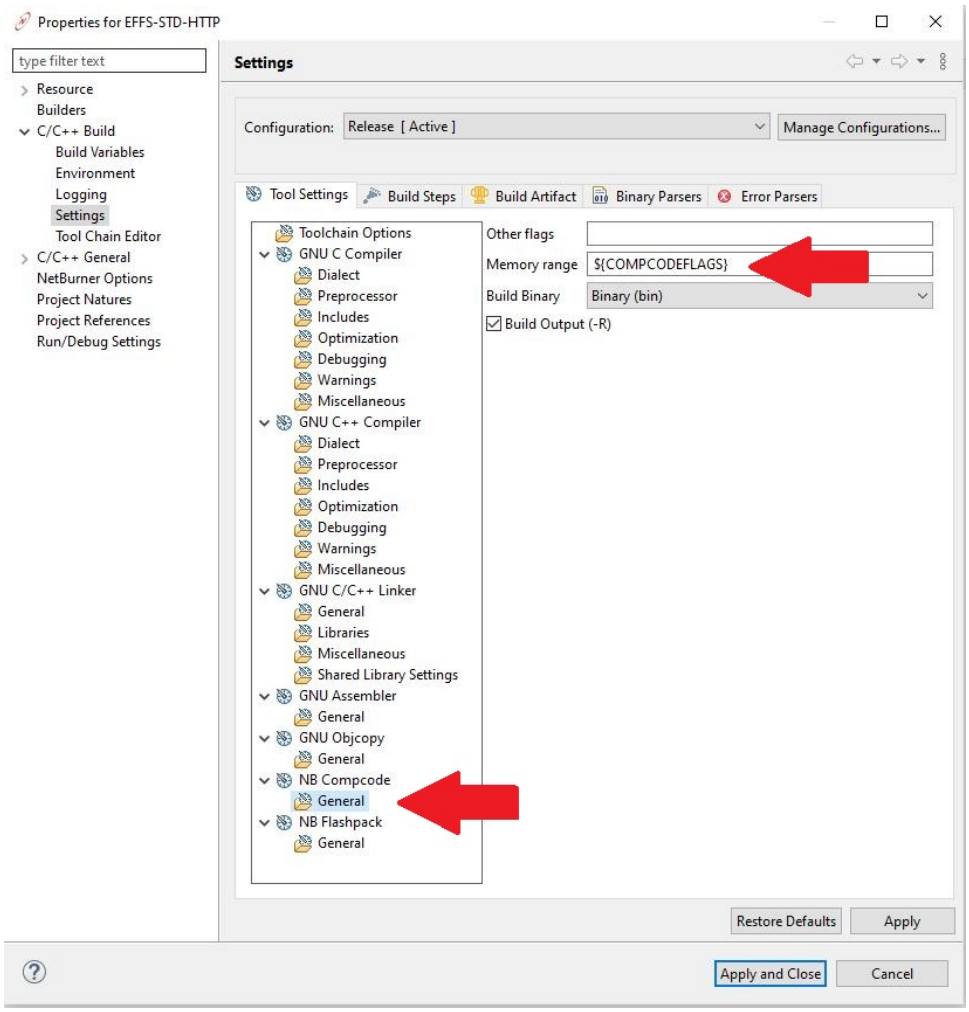


Figure 6.39 Compcode default values

5. Delete ${COMPCODEFLAGS} and paste the memory range you copied in the previous step, then edit it to specify the desired memory range. In this example we will change 0x800000 to 0x700000. So you end up with: 0x04000 0x700000. This would be used to allocate 0x100000 bytes to the EFFS-STD file system space. When you are finished editing, click on Apply or Apply and Close.

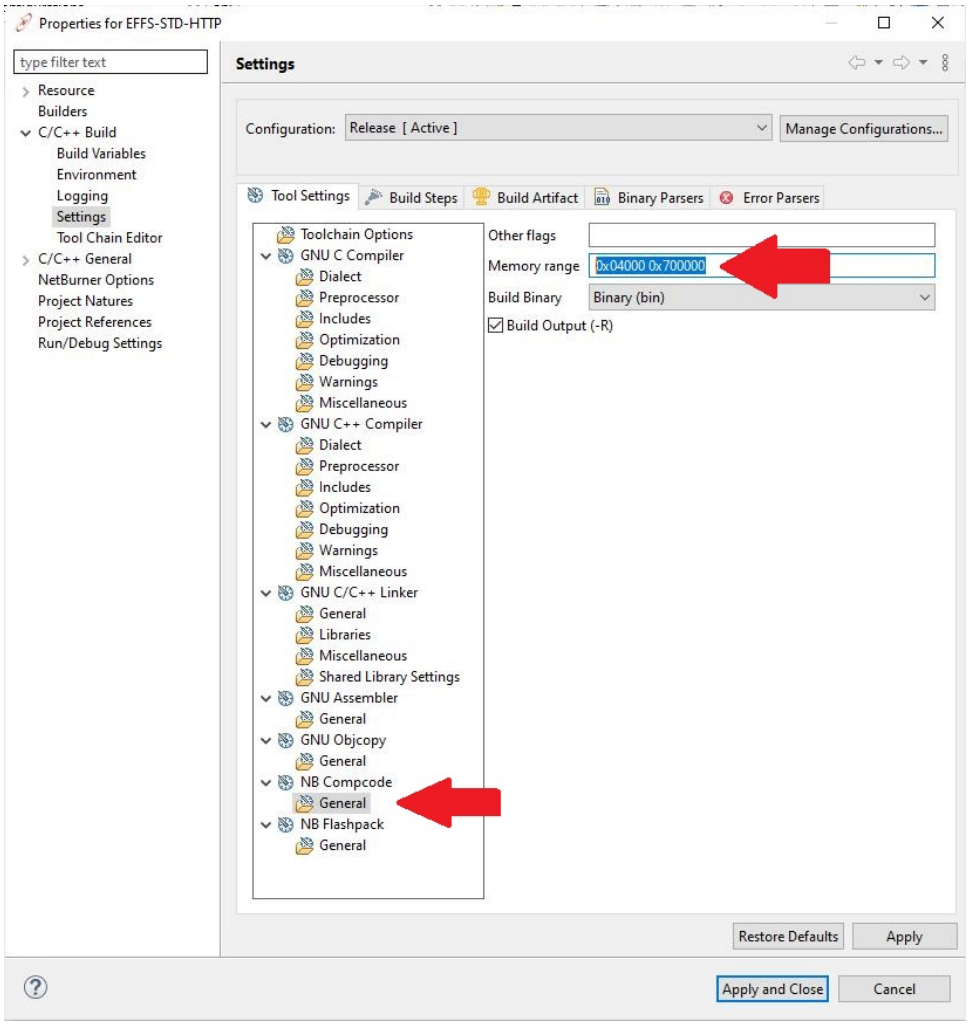


Figure 6.40 Compcode default values

Chapter 7

PC Tools and Utilities

This guide describes the PC tools and utilities that run as stand-alone programs, as well as automatically as part of the NetBurner development tools suite. The executable version of the tools are located in `\nburn\pcbin`, and the source code is located in `\nburn\pctools`.

7.1 Multi-threaded TTY Serial Terminal (MTTTY)

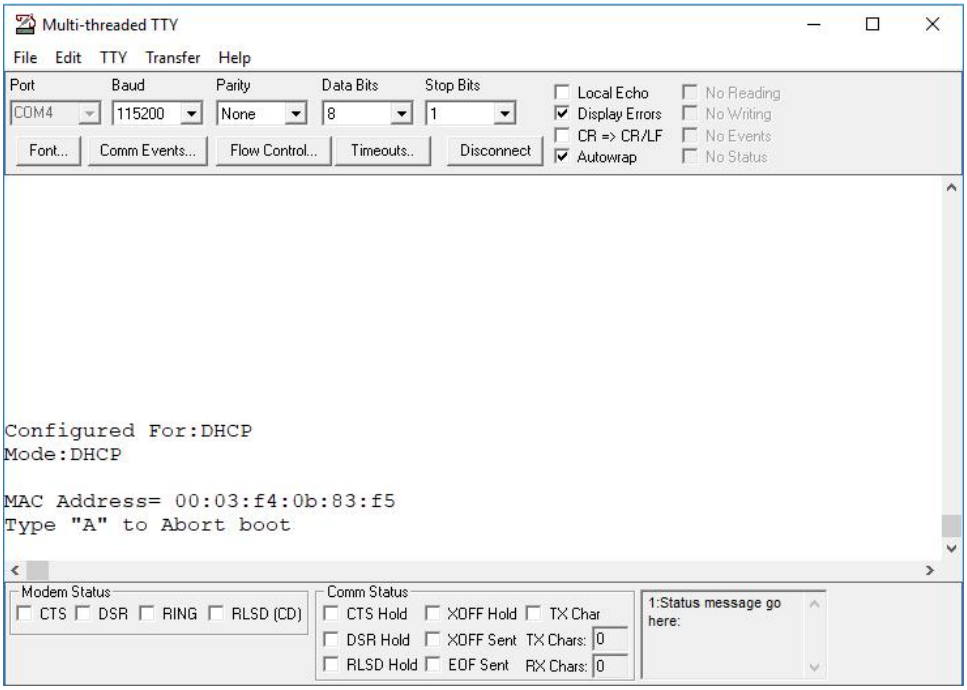
Multi-Threaded TTY (MTTTY) is a serial terminal utility that you can use to communicate with the RS-232 serial ports on your NetBurner device or development kit. It is multi-threaded in that you can run multiple instances of MTTTY for each serial port.

MTTTY is commonly used for:

- Displaying status messages sent from the NetBurner device, such as when using iprintf()
- As a serial interface to any application

MTTTY can be started from:

- Inside NBEclipse by clicking on the MTTTY icon
- From the Windows NetBurner program group
- From a command line. The path is: `\nburn\pcbin\mttty.exe`



7.1.1 MTTTY FAQ

- The default serial port used for stdio for NetBurner devices is UART0.
- The DB9 connector pinout on NetBurner development boards will have the RS-232 rx and tx signals reversed, so that a null-modem adapter is not required and you can use a straight-through serial cable.
- The DB9 connector pinout on NetBurner Serial-To-Ethernet devices, such as the SB700EX and SB800EX, will be identical to a standard PC type pinout, so a null-modem adapter will be required. (The serial cable that comes with the NetBurner development kits for these specific devices is a null-modem cable)

7.2 Wget

Primary method to program an application into a NetBurner device by posting the application file to the device's config server. It is used automatically by both NBEclipse and the Makefile command line tools. The default flags to call wget are as follows: `wget -nv -O- --progress=dot:micro --post-file=<file name> <Device IP>:20034/appupdate.htm`

Wget also supports loading code onto a device that has been protected by username/password authentication. On the command line, the device username and password are configured by setting the environment flags NBUSER and NBPASSWORD. For example, in a make load, set the username and password as follows: `make load NBUSER=foo NBPASSWORD=bar`. In NBEclipse, username and password are set via the project options. Right click on a project and select `Properties`. Username and password text input boxes can be found under `NetBurner Options`. Finally, if manually utilizing wget, use the following wget flags to set a username and password: `--user=foo --password=bar`.

7.3 NBUpdate

Alternative method used to program an application into a NetBurner device by posting the application file to the device's config server. It is invoked as part of the make system for NetBurner development kit NBEclipse and command line builds. It can be replaced with WGET. The format is `nbupdate <file name> <device IP>`

7.4 Smart Traps

While smart traps is not a specific PC application, the interface is accomplished by running the MTTTY serial terminal. During software development it is possible for a software developer to download an application with a coding error severe enough to cause an application to crash. Common causes of a system crash are: task stack overflow, indexing an array out of bounds and bad pointer assignments. When a system crash occurs a device will usually recover by a reboot of the system. The purpose of the SmartTrap utility is to provide more information about the system when the crash occurred to assist with debugging the problem. Smart traps are automatically enabled by the init() function. Note that smart traps are disabled when building an application in debug mode.

- SmartTraps are commonly used for debugging an application crash
- SmartTraps are used in conjunction with the MTTTY serial terminal program. The SmartTraps information will be displayed on the debug serial port

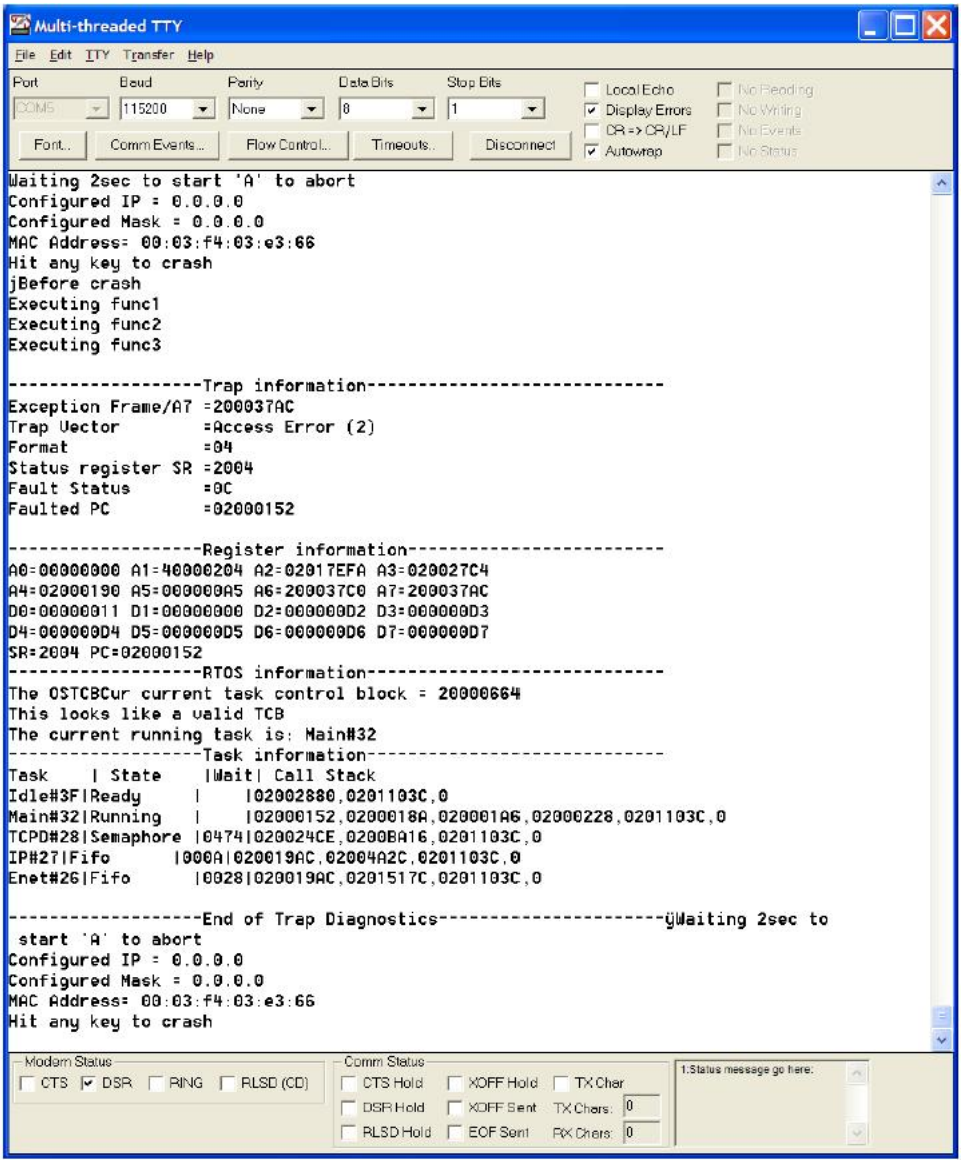


Figure 7.1 Example Smart Traps Output

The output has four sections. It is beyond the scope of this document to go into detail on each of the processor registers, please refer to the processor manual for more information, located in the `\nburn\docs` directory of your development tools installation.

7.4.1 Trap Information

This section displays the processor status and error registers. The most significant is the Faulted PC, with is the Program Counter register indicating the area where the fault occurred. In this example the program counter value at the time a trap was detected is 0x02000152.

7.4.2 Register Information

This section displays the processor's Address and Data registers.

7.4.3 RTOS Information

Identifies the RTOS Task Control Block and the current running task. The priority number is in hexadecimal. Traps caused by stack overflows may corrupt this section of the report.

7.4.4 Task Information

This is a very useful section of the report. It identifies each task, as well as its current state, wait time, and call stack. If you look at the Main task, you can see 6 entries with 0x02000152 (the faulted pc) at the left, and 0 at the right. The value of 0 is the start of the call stack. Each hexadecimal address between those two numbers represents each function that was called.

Each address can be related to the corresponding line number in the source code by using the WinAddr2Line utility. By selecting the .elf file location and each address in the call stack you can determine the calling sequence starting from 0 on the right and moving to the left. The last line signifies the end of the SmartTrap output. Since the boot monitor is configured to "boot to application", the device reboots after the trap occurs.

7.5 Task Scan

TaskScan is an network connected debugging tool that can be used to view the RTOS tasks and status of your running NetBurner application. This tool is unique in that it is active in the release build of your code, rather than the debug build which is compiled without optimization. It is very useful to determine such things as why a specific task is not running the way you had anticipated, if it is waiting on other tasks, and its call stack. TaskScan is only active when the PC program generates a request, so it will not impact your release code execution speed otherwise.

The init() function will automatically include taskmon.h and call EnableTaskMonitor(). TaskScan requires that you have the Executable and Linking Format (ELF) file for the specific application that is running on your NetBurner device. The .elf file is created by the compiler each time you build your application. It contains information TaskScan needs to link the application executable to the source code and task information. For example, if you have a project named MyProject, a file named MyProject.elf will be created.

TaskScan is commonly used for:

- Determining which tasks are running, their state and call stack.
- Looking for blocked tasks.

TaskScan can be started from:

- From the Windows NetBurner program group
- From a command line. The path is: `\nburn\pcbin\taskscan.exe`

7.5.1 TaskScan FAQ

- You must have a valid IP address and mask that can communicate with your PC that is running TaskScan
- TaskScan essentially takes a snapshot picture of what your application is doing at the time your press the Scan button. It does not otherwise impact the application
- You must have the exact .elf file used to create the application that is running on your NetBurner device to get valid information
  After starting the Task Scan application use the Browse button to locate your project's elf file, and the Find button to locate the IP address.

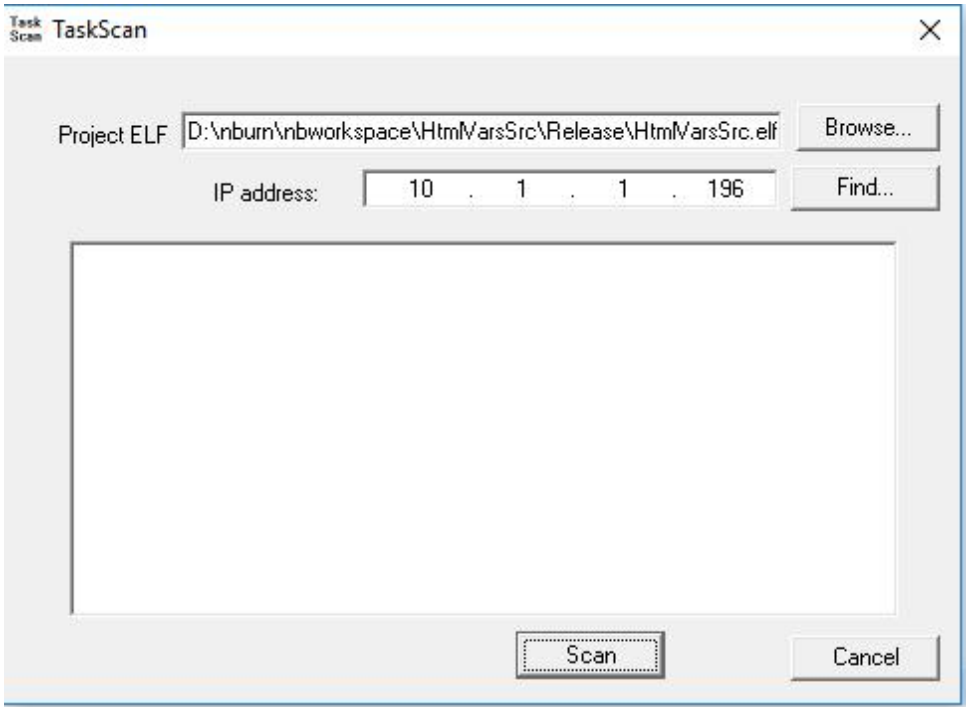


Figure 7.2 Starting Task Scan

Then select the Scan button:

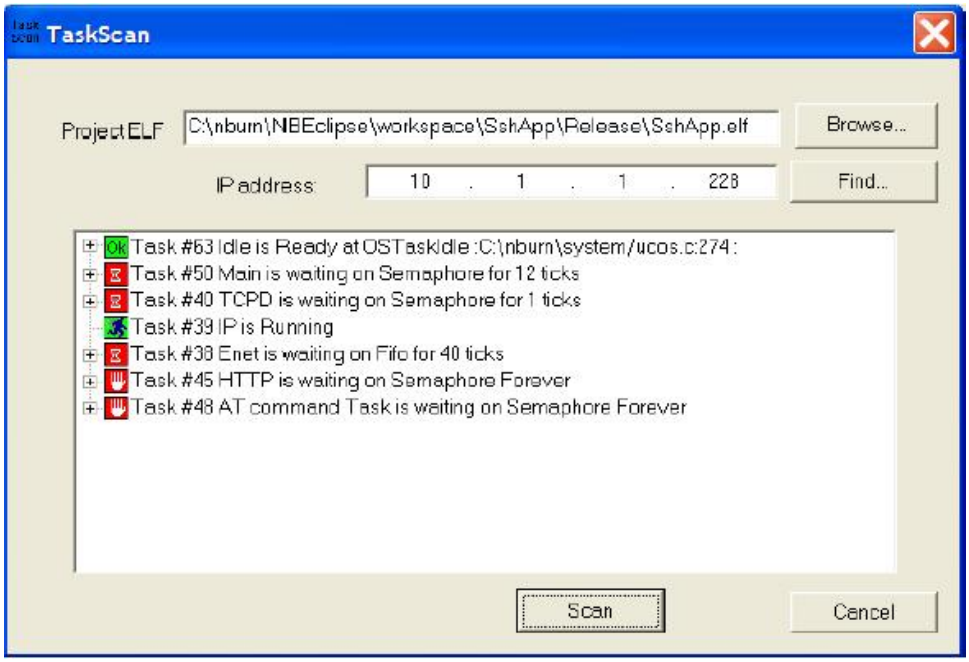


Figure 7.3 Scan Results

Icon	Description
Green OK	Task is ready to run
Green Running Man	Task is currently running
Red Hour Glass	Task is waiting for the specified number of time ticks
Red Hand	Task is waiting/blocked without a timeout parameter

Figure 7.4 Task Scan Icons

7.6 WinAddr2Line

Utility to determine the source code location for a specified memory address and .elf file. This application is typically used in conjunction with the SmartTraps utility when debugging an application that is crashing.

WinAddr2Line is commonly used for:

- Determining the source code address using a faulted program counter memory address from SmartTraps.
- If the cause of a crash is severe memory corruption due to bad pointers, it is possible for the Faulted Program Counter to be pointing to an address in which not code exists. In this case WinAddr2Line will not be able to provide a source code reference because no source code exists at that location.

WinAddr2Line can be started from:

- NBEclipse main tool bar
- The Windows NetBurner program group.
- The command line. The path is: `\nburn\pcbin\winaddr2line.exe`.

7.6.1 Using WinAddr2Line

- Use the Browse button to select the .elf file for your application and type in the hexadecimal memory address, usually obtained by SmartTraps as the Faulted PC.
- Click on the Decode button.

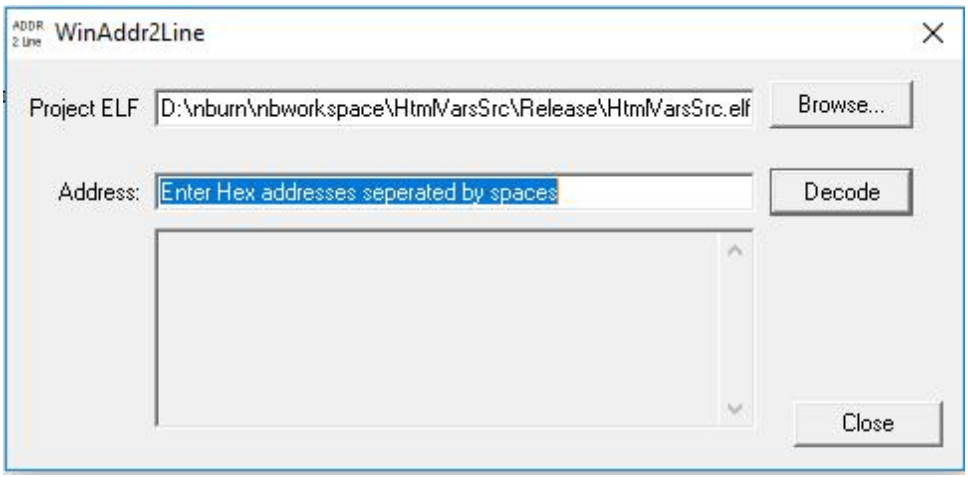


Figure 7.5 Select file and enter address in hexadecimal

7.7 IPSetup

The Windows IPSetup utility was used for network configuration in releases prior to 3.x. Network configuration is now down through the network configuration server on the NetBurner device. See Device Discovery and Configuration and Config Server Programming Guide for documentation on the 3.x network configuration system.

7.8 AutoUpdate

Autoupdate was used for loading applications to a device in releases prior to 3.x. Updating code is now done through the configuration server, nbupdate utility, or WGET.

7.9 Virtual Comm Port

The virtual comm port driver for Windows creates a virtual serial port and associates it with a network IP address and port number. It is typically used for legacy Windows applications without the ability to communicate over a network.

7.10 CompHtml

This utility is used by the NetBurner make system. It takes the contents of a project's html directory, implements the NetBurner dynamic web content scheme, callbacks, variables, access priorities, and mime types, and creates a file named htmldata.cpp.

7.11 CompCode

This utility is used by the NetBurner make system for 5441x platforms. It processes a .s19 file into a compressed binary with checksum and header.

7.12 flashpack

This utility is used by the NetBurner make system for ARM platforms. It processes the application.elf file into a compressed binary with checksum and header.

Chapter 8

Platform References

This section provides the hardware reference information for your NetBurner platform, including memory maps, boot monitor and module recovery procedures in the event you download an application that is crashing the device.

- MODM7AE70 Platform Reference
- MOD54415, MOD54417 Platform Reference
- NANO54415 Platform Reference
- SB800EX Platform Reference
- SBE70LC Platform Reference
- Recovery: MODM7AE70, SBE70LC
- Recovery: MOD5441x, NANO54415, SB800EX

8.1 MODM7AE70 Platform Reference

8.1.1 Introduction

This document provides the memory map and locations of reference materials for those who wish to add additional hardware to their NetBurner device.

8.1.2 MODM7AE70 Processor Information

The MODM7AE70 uses the Microchip ARM SAME70 microprocessor. The reference manual and datasheet provide in-depth information on the processor, including register settings, bus configuration and timing information. It is located in the `<nburn_install>\docs\Arm` directory of your NetBurner installation.

8.1.3 Datasheet

The datasheet for the MODM7AE70 module includes information on connectors, signal names, and operational parameters. It is located on the MODM7AE70 product page: `MODM7AE70 Datasheet`.

8.1.4 Development Board Schematic

The MOD-DEV-70CR development board schematic is located in the `<nburn_install>\docs\Net↩Burner\platform\Schematics` directory. This schematic can be used for design ideas in your own hardware implementation for power, RS-232, RS-485, and SD Flash card implementation.

8.1.5 Memory Operation

The MODM7AE70 uses 3 types of memory:

- 2MB Flash memory for non-volatile system configuration storage, user application storage, the application in a compressed format, and the optional EFFS-STD file system (note this is different from the EFFS-FAT flash card file system).
- 8MB SDRAM containing the uncompressed application as well as any application data.
- 384KB on-chip SRAM located in the SAME70 microprocessor.

At power-up the application is decompressed from Flash memory to SDRAM and execution begins. If the application is continuously trapping the device can be recovered using the jumper recovery method as described in the Recover section, or if you have serial communication enabled the boot sequence can be aborted by sending the appropriate serial command when prompted after a power cycle.

8.1.6 Memory Map

Region	Size	Address Range	Description
Boot Loader	6k	0x00400000 to 0x004017FF	Initial boot and application extraction
Config Record	10k	0x00401800 to 0x00403FFF	System and User Configuration Data Flash
Cert. Storage	8k	0x00404000 to 0x00405FFF	Optional Security certificate storage
User Flash	8k	0x00406000 to 0x00407FFF	Unstructured binary User Flash Storage
App Flash	1.97M	0x00408000 to 0x005FFFFF	Compressed Application and opt file system
Vector Table	-	0x20400000 to 0x2040013F	SRAM - Interrupt Vector Table
Fast App RAM	384k	0x20400180 to 0x2045FFFF	SRAM - Fast Application RAM
Backup RAM	1k	0x40074000 to 0x400743FF	Low power backup RAM
External Bus	-	0x60000000 to 0x60FFFFFF	Chip Select 0
External Bus	16M	0x61000000 to 0x61FFFFFF	Chip Select 1
External Bus	16M	0x62000000 to 0x62FFFFFF	Chip Select 2
External Bus	16M	0x63000000 to 0x63FFFFFF	Chip Select 3
App RAM	8M	0x70000000 to 0x707FFFFF	SDRAM - Standard Application RAM where code is decompressed and variables are allocated
QSPI XIP	-	0x80000000 to 0x80FFFFFF	QSPI (User add-on) - For utilizing memory Mapped Quad SPI memories

8.1.7 External Bus Interface

The MODM7AE70 provides 17 address lines (A0-A16) and a 16-bit data bus line (D0-D15). The data and address lines shared with the SDRAM are buffered to the module's P1 header pins. Please refer to the MODM7AE70 schematic and datasheet for connector and signal locations. The Microchip SAME70 datasheet, section 33 on External Bus Interface (EBI) and section 34 on Static Memory Controller (SMC), provide details on peripheral functionality and configuration.

8.2 MOD54415, MOD54417 Platform Reference

8.2.1 Introduction

This document provides the memory map and locations of reference materials for those who wish to add additional hardware to their NetBurner device. The MOD54415 and MOD54417 are very similar. The difference is the

MOD54417 provides two hardware network interfaces. The designation MOD5441x in this document applies to both modules.

8.2.2 MCF5441x Processor Information

The MOD54415 uses the NXP MCF54415 microprocessor, and the MOD54417 uses the MCF54417 microprocessor. The NXP MCF5441x reference manual and datasheet provide in-depth information on both processors and is located in the <nburn_install>\docs\NXP directory of your NetBurner Network Development Kit installation.

8.2.3 Datasheet

The datasheet for the MOD5441x module can be found on the NetBurner website: MOD5441x Datasheet. The datasheet is the primary control document for connector pin-outs and signal functions.

8.2.4 Development Board Schematic

The MOD-DEV-70CR development board schematic is located in the <nburn_install>\docs\NetBurner\platform\Schematics directory. This schematic can be used for design ideas in your own hardware implementation for power, RS-232, RS-485, and SD Flash card implementation.

8.2.5 Memory Operation

The MOD5441x uses 4 types of memory:

- A 4MB write protected serial SPI Flash chip containing the Boot Monitor and Alternate Boot monitor. This provides a recovery method should events such as a bad application that causes continuous traps, or application Flash memory corruption.
- A 32MB parallel Flash chip containing non-volatile system configuration storage, user application storage, the application in a compressed format, and the optional EFFS-STD file system (note this is different from the EFFS-FAT flash card file system).
- A 64MB DDR2 Memory chip that contains the uncompressed application as well as any application data.
- 64KB on-chip SRAM located on the MCF5441x microprocessor.

Note

The reference to "2.x" is there for customers migrating from the NetBurner 2.x development tools. 2.x System Configuration is not used when running 3.x applications.

4MB SPI Boot Flash	32MB Parallel Flash	64MB DDR2 RAM
Boot Monitor	2.x System Configuration	Application (decompressed)
Alternate Boot Monitor	User Parameters	System Stack
	Application (compressed)	
	3.x System Configuration	

The Boot Monitor executes at power-up and attempts to decompress the application into DDR2 RAM and begin execution. If the application is corrupted or does not exist, it will attempt to run the Alternate Boot Monitor. If the Alternate Boot Monitor cannot be run, the system will remain in the Boot Monitor. The Alternate Boot Monitor provides network communication to enable an application download, and is the normal recovery method should

unrecoverable application errors occur. The Boot Monitor only provides serial communication.

8.2.6 Memory Map

The MOD5441x provides 16 address lines (A0-A15) and a 16-bit data bus (D16-D31) running in mode 2. An address bus latch is included on the MOD5441x so external latch is not required for this mode. If you are adding peripherals to your NetBurner device's address/data bus, you can choose unused memory locations from the table below.
Once a range has been selected, you will need to configure the appropriate chip select address and option registers in the MCF5451x processor. Please refer to the chip select sections of the NXP MCF5441x processor manual for details on the register configuration. Unlike many ColdFire processors, you are not free to use any unused address for chip selects. The range is restricted. In this table we have used "undefined" for restricted ranges and "unused" for ranges that are valid for additional chip selects. Please see the MCF5441x Manual Section 20.3.1 and the Note above Figure 20-1 for additional details.

Memory Region	Address Range	Description
Undefined	0x00000000 to 0x01FFFFFF	Undefined area to catch null pointers
Unused	0x02000000 to 0x3FFFFFFF	Available to applications
DDR2 RAM	0x40000000 to 0x43FFFFFF	64MB of DDR2 RAM
VBR	0x40000000 to 0x400003FF	1kB processor vector base register
RAMBAR	0x80000000 to 0x8000FFFF	64kB Processor internal SRAM
Parallel Flash	0xC0000000 to 0xC1FFFFFF	32MB parallel flash memory
System Config	0xC0000000 to 0xC001FFFF	128kB system configuration flash sector
2.x Config Rec	0xC0000000 to 0xC0000400	1kB 2.x configuration storage
3.x Certificate	0xC0000400 to 0xC0002400	8kB 3.x certificate storage
3.x System Config.	0xC0002400 to 0xC001FFFF	119kB 3.x system configuration
User Params	0xC0020000 to 0xC003FFFF	128kB user parameter storage
Application Code	0xC0040000 to 0xC1FFFFFF	Compressed application code
Unused	0xC2000000 to 0xDFFFFFFF	Available to applications
IPSBAR	0xE0000000 to 0xFFFFFFFF	Processor internal device registers - accessible using the SIM structure defined in `sim5441x.h`

8.3 NANO54415 Platform Reference

8.3.1 Introduction

This document provides the memory map and locations of reference materials for those who wish to add additional hardware to their NetBurner device.

8.3.2 MCF54415 Processor Information

The NANO54415 uses the NXP MCF54415 microprocessor. The reference manual and datasheet provide in-depth information on the microprocessor, including register settings, bus configuration and timing information. It is located in the `<nburn_install>\docs\NXP` directory of your NetBurner installation.

8.3.3 Datasheet

The datasheet for the NANO54415 module includes information on connectors, signal names, and operational parameters. It is located on the NANO54415 product page: `NANO54415 Datasheet`.

8.3.4 Development Board Schematic

The NANO54415 development board schematic is located in the <nburn_install>\docs\Net↩ Burner\Platform\Schematics directory. This schematic can be used for design ideas in your own hardware implementation for power, RS-232, RS-485, and SD Flash card implementation.

8.3.5 Memory Operation

The NANO54415 uses 4 types of memory:

- A 4MB write protected serial SPI Flash chip containing the Boot Monitor and Alternate Boot monitor. This provides a recovery method should events such as a bad application that causes continuous traps, or application Flash memory corruption.
- A 8MB serial SPI Flash chip containing non-volatile system configuration storage, user application storage, the application in a compressed format, and the optional EFFS-STD file system (note this is different from the EFFS-FAT flash card file system).
- A 64MB DDR2 memory chip that contains the uncompressed application as well as any application data.
- 64KB on-chip SRAM located on the MCF54415 microprocessor.

The Boot Monitor executes at power-up and attempts to decompress the application into DDR2 RAM and begin execution. If the application is corrupted or does not exist, it will attempt to run the Alternate Boot Monitor. If the Alternate Boot Monitor cannot be run, the system will remain in the Boot Monitor. The Alternate Boot Monitor provides network communication to enable an application download, and is the normal recovery method should unrecoverable application errors occur. The Boot Monitor only provides serial communication.

Note

The reference to "2.x" is there for customers migrating from the NetBurner 2.x development tools. 2.x System Configuration is not used when running 3.x applications.

4MB SPI Boot Flash	8MB SPI Flash	64MB DDR2 RAM
Boot Monitor	2.x System Configuration	Application (decompressed)
Alternate Boot Monitor	User Parameters	System Stack
	Application (compressed)	
	3.x System Configuration	

8.3.6 Processor Runtime Memory Map

The runtime memory map for the location of the microprocessors onboard SRAM and DDR2 RAM is shown below:

Memory Region	Address Range	Description
Undefined	0x00000000 to 0x01FFFFFF	Undefined area to detect null pointers
DDR2 RAM	0x40000000 to 0x43FFFFFF	64MB of DDR2 RAM
VBR	0x40000000 to 0x400003FF	1kB Processor Vector Base Register
RAMBAR	0x80000000 to 0x8000FFFF	64kB Processor internal SRAM
IPSBAR	0xE0000000 to 0xFFFFFFFF	Processor internal device registers

8.3.7 Application SPI Flash Memory

There is a total of 8MB SPI Flash memory used to store the compressed application and non-volatile parameters. Read/write operations are only accessible through system API functions. The memory does not appear in the microprocessor's address map.

SPI Address	Description
0x000000	8kB 2.x System Configuration Record storage
0x002000	8kB User Parameter Storage
0x004000	7.9MB Compressed Application Code
0x7EE000	8kB 3.x Reserved System Configuration Storage
0x7F0000	64kB 3.x System Configuration

8.4 SB800EX Platform Reference

8.4.1 Introduction

This document provides the memory map and locations of reference materials for those who wish to add additional hardware to their NetBurner device.

8.4.2 MCF54415 Processor Information

The SB800EX uses the NXP MCF54415 microprocessor. The reference manual and datasheet provide in-depth information on the microprocessor, including register settings, bus configuration and timing information. It is located in the `<nburn_install>\docs\NXP` directory of your NetBurner installation.

8.4.3 Datasheet

The datasheet for the SB800EX module includes information on connectors, signal names, and operational parameters. It is located on the SB800EX product page: SB800EX Datasheet.

8.4.4 Memory Operation

The SB800EX uses 4 types of memory:

- A 4MB write protected serial SPI Flash chip containing the Boot Monitor and Alternate Boot monitor. This provides a recovery method should events such as a bad application that causes continuous traps, or application Flash memory corruption.
- An 8MB serial SPI Flash chip containing non-volatile system configuration storage, user application storage, the application in a compressed format, and the optional EFFS-STD file system (note this is different from the EFFS-FAT flash card file system).
- A 64MB DDR2 memory chip that contains the uncompressed application as well as any application data.
- 64K on-chip SRAM located on the MCF54415 microprocessor.

The Boot Monitor executes at power-up and attempts to decompress the application into DDR2 RAM and begin execution. If the application is corrupted or does not exist, it will attempt to run the Alternate Boot Monitor. If the Alternate Boot Monitor cannot be run, the system will remain in the Boot Monitor. The Alternate Boot Monitor provides network communication to enable an application download, and is the normal recovery method should unrecoverable application errors occur. The Boot Monitor only provides serial communication.

Note

The reference to "2.x" is there for customers migrating from the NetBurner 2.x development tools. 2.x System Configuration is not used when running 3.x applications.

4MB SPI Boot Flash	8MB SPI Flash	64MB DDR2 RAM
Boot Monitor	2.x System Configuration	Application (decompressed)
Alternate Boot Monitor	User Parameters	System Stack
	Application (compressed)	
	3.x System Configuration	

8.4.5 Processor Runtime Memory Map

The runtime memory map for the location of the microprocessors onboard SRAM and DDR2 RAM is shown below:

Memory Region	Address Range	Description
Undefined	0x00000000 to 0x01FFFFFF	Undefined area to detect null pointers
DDR2 RAM	0x40000000 to 0x43FFFFFF	64MB of DDR2 RAM
VBR	0x40000000 to 0x400003FF	1kB Processor Vector Base Register
RAMBAR	0x80000000 to 0x8000FFFF	64kB Processor internal SRAM
IPSBAR	0xE0000000 to 0xFFFFFFFF	Processor internal device registers

8.4.6 Application SPI Flash Memory

There is a total of 8MB SPI Flash memory used to store the compressed application and non-volatile parameters. Read/write operations are only accessible through system API functions. The memory does not appear in the microprocessor's address map.

SPI Address	Description
0x000000	8kB 2.x System Configuration Record storage
0x002000	8kB User Parameter Storage
0x004000	7.9MB Compressed Application Code
0x7EE000	8kB 3.x Reserved System Configuration Storage
0x7F0000	64kB 3.x System Configuration

8.5 SBE70LC Platform Reference

8.5.1 Introduction

This document provides the memory map and locations of reference materials for those who wish to add additional hardware to their NetBurner device.

8.5.2 SBE70LC Processor Information

The SBE70LC uses the Microchip ARM SAME70 microprocessor. The reference manual and datasheet provide in-depth information on the processor, including register settings, bus configuration and timing information. It is located in the <nburn_install>\docs\Arm directory of your NetBurner installation.

8.5.3 Datasheet

The datasheet for the SBE70LC module includes information on connectors, signal names, and operational parameters. It is located on the SBE70LC product page: SBE70LC Datasheet.

8.5.4 Development Board Schematic

The SB70LC-ADPT-100 development board schematic is located in the <nburn_install>\docs\NetBurner\platform\Schematics directory. This schematic can be used for design ideas in your own hardware implementation for power, RS-232, RS-485, and SD Flash card implementation.

8.5.5 Memory Operation

The SBE70LC uses 3 types of memory:

- 2MB Flash memory for non-volatile system configuration storage, user application storage, the application in a compressed format, and the optional EFFS-STD file system (note this is different from the EFFS-FAT flash card file system).
- 8MB SDRAM containing the uncompressed application as well as any application data.
- 384KB on-chip SRAM located in the SAME70 microprocessor.

At power-up the application is decompressed from Flash memory to SDRAM and execution begins. If the application is continuously trapping the device can be recovered using the jumper recovery method as described in the Recover section, or if you have serial communication enabled the boot sequence can be aborted by sending the appropriate serial command when prompted after a power cycle.

8.5.6 Memory Map

Region	Size	Address Range	Description
Boot Loader	6k	0x00400000 to 0x004017FF	Initial boot and application extraction
Config Record	10k	0x00401800 to 0x00403FFF	System and User Configuration Data Flash
Cert. Storage	8k	0x00404000 to 0x00405FFF	Optional Security certificate storage
User Flash	8k	0x00406000 to 0x00407FFF	Unstructured binary User Flash Storage
App Flash	1.97M	0x00408000 to 0x005FFFFF	Compressed Application and opt file system
Vector Table	-	0x20400000 to 0x2040013F	SRAM - Interrupt Vector Table
Fast App RAM	384k	0x20400180 to 0x2045FFFF	SRAM - Fast Application RAM
Backup RAM	1k	0x40074000 to 0x400743FF	Low power backup RAM
External Bus	-	0x60000000 to 0x60FFFFFF	Chip Select 0
External Bus	16M	0x61000000 to 0x61FFFFFF	Chip Select 1
External Bus	16M	0x62000000 to 0x62FFFFFF	Chip Select 2

Region	Size	Address Range	Description
External Bus	16M	0x63000000 to 0x63FFFFFF	Chip Select 3
App RAM	8M	0x70000000 to 0x707FFFFF	SDRAM - Standard Application RAM where code is decompressed and variables are allocated
QSPI XIP	-	0x80000000 to 0x80FFFFFF	QSPI (User add-on) - For utilizing memory Mapped Quad SPI memories

8.6 Recovery: MODM7AE70, SBE70LC

8.6.1 Introduction

Applicable Hardware Platforms: MODM7AE70 and SBE70LC.
Device recovery in the event of an errant application running on the device is an important part of the development process. This document describes recovery procedures for a device stuck in a repeating trap/reset condition, or a totally unresponsive device.

8.6.2 Repeating Traps or Resets

The easiest method of recovery is to abort the user application boot process and run only the Configuration Server. The Configuration server will enable a new application to be downloaded using:

- The Configuration Server web page at `<device IP>:20034`
- The nbupdate utility
- The wget utility
- The device serial port

8.6.2.1 Step 1: Abort to the Configuration Server

1. Run a serial terminal such as MTTTY and connect to the device serial port.
2. Power cycle the module.
3. Press "A" when the message, `Type "A" to Abort boot` is displayed. Note that "A" is the default abort command, but the command is configurable in the Configuration Server settings. If the abort command is different, use that command instead of "A".
4. Wait for the > prompt to display.

The Configuration Server is now active on both network and the device serial port.

8.6.2.2 Step 2: Download a New Application

At this point a new application can downloaded through the device web page, nbupdate or serial port.

8.6.2.2.1 Web Page

1. Locate the device using discover.netburner.com, the localdiscover utility, or open a web browser and type the device address in the URL field: `<device IP>:20034`. For example, `192.168.1.10:20034`.
2. In the Update Application section, select the new .bin file to download and select the Send File button.

8.6.2.2.2 NBUpdate

1. Open a command prompt.
2. Type `nbupdate <file name> <device IP>`

8.6.2.2.3 Serial Port

Note

MTTTY is a serial terminal program. You should be able to use any serial terminal program of your choice. When starting the download the binary file must be sent as text or raw data.

Warning

The .bin file is a binary file. You must have serial software flow control disabled. If using MTTTY, from the main menu select `TTY > Flow Control` and uncheck both Xon/Xoff Input Control and Xon/Xoff Input Control. Otherwise the data stream will be corrupted.

1. At the `>` prompt, type `fla` into the serial prompt and then hit enter.
2. At the message 'Begin Download', start the file download (if using MTTTY, type the `F5` key).
3. When the file window opens, navigate to the `.bin` application file that you want to load.
4. Hit 'Open', and the application will start the load process.

8.6.3 Unresponsive Device or No Serial Port Access

In a situation in which the device is completely unresponsive and aborting to the Configuration Server is not possible, or if there is no access to the device serial port, shorting two hole locations on the device PCB (also called "jumper recovery") will reset it to a factory default state. The PCB hole locations are 0.1 inches apart so that a shorting jumper can be used. This procedure will erase flash memory including the configuration and application. The device will then be ready to receive a new application using any of the procedures described in the download a New Application section.

1. Power off the device.
2. Short the jumper recovery PCB locations with a shorting jumper or wire.
3. Power on the device.
4. Wait for the device to finish erasing the flash and loading the default application. Progress is indicated by the two blinking LEDs on the device (D1 and D2). When the LEDs stop blinking the reset is complete. For the SBE70LC, the jumper should be removed after 2 seconds. If it is left inserted for 30 seconds or more, it will continually reset.
5. Remove the short jumper or wire.

8.6.3.1 Recovery Jumper Locations

8.6.3.1.1 MODM7AE70 The reset jumper location is labeled "Recovery", located on the end of the board opposite the RJ-45 connector.

Figure 8.1 MODM7AE70 Recovery Jumper Location

8.6.3.1.2 SBE70LC The reset jumper location is labeled "JP6", located on the end of the board opposite the RJ-45 connector.

Figure 8.2 SBE70LC Recovery Jumper Location

8.7 Recovery: MOD5441x, NANO54415, SB800EX

NetBurner 5441x based devices include: MOD54415, MOD54417, NANO54415 and the SB800EX. These devices have a Recovery Jumper. This is a 2 hole pin location on the circuit board. When shorted during the boot process, the device will boot to the Alternate Boot Monitor rather than the application. If the configuration section of flash has been corrupted, it will also use a safe default version instead of what is in flash memory.

8.7.1 Step 1: Abort to the Alternate Boot Monitor

There is the primary boot monitor, accessible by entering the 'A' (0x41) character immediately on boot. This should allow you to recover the module normally. If for some reason that fails, or you cannot enter the 'A' character, the

have a secondary boot monitor to allow the user to recover from (virtually) all software or configuration faults. The procedure is:

1. Power off the module.
2. Short the 2 pins at the alternate boot jumper location (see below).
3. Power on the module.
4. Wait for the module to finish boot and enter the monitor.
5. Remove the short on the alternate boot jumper location.

8.7.2 Recovery Jumper Locations

8.7.2.1 NANO54415

The boot jumper is a pair of circular pads located near the middle line of the board near the connector.

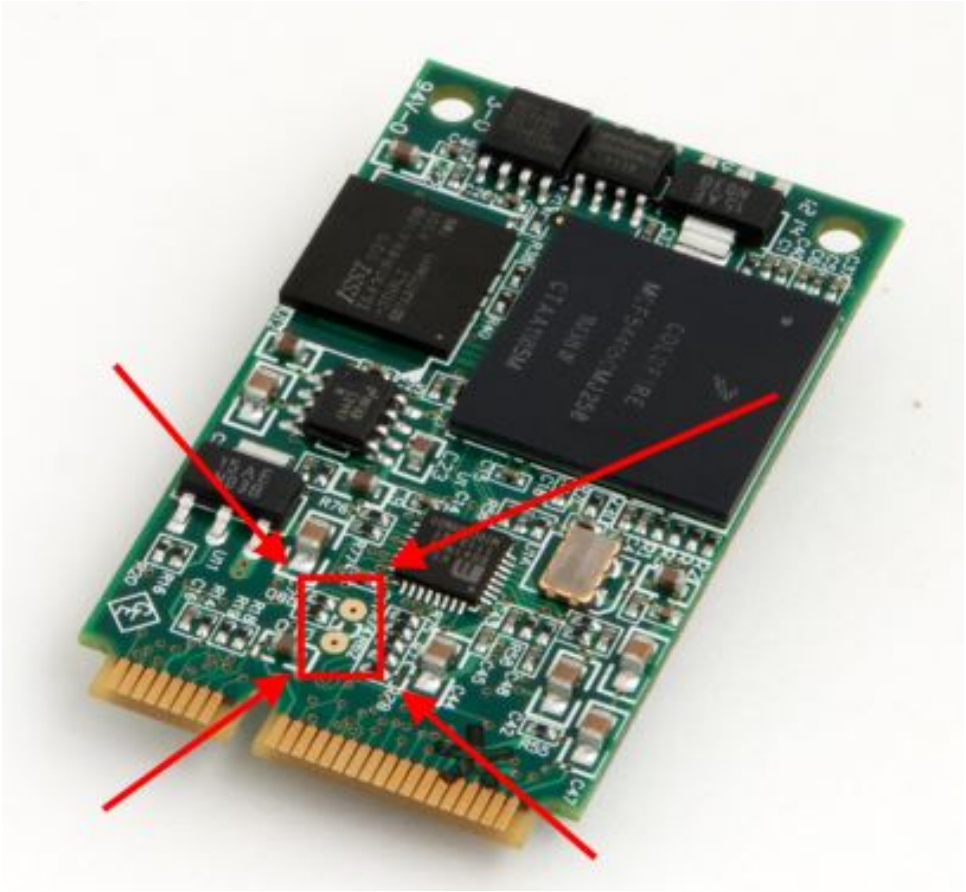


Figure 8.3 NANO54415 Recovery Jumper Location

8.7.2.2 MOD5441x

The boot jumper is the unpopulated header 'TP1', located near the Ethernet jack.

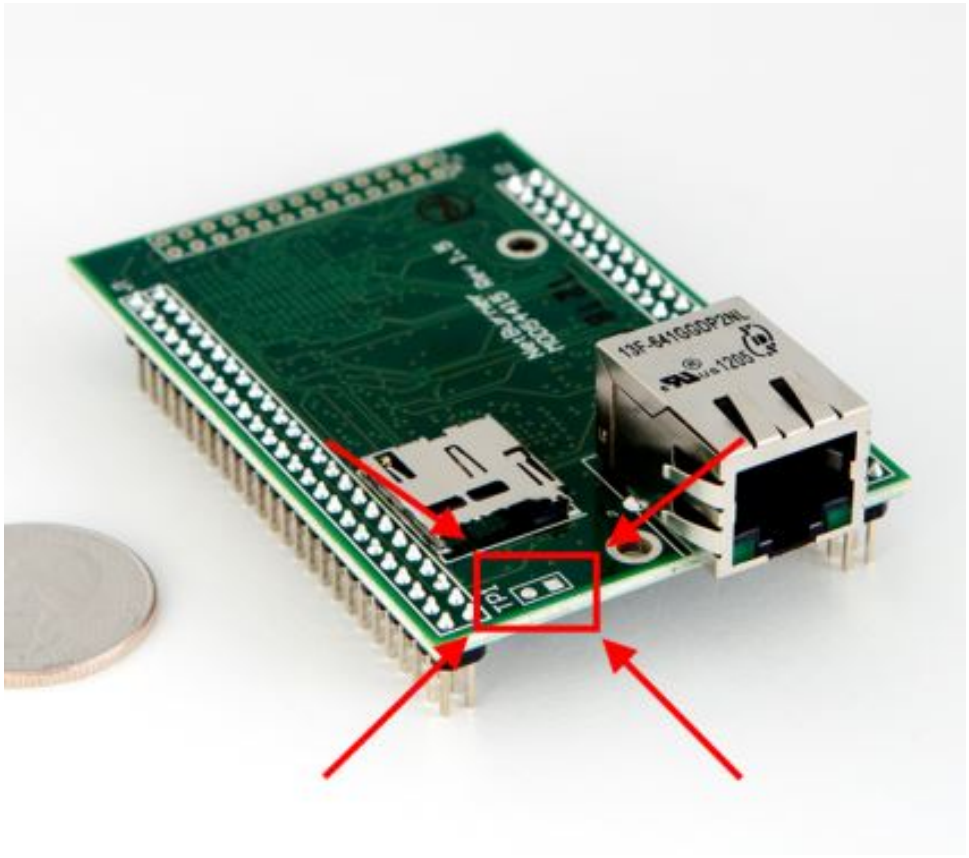


Figure 8.4 MOD5441x Recovery Jumper Location

8.7.2.3 SB800EX

The boot jumper is the unpopulated header 'JP1', located near the center of the board.

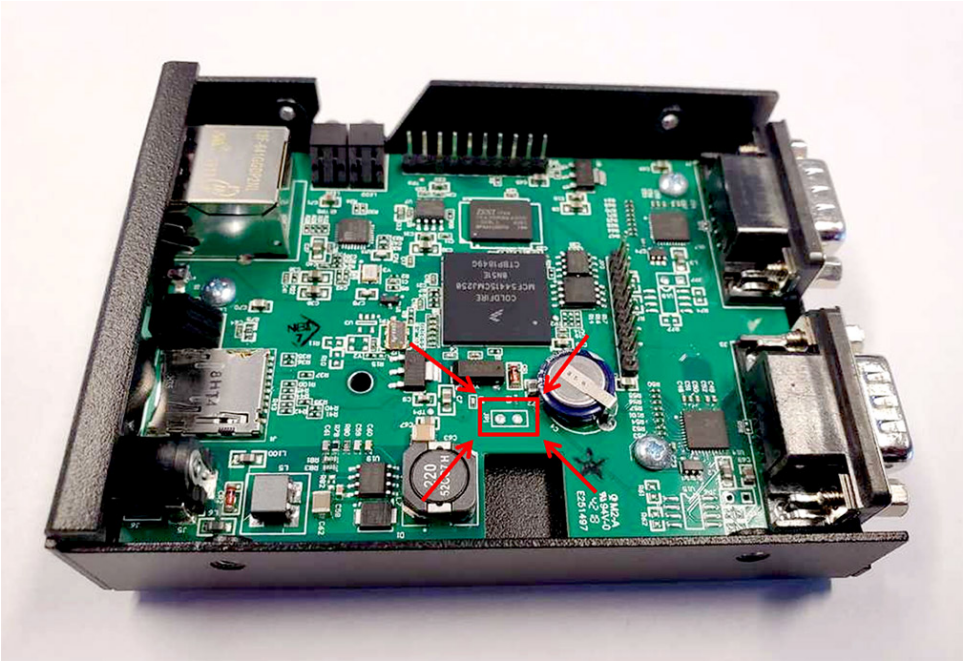


Figure 8.5 SB800EX Recovery Jumper Location

8.7.3 Step 2: Download a New Application

Once the Alternate Boot Monitor is running you should be able to locate your device using:

8.7.3.1 Network

The 5441x devices predate the NetBurner 3.x Development Tools and network recovery requires the use of the AutoUpdate utility. Run the AutoUpdate utility and select the device and new application to download. Autoupdate requires _APP.s19 file format. The original, factory application is a safe application to load on to a device that needs to be recovered. It can be found in the `platform\<PLATFORM>\original` directory.

8.7.3.2 Serial Port

Note

MTTTY is a serial terminal program. You should be able to use any serial terminal program of your choice. When starting the download the binary file must be sent as text or raw data.

Warning

The .bin file is a binary file. You must have serial software flow control disabled. If using MTTTY, from the main menu select `TTY > Flow Control` and uncheck both Xon/Xoff Input Control and Xon/Xoff Input Control. Otherwise the data stream will be corrupted.

1. At the `NB>` prompt type `fla` then hit enter.
2. At the message 'Begin Download', start the file download (if using MTTTY, type the `F5` key).
3. When the file window opens, navigate to the `.bin` application file that you want to load.
4. Hit 'Open', and the application will start the load process.

Once the download is complete the device will reboot and run the new application.

Chapter 9

Production & Deployment

9.1 Production Releases and Customer Deployment

Once product development is complete, there are a number of choices for device location and configuration:

- How to find the device on a network
- How to program the application into devices in production
- How to configure devices in production
- How Customers can change configuration settings

Note

Utility source code is located in `\nburn\pctools`, and built versions are in `\nburn\pcbin`. You may modify or rebrand these utilities and distribute to your end customers, so long as they are only run with NetBurner target hardware.

9.2 Finding a Device on a Network

A device can be located the following ways:

- If the device and computer have Internet access, open a web browser and go to discover.netburner.com
- Use the localdiscover utility. This is a multi-platform utility written in Golang for Windows, macOS and Linux. The compiled version can be distributed to your customers, but not the Golang source code.
- The Python find utility, located in `\nburn\pctools\find`.
- Use the debug/console serial port

9.3 Application Programming

Applications are programmed into flash memory by sending an application image to the device's configuration server, located at its IP address on port 20034. This can be done in a number of ways, all of which are multi-platform (Windows, macOS, and Linux):

- A web browser from the device's configuration server web page
- The nbupdate command-line utility
- The WGET command-line utility
- A serial port using the serial command processor

9.3.1 Configuration Web Page

To send a file from a web browser, enter the URL of your device. The URL format is "<ip address>:<port number>", such as: "10.1.1.100:20034". Once at the web page, browse for the application image file and select "Update".

9.3.2 nbupdate Utility

nbupdate is a multi-platform command-line utility written by NetBurner. The format is: `nbupdate <filename> <ip address>`. For example, `nbupdate myApp.bin 10.1.1.196`, or `nbupdate myApp.s19 10.1.1.196`.
If the environment variable DEVIP is set the IP address parameter is not required. For example, `SET DEVIP=10.1.1.196`. Now you can call nbupdate with only the application file name. To determine the current setting of DEVIP, type "SET DEVIP".

9.3.3 wget Utility

WGET is an industry standard utility that allows you to download files over TCP/IP protocols FTP, HTTP and HTTPS. If you are using macOS or Linux, WGET is usually already installed (or can using your package manager). Windows users will need to install it, please refer to https://www.gnu.org/software/wget/.
The WGET utility can be used for a number of things:

- Updating the device application image
- Retrieving device configuration information in JASON
- Updating device configuration information in JASON
  In the examples below we use an example device IP address of 10.1.1.60, and an application name of myApplication.bin.

9.3.3.1 Program an Application

To program a new application into the device:
`wget --post-file=myApplication.bin 10.1.1.60:20034/appupdate.html`
All updates are processed by the Configuration Server running on the device. The `--post-file` specifies a POST operation is to be performed. The IP address and configuration port number are next, along with the server post destination that will always be appupdate.html.

If using authentication with a username of "user", and password of "pass":
`wget --auth-no-challenge --user=user --password=pass --post-file=myApplication.bin 10.1.1.60:20034/appupdate.html`

9.3.3.2 Read The Configuration

Read the current configuration into a file named config.txt: `wget 10.1.1.60:20034/Config`.
In this case the destination is Config, rather than appupdate.html.
If you would like to specify a directory to store the file, use the `-P` option: `wget 10.1.1.60:20034/Config -P \myDir`

9.3.3.3 Program New Configuration Settings

The easiest way to do this is to first read the configuration and save to a file so you have a template to work with, then edit the settings you need. Then post the configuration file to each new device:
`wget --post-file=newconfig 10.1.1.60:20034/Config`

9.4 Customer Configuration

Once deployed, your customers can modify the configuration by:

- Using the interactive web interface at 10.1.1.60:20034. Note that you can create your own web interface to override the default NetBurner interface, and use your custom brand.
- If connected to the Internet, a web browser at: discover.netburner.com, which provides a link to the device configuration page and also the device's main web page on port 80.
- The NetBurner localdiscover utility, which provides a link to the device configuration page, and also the device's main web page on port 80.
- Using WGET
- Your application can provide the means to change configuration

Chapter 10

Programmers Guide

The documents in this section are provide Programmers with background information and examples on the various aspects of the NetBurner development system.
Topic Links:

- Introduction
- Config Server Programming Guide
- Custom Branding
- IPv4/IPv6 Dual Stack Guide
- EFFS Programming Guide
- File Descriptors
- HTML Processing
- Interrupt Handling
- JSON Lexer
- NetBurner RTOS
- Network Protocols
- SSL/TLS Programming Guide
- SSH Programming Guide
- Web Server

10.1 Introduction

The NetBurner Programming Guide is intended to provide an overview of the features and capabilities of the NetBurner Network Development Kit. The primary goal of this guide is to provide a brief explanation of common network applications and illustrate how you can implement these applications using NetBurner hardware, software and development tools. Prerequisites for this guide:

- You have installed the NetBurner Network Development kit
- You have a running NetBurner device that is configured for your network, and you are familiar with the NBEclipse development (ref: NBEclipse Getting Started Guide), or you are using your own environment and the command line tools.
- You are familiar with the network configuration of your target device
- You have successfully created a project and can run applications on your target device

The approach of this guide is to learn by example. The first program example, called Template, can be used as a starting point for most applications, and each application in this guide uses it as a base.

10.1.1 Source Code for Example Programs

Source code for the examples in this document are located in the `\nburn\examples` directory of your NetBurner tools installation.

10.1.2 Tools and Library Version Information

This documentation applies to the following version of the NNDK, which utilizes the listed external tools and libraries.
NetBurner Network Development Kit Tools 3.3.2
wolfSSL 4.3 Utilized by the SSL/TLS library.
GNU GCC 8.1 The documentation for this compiler can be found can be found here.

10.1.3 Application Wizard Project

We will start with a basic application created with the NBEclipse Application Wizard. If you are not familiar with creating an application or NBEclipse please refer to the NBEclipse Getting Started Guide before continuing. When running the Application Wizard select the Standard Initialization and Web server options. Once the project is complete, you should have a main.cpp file as shown below:

```
#include <predef.h>      // System level options and definitions
#include <stdio.h>       // Standard I/O functions
#include <nbrtos.h>      // NetBurner Real Time Operating System (RTOS)
#include <http.h>        // HTTP functions
#include <init.h>        // Initialization functions

const char * AppName = "AppWizard";     // Name of application. Will be displayed by discovery programs

void UserMain(void * pd)
{
    init();                                          // Initialize system
    WaitForActiveNetwork(TICKS_PER_SECOND * 5);      // Wait up to 5 seconds for active network activity
    StartHttp();                                     // Start HTTP web server. Note StartHttps() starts
      secure web server

    iprintf("Application %s started\n", AppName );  // Print message to stdout, which is the debug serial
      port by default

    while (1)                                        // Infinite loop
    {
        OSTimeDly(TICKS_PER_SECOND);
    }
}
```

The system will automatically start the RTOS and UserMain() as a its own task. This simple application is a fully functional network application with a web server.
The init() function will do the following:

- Initialize stdio to be the debug/console port.
- Read and process the system configuration information. This includes things such as network interface settings, serial port settings and boot options.
- Initialize the network stack.
- Set the RTOS task priority of UserMain() to MAIN_PRIO.
- Enable the Task Monitor utility support.

- Whenever the project is built in debug mode, enable the GDB debugger.

The WaitForActiveNetwork() function will wait for an active network link before proceeding, up until the specified timeout.
StartHttp() starts the web server. The default port is 80. If you wish to start on a different port you can specify the port number as a parameter to the function.
The system supports `printf()` and `iprintf()`. The `iprintf()` function (i = integer only) will consume less system resources if you do not need floating point support.
Your application should never return from `UserMain()`; the while() loop will run forever. The
`#define TICKS_PER_SECOND`
should be used with the delay function in case the system ticks per second value is ever modified.

10.2 Config Server Programming Guide

10.2.1 Introduction

The NetBurner configuration system is used to configure and store system and application defined values and data in flash memory. These values can be accessed and modified through:

- The System Configuration Web Server (port 20034)
- Your application
- Your web page interface
- Serially through the Serial Config Server

The web interface also provides a convenient way of uploading a new application onto the module. While the web interface provided is fully functional, it is possible override this with a custom web interface in your application for purposes such as branding or adding/removing features.
The configuration data itself is stored and presented as JSON blobs. It's possible to access, modify, and send the entire JSON blob, or any subsection of the data, down to individual components.

10.2.2 Configuration Parameters

The configuration server holds the module's boot settings, network interface settings, and any custom application information you wish to save, display or modify. Your application can view and/or modify these parameters a variety of ways:

- The system configuration server web page
- Your own custom configuration server web page
- Any type of web interface in your application
- In your application code directly

The configuration system has a tree structure with "leafs". There are a number of built-in leafs described below, such as "Config", "AppData", and "Boot".

10.2.2.1 Config

`Config` is the top level object that contains everything else. Underneath are the `AppData` and `Sys` settings. There is also a `Version` field that is updated automatically when new settings are saved. The `Reboot` option, when checked, will force the module to reboot when the configuration record is updated.

10.2.2.2 Config.AppData

The AppData leaf is set aside at the top leaf for user application data. The application can select from the many configuration objects to store strings, integers, list boxes, etc. More information on how to use these can be found in the BasicConfigVariable NetBurner Basic Config Demo.

10.2.2.3 Config.Sys

The `Sys` leaf contains the device system settings such as boot parameters, application name, and network interface settings.

10.2.2.3.1 Config.Sys.Boot Boot settings include:

- `Abort`: This specifies the keys that needs to be pressed in order to break out of the normal boot sequence. This prevents anything after the `init()` function from running in `UserMain()`. This will load the serial configuration server, enable the config web interface and effectively stops the user's application from running. The default value for this is "A".
- `BootBaud`: The baud rate that the `BootUart` will use.
- `BootDelay`: How long the application will delay to wait for the `Abort` characters before continuing.
- `BootQuiet`: Whether or not to display the boot output data on the `BootUart`.
- `BootUart`: The UART used by the system to display output related to the device booting.
- `Password`: When this is set along with `User`, both will be required when making changes to the congif settings, or when updating the user application.
- `Serial Config`: There are four different options for this, which are outlined below.
  - **DuringBoot**: This will make the serial configuration server available during the boot process by using the `Abort` characters set above.
  - **AlwaysEnabled**: This will set the serial configuration server to always be available through the `Boot↩Uart`. Note that if this is selected, attempting to receive other serial input through the `BootUart` (such as debug commands, etc.) has the potential to break the configuration record.
  - **PauseAfterBoot**: This option will cause the boot process to pause indefinitely before the user's application is launched (specifically after `init()` is called), and will cause the serial configuration server to be enabled over the `BootUart`. The user's application can be resumed by typing `boot`.
  - **Disabled**: This prevents the serial config server from being enabled, and does not pause during the boot process, which stops the user from being able to enter the `Abort` character sequence.
- `User`: When this is set along with `Password`, it must be entered when trying to make changes to the configuration record's values or uploading a new application to the module.

10.2.2.3.2 Config.Sys.NetIf The settings underneath NetIf deal with the different network interfaces available on the module. This will include all of the Ethernet ports, as well as any Wifi ports available. What is actually listed underneath these interfaces will depend on the interface itself. Common settings are listed below:

10.2.2.3.2.1 Config.Sys.NetIf.Ethernet0 Ethernet settings include:

- `DeviceName`: A name given to the interface, which is used to register for DDNS and NetBIOS.
- `DhcpDiscoverSec`: How long to wait after boot before sending a DHCP Discover message.
- `DiscoveryReportInterval`: How often a device should report itself to the discovery server.
- `DiscoveryReportUrl`: The location of the discovery server. By default, this points to NetBurner's discovery server. However, if desired, this field can changed to be blank so that the device does not report itself. It can also be set to another URL if the user wishes to run their own discovery server.
- `MAC`: The MAC address of the interface.

Config.Sys.NetIf.Ethernet0.IPv4 IPv4 Settings include:

- `ActiveAddr`: The current IPv4 IP address in use.
- `ActiveDNS1`: The current IPv4 DNS server (1) in use.
- `ActiveDNS2`: The current IPv4 DNS server (2) in use.
- `ActiveGate`: The current IPv4 gateway in use.
- `ActiveMask`: The current IPv4 mask in use.
- `AutoIPAddr`: The current IPv4 auto address in use.
- `AutoIPEn`:
- `Mode`: Specifies which method to use to acquire IPv4 active values. The options are as follows:
  - `DHCP`: Use the values provided by the DHCP server.
  - `DHCP w Fallback`: Try to get values from the DHCP server, and use the static values as a backup.
  - `Static`: Use the static values defined in the interface's config record.
  - `Disabled`: Disable the interface.
- `StaticAddr`: The manually configured IPv4 IP address.
- `StaticDNS1`: The manually configured IPv4 DNS(1).
- `StaticDNS2`: The manually configured IPv4 DNS(2).
- `StaticGate`: The manually configured IPv4 gateway.
- `StaticMask`: The manually configured IPv4 mask.

Config.Sys.NetIf.Ethernet0.IPv6 IPv6 settings include:

- `ActiveAddr`: Lists the current IPv6 addresses in use.
- `ActiveDNS`: Lists the current IPv6 DNS servers in use.
- `ActiveRoutes`: Lists the current IPv6 routes in use.
- `Mode`: Specifies which method to use to acquire IPv4 active values. The options are as follows:
  - `DHCP`: Use the values provided by the DHCP server.
  - `DHCP w Fallback`: Try to get values from the DHCP server, and use the static values as a backup.
  - `Static`: Use the static values defined in the interface's config record.
  - `Disabled`: Disable the interface.
- `StaticAddr`: The manually configured IPv6 IP address.
- `StaticDNS1`: The manually configured IPv6 DNS(1).
- `StaticDNS2`: The manually configured IPv6 DNS(2).

10.2.3 The Configuration Web Interface

The device's configuration web server is available on port number 20034. To view in a web browser, type the device's IP address followed by the port number. For example, "10.1.1.100:20034". The default system configuration server web page is shown below. Note that you can easily create your own custom web page for things as simple as branding with your own logo, to formatting and adding fields in any way you wish. Clicking on the tree structure leafs will navigate through the settings.

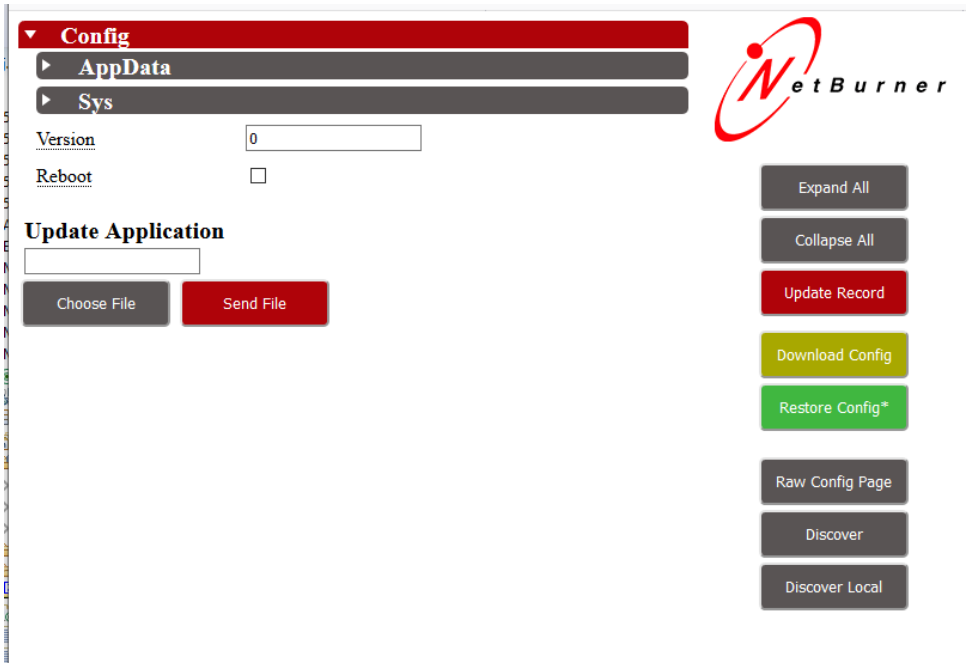


Figure 10.1 Config Web Interface

Located on the right side of the interface, you will also notice that there are several buttons. These provide the following functionality:

- `Expand All`: This expands the entire config tree structure.
- `Collapse All`: This collapses the entire config tree structure.
- `Update Record`: This updates the config record with the values that are currently entered in config structure on the web page.
- `Download Config`: Download the configuration object in JSON.
- `Restore Config`: Upload a JSON configuration object.
- `Raw Config Page`: Display a non-stylized version of the web interface, which can be edited.
- `NB Discover`: This takes you to `NetBurner's Discover` page.
- `Discover Local`: Discover device on the local network, Internet access in not required.

The Update Application section provides for an application update by selecting a .bin file. Use the Choose File button to select the image, followed by selecting Send File.

10.2.4 Serial Configuration Interface

In addition to the web interface, it's possible to navigate and modify the config record through a serial terminal. To do this, you will need to enter the serial config server through the `BootUart` using a method defined by the `SerialConfig` option in the Boot settings outlined above.
The most common method will be to type in the string specified by the `Abort` setting listed in the Boot configuration section. Reset the module and send the abort string before the boot timeout expires. The boot delay text is:
`Type \<Abort Value>\ to Abort boot...`
The default Abort Value is a capital 'A'. Upon successfully entering the serial configuration monitor, the > prompt will be displayed:

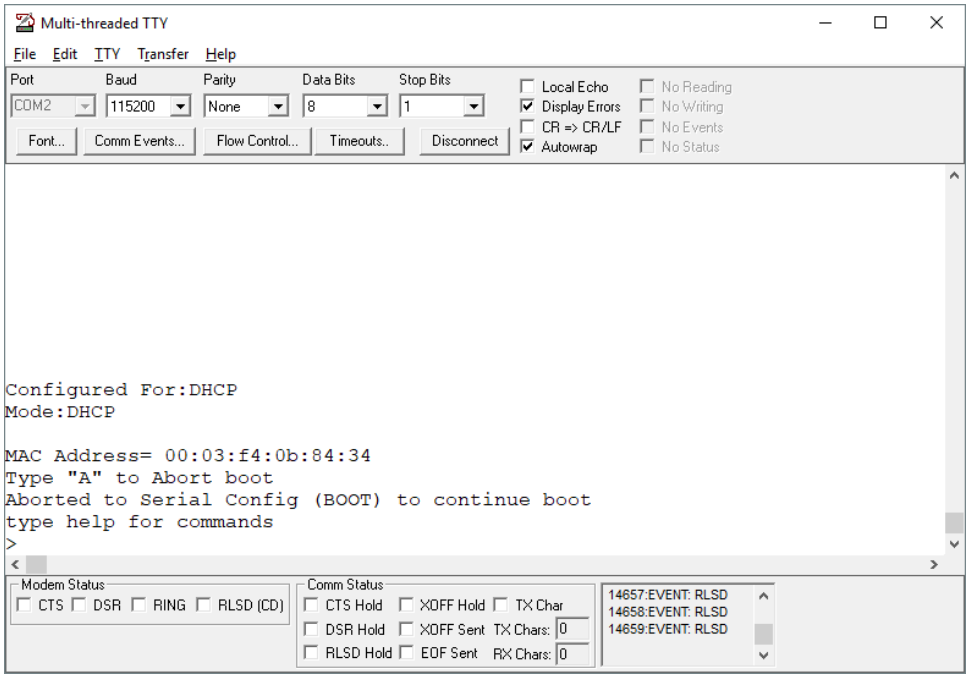


Figure 10.2 Serial Config Prompt

Navigating through the config data via the serial menu has been designed to operate much like a standard file system. Typing the name of a config object will move you into that section of the config data, much like navigating into a directory through a command prompt. Where you are in the config structure will be displayed at the prompt. For example, from the initial prompt, >, if you were to type `Config.Sys` and hit enter, the prompt would then read `Config.Sys.>`, illustrating that you are currently at the system settings for the config object.
In addition, several commands have been added to aid in navigating and modifying the config data, as well updating the device. Typing "help" at the prompt will display a list of these commands. Each of these is explained below:

- `help`: Displays the commands that can be used in the serial config system.
- `boot`: This will exit the serial config server, and continue to boot the user's application.
- `reboot`: This will reboot the module.
- `flash`: This will allow to you upload a new application serially. To do this in MTTTY, after typing this command hit "F5" to open a file dialog box and select a new `.bin` file to upload.
- `<configvar>?`: Type the name of a config variable followed by a "?" to list the config structure underneath the config variable specified. For example, typing `Config?` will show the entire contents of the config structure, while `Config.Sys` will show all of the system settings.
- `<configvar>=<value>`: This will assign a value to a specific config variable.
- `..`: This will take you up one level in the config object structure from the current position.

- `ls`: This will show the current level of the config object.
- `save`: This will save any changes made to the values stored in the config object.

10.2.5 Creating a Custom Web Interface

The config web interface was designed so that it could be easily replaced by customers with their own web interfaces and logos. To do this, simply replace `<NNDK Install Directory>\nbrtos\source\ROOT.html` with your own file, and rebuild the system libraries as outlined in the Rebuilding Projects & Libraries NBEclipse Guide.
If you simply want to replace the logo show in the config web interface, replace `LOGO.gif` with your own image, and rebuild the system libraries as mentioned above.
An example has been setup that shows how the current config web interface has been structured and built. It can be found in the examples folder of your NNDK install at `<NNDK Install Directory>\examples\Configuration\BasicWebConfig`.

10.2.6 Configuration API Examples

There are several examples that outline how to properly use NetBurner's config objects. These can all be found in the examples folder of your NNDK install at `<NNDK Install Directory>\examples\Configuration\`. They are listed here with brief descriptions for your convenience.

- BasicConfigVariable `Basic Config Variable`: This program shows the basics of using NetBurner's config system objects.
- BasicWebConfig `Basic Web Config`: This program shows the basics of using NetBurner's config system and presenting the information through a dynamically generated web interface.
- CustomWebConfig `Custom Web Config`: This program shows how to use NetBurner's config system combined with a more complicated web interface featuring BootStrap and jQuery.

10.3 Custom Branding

NetBurner is an OEM manufacturer and our products can be customized to present your custom brand to your customers. Some customizations are very simple such as changing the logo on the configuration screens, to more complex such as having your own cloud server instead of using discover.netburner.com. Some suggestions for customization and branding are:

- Replace the company logo on the configuration server page. Please refer to the CustomWebConfigSystem example.
- Create your own configuration interface as part of your application. Examples are located in the `\nburn\examples\web` folder. The configuration server is not required to change system settings. An application has full access to display, change and save any configuration setting through a web interface, or within the application code itself.
- Order your NetBurner device without a NetBurner label. We also provide label templates so that you can change just the logo and part number.
- Customize the NetBurner utilities, such as `localdiscover` or `find`. Source code for these utilities are located in `\nburn\pctools`.

- Create your own cloud server in place of `discover.netburner.com`. This is very advanced.

10.4 IPv4/IPv6 Dual Stack Guide

Releases prior to 2.8.0 use a type of IPADDR that was a 32-bit number. The most significant change for 2.8.0 and beyond is that the IPADDR type is now an object that can hold IPv4 or IPv6 address information. This document will provide more information on this issue in the IPADDR class section.

10.4.1 Obtaining IPv6 addresses

IPv6 addressing is quite a bit different than IPv4, in which there is typically one IP address per network interface which is assigned by DHCP or set to a static address. In order for an interface to have more than one IP address it can use the Multihome feature. For example, a host's IPv4 AutoIP address.

10.4.1.1 Link-Local

When using IPv6, all interfaces have multiple IP addresses. For starters there is the 'link-local' address (fe80:↩ :/64). This is an address that is valid only for the local link that the device is connected to, and is used to negotiate the other addresses for the interface. While it can be used for any communication on the link, it is not a routable address. The link-local address is derived from the MAC address of the interface. As a result it is fixed for a given interface and determined without having to be explicitly set. For example, a device with the mac 00:03:f4:01:23:47 would have the link-local address fe80::203:f4ff:fe01:2347.

10.4.1.2 Router Auto-Configuration

In addition to the link-local address, IPv6 routers can tell the device to auto-configure based on the router's assigned prefix. To configure the end address, the device takes a number of bits of the router's prefix (usually 64), and combines this with its auto-configure address (the calculated portion of link-local) to obtain the prefix address. In our previous case, let's assume a router advertised a prefix of 2001:db8:1:2::. In this case, the full prefix address would be 2001:db8:1:2:203:f4ff:fe01:2347. This applies for all routers on the link that advertise auto-configuration. It's entirely possible (and reasonable in large corporate networks!) for an interface to have multiple prefix addresses from multiple routers.

10.4.1.3 DHCPv6

IPv6 addresses can also be assigned by a DHCPv6 server. Like DHCP for IPv4, this is used for assigning addresses to devices, along with passing other information like DNS and NTP servers. It is also possible for routers to notify devices that there is a DHCP server present on the network. In the case where such notification occurs, the Net↩ Burner will automatically configure to obtain the relevant information. There are two types of DHCP servers in IPv6 networks: Information-only and full Stateful Configuration. Information-only servers exist solely to provide additional information about the network, but do not assign IP addresses. These will allow for the auto-configuration of DNS, NTP servers, or other desired information. Full Stateful configuration additionally assigns addresses for use on the device's link. It is possible to manually start the DHCP client as well, for use on networks without a router.

10.4.1.4 Static IP Addresses

Finally, if all of these methods are insufficient, it is possible to manually assign and address to the device's interface as well.

10.4.2 IPADDR Class

The IPADDR type in NetBurner tool releases prior to 2.8.0 was a 32-bit unsigned integer. In order to be able to handle IPv6 addresses with the least impact to existing applications, it is now a C++ object. You will not need to

know C++ in order to use these objects. Most function calls will not have to change, and the system will do the appropriate action for both IPv4 and IPv6 addresses. One major change for users that have created custom storage structures that store IPADDRs in UserParams or a file on a file system is that when compiled for IPv6, the IPADDR is now 128 bits instead of 32. This means that unless plans are made for this, reading back stored IPADDRs will cause a problem for the application configuration in previously deployed devices that are updated. One way to address this issue is to use something like a 32-bit storage "key". Whenever reading the configuration structure check that the key in the application matches the key stored in memory. If different, then you must initialize the values. Any time you make a change to the stored information, such as adding a variable, increment the key.
The most significant difference is that while an unsigned 32-bit integer can be checked for a value of 0, an object cannot. For example, many applications check for a static IP address setting with `if (EthernetIP == 0)`. If EthernetIP is now an `IPADDR` object, the test should now be: `if (EthernetIP.IsNull())`.
For function parameters that were previously set to 0 to represent a null value, you can use `IPADDR::NullIP()`. For example, in previous releases a 0 was used for parameter 3, the DNS Server IP address, to tell the system to use the current runtime system DNS IP address value. This will generate an error in the new tools release since the 3rd parameter must be a typed object rather than an integer value. You may also use the `INADDR_ANY`, which is defined as `IPADDR::NullIP()` in the system files.

```
DNSResult = GetHostByName( serverName, &serverIp, 0, TICKS_PER_SECOND * 20
);
```

Becomes

```
DNSResult = GetHostByName( serverName, &serverIp, IPADDR::NullIP(), TICKS_↩
PER_SECOND * 20 );
```

10.4.3 IPADDR Member Functions

C programmers are used to creating structures that contain variables. A very simplistic way to look at a C++ object is that it is a structure that contains both variables and function declarations. These functions are called member functions. They are called using a dot notation instead of calling the function by itself. For example, in prior releases to print an IP address to the debug serial port (stdout) you might use the function: `ShowIP(EthernetIP)`. In the new release this becomes: `EthernetIP.printf()`. To print to a file descriptor such as a TCP socket, you can use: `EthernetIP.fdprintf(fd)`.

Some common functions are described below in their simplest form. Many of the functions have optional parameters, and experienced C++ programmers may want to know more detailed information, such as which are static and default parameters. Please refer to the NetBurner API documents for more information: IPADDR6

10.4.3.1 Setting IP Address Values with Member Functions

Member functions are called using a "." after the object instance name. For example:

```
IPADDR myIpObject;
myIpObject.SetNull();   // set the IP address to null
```

10.4.3.2 Member functions to set an IP address:

- `void SetNull()` Set the IP address of an existing IPADDR object to null
- `void SetFromAscii(const char *pStr)`, Set the IP address to the value of the ASCII representation in the string pointed to by pStr
- `void SetFromIP4(IPADDR4 ip)`, Set the IP address of an IPADDR4 object from an existing IPADDR4 object

10.4.3.3 Setting IP Address Values with Static Functions

A "static" function is one that is not attached to any particular instance of an object. They are invoked with a double colon "::". For example, IPADDR::NullIP() is used to represent an object instance with an IP address of null.

- `static IPADDR6 NullIP()`, Return an IPADDR6 object whose IP address is NULL. For example, IPADDR::NullIP();
- `IPADDR AsciiToIp(const char * pStr)`, Sets the IP address to the value represented by the ASCII string pointed to by pStr. For example, `IPADDR myIpAddr = IPADDR::AsciiToIp("192.168.1.1");`

10.4.3.4 Reading and Checking IP Address Values

- `bool IsNull()`, Returns true if the IP address is null
- `bool IsLoopBack()` Returns true if the IP address is the loopback address
- `bool IsMultiCast()` Returns true if the IP address is a multicast address
- `bool IsLinkLocal()` Returns true if the IP address is the IPv6 link local address. A link local address is of the format
  `FE80::/10`
  .
- `bool IsEmbeddedIPV4()` The same storage space is used to represent an IPv4 or IPv6 address. The function returns true if the address is
  `::FFFF:xx.xx.xx.xx`
  , where the x's represent the IPv4 address.
- `MACADR McastMac()` Returns the MAC address for multicasts at this IP address, or null if not a multicast address.

10.4.3.5 Output an IP Address

- `void printf()` Print the IP address value to stdout. The default stdout is the debug serial port
- `void fdprintf(int fd)` Print the IP address value to the specified file descriptor
- `int sprintf(char *pStr, int maxl)` Sends output to the string pointed to by pStr. maxL = Maximum number of characters to write. Returns the number of characters written.

10.4.4 Domain Name Service (DNS)

The function signature for DNS changes slightly between IPv4 and IPv6. However, if you use the default parameters, your code can be identical in both cases.

- `GetHostByName4()` IPv4 only and always available
- `GetHostByName6()` Available when IPV6 is enabled
- `GetHostByName()` Defined to GetHostByName4() in IPv4 mode and GetHostByName6() in IPv6 mode

The following code example will work in both cases:
`GetHostByName("thename.com", &resultIP, IPADDR::NullIP(), 10 * TICKS_PER_SECOND);`
This will put the IP address of "thename.com" in the resultIP. If you need DNS records other than just the name you can add additional type parameters. The types are:

- DNS_A, IPv4 address record
- DNS_CNAME, Canonical name record
- DNS_MB, Mailbox resource record
- DNS_MG, Mail group resource record
- DNS_MX, Mail exchange record
- DNS_AAAA IPv6 address record

When using DNS in IPv6 you need to decide which address to choose if the server has both an IPv6 and IPv4 address. The function GetHostByName6() has two default value types:

```
int GetHostByName6( const char *name,            // Name to resolve
                    IPADDR *pResultIP,           // Stores resultant IP
                    const IPADDR &dnsServerIP,   // DNS server IP address to use, or IPADDR::NullIP() for
      system
                    WORD timeout,                // Timeout in system time ticks
                    WORD TYPE1 = DNS_A,          // Defaults to IPv4 address
                    WORD TYPE2 =DNS_AAAA );      // Defaults to IPv6 address
```

The function will first try to get the TYPE1 value. If that fails, it will try to get the TYPE2 value. In the default case as defined in the header file it tries to get an IPv4 address, and if that fails it gets an IPv6. If you want to reverse this priority and try to get the IPV6 first then add the additional types in the function call:

```
'GetHostByName("thename.com", &resultIP, IPADDR::NullIP(),  10 * TICKS_PER_SECOND, DNS_AAAA, DNS_A);'
```

10.4.5 Note for OSX Users

At the time of this writing, BSD derived network stacks (including Apple's OS X) are not able to navigate to link-local addresses in a browser. You must use something refer to as a scope ID.

10.5 EFFS Programming Guide

This guide covers the NetBurner implementation of the HCC Embedded Flash File System (EFFS) and NetBurner flash card hardware interface. The EFFS is just one part of the NetBurner suite of tools and software. You may need to reference other documents for configuration and hardware specifics for your particular NetBurner platform. Topics Covered:

- An overview of examples, supported platforms and debugging information
- Overview of the flash card EFFS-FAT File System Operation
- Overview of the on-chip EFFS-STD File System Operation
- Hardware interface design for external flash cards

10.5.1 Example Programs

A list of examples is provided in the Example Applications section of this document.

10.5.2 Reference Manuals for the EFFS API

The function call reference manuals for the EFFS-FAT and EFFS-STD file systems are located in the `\nburn\docs\NetBurner\EFFS` folder.

10.5.3 Supported Hardware Platforms

All platforms support the on-board flash chip EFFS-STD file system. The implementation of the flash card EFFS-FAT file system is as described below. All devices can implement an external flash card design, while some platforms also have a flash card socket located on the device itself. There are two types of external flash card implementations:

- SPI: operates in native SPI mode with a single data channel
- SDIO: operated with four data channels

Platform	External	On-board
MODM7AE70	SPI	N/A
MOD54415	SPI	SPI
MOD54417	SPI	SPI
SB800EX	SDIO	
NANO54415	SPI	SDIO

The flash card sockets on the MOD-DEV-70CR and NANO development boards support SPI mode.

Topic Links:

- Flash Card Hardware Interface Design
- EFFS-FAT File System Operation
- EFFS-STD File System with On-chip Flash

10.5.4 Debug Port

Throughout this guide, we will refer to the "debug port". The debug port is one of the RS-232 ports that can be used to interact with your NetBurner device in the example programs. By default `stdout`, `stdin` and `stderr` are mapped to the debug port, so when you use functions like `printf()`, `scanf()`, `gets()`, etc. they read and write to the debug port. All of this is configurable. You can also disable the debug port and use the port as a general purpose UART, or you can reassign the stdio file descriptors to use other serial or network interfaces.

10.5.5 Flash Card Hardware Interface Design

10.5.5.1 Flash Card Hardware Interface Design

10.5.5.1.1 MMC/SD Hardware Interface

10.5.5.1.1.1 Schematic Representation The SD/MMC flash card interface is installed on NetBurner Network Development Kit (NNDK) development board. The schematic representation is shown below:

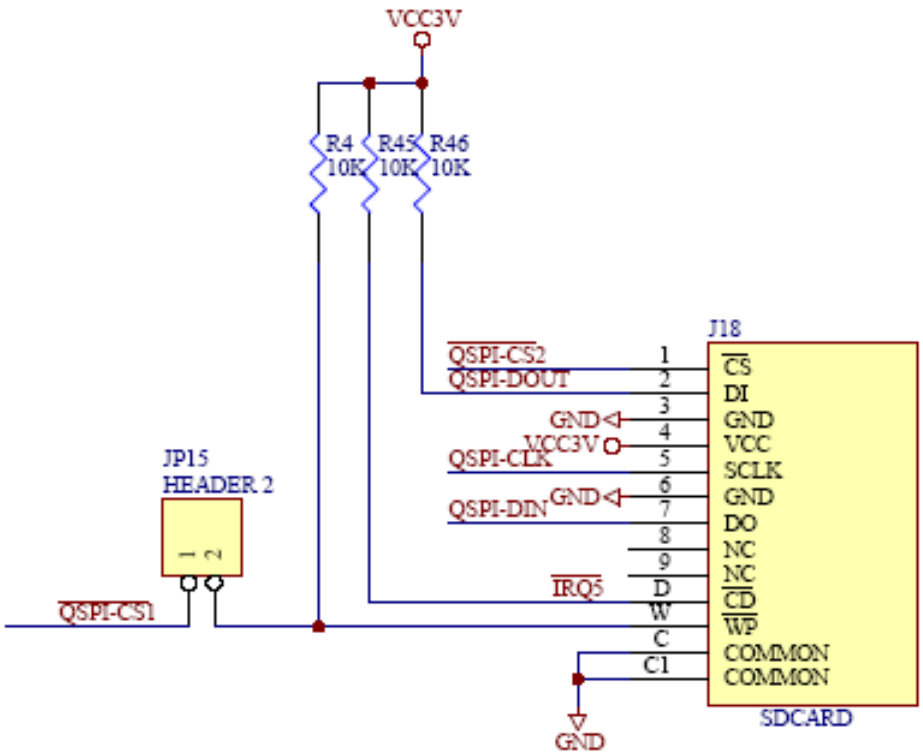


Figure 10.3 Development Board Schematic

SD/MMC Pin	SD/MMC Signal Name	Modxxxx Signal Connection
1	/CS, Chip Select	J2-35 - with 10k pull-up*
2	DI, Data In	J2-28 - QSPI DOUT
3	GND	Ground
4	VCC	VCC 3.3V
5	CLK	J2-J5 - QSPI CLK
6	GND	Ground
7	DO, Data Out	J2-J7 - QSPI DIN
8	NC, No Connection	No Connection
9	NC, No Connection	No Connection
D	CD, Card Detect	J2-47 - with 10k pull-up*
W	WP, Write Protect	J2-40 - with 10k pull-up*
Com	Common	Ground
Com	Common	Ground

10.5.5.1.1.2 Signal Description These signals can be any available GPIO pins on the NetBurner module, with a corresponding software change in `mmc_m7.cpp` for the MODM7AE70 or `mmc_mcf.cpp` for the other platforms to specify the signal name.

10.5.5.1.1.3 SD/MMC Connector Part Number The SD/MMC flash card connector used on the NetBurner development board is available from Mouser Electronics, part number 688-SCDA1A0901.

10.5.5.1.1.4 SD/MMC Card Compatibility The SD/MMC must support native SPI mode transfers, which is common for most SD/MMC cards. Up to 2GB is supported for standard SD/MMC and up to 32GB is supported for SDHC.

10.5.5.1.1.5 Exclusive Use of the SPI The EFFS normally requires exclusive use of the SPI interface. The standard software and drivers operate in this mode. There are also performance reasons for exclusive use. While it may be possible to share the SPI with other peripherals, it is not supported in the development tool suite.

10.5.5.1.1.6 SD/MMC Interface Control Pins You can modify the default pin selections for the interface control signals in `\nburn\platform\<platform>\mmc_m7.cpp` for the MODM7AE70, or `nburn\platform\<platform>\mmc_mcf.cpp` for the other platforms. The relevant functions in this file are lsited below. Note that the values in these functions must be used on the NetBurner development board since the board is wired in this configuration.

```
void MMC_BaseInit(int CurrentDrive)  // Configure pins to be used for SD/MMC interface control
int get_cd(int currentDrive)         // Get the Card Detect state
int get_wp( void )                   // Get the Write Protect state
void spi_cs_lo( void )               // Set SPI chip select low
void spi_cs_hi( void )               // Set SPI chip select high
```

10.5.5.1.1.7 Interrupt Driven SPI For the ColdFire based platforms (MOD5441X, SB800EX, and NANO54415), you can enable the SD/MMC SPI interface to work with the NetBurner interrupt driven SPI driver instead of the default polling mode driver. This will result in slightly lower SD/MMC performance but overall higher system performance. This is useful if you find that your network performance or user tasks run poorly during file system accesses. A modification must be made in `nburn\platform\<platform>\mmc_mcf.cpp`, followed by a rebuild of the platform system directory. Near to top of this file, the following line should be uncommented:

```
define SD_IRQ_SPI
```

10.5.6 EFFS-FAT File System Operation

10.5.6.1 EFFS-FAT File System Operation

Typical file system operation will involve mounting a drive, opening and closing files, and reading and writing files. The following is a list of the most common function calls used to perform these operations. For a complete list of functions refer to the EFFS Programming Guide.

10.5.6.1.1 Common EFFS FAT Function Calls Create/delete working directory for current task priority:

```
int f_enterFS(void)
void f_releaseFS(void)
```

Mount/dismount a flash card:

```
int f_mountfat(MMC_DRV_NUM, mmc_initfunc, F_MMC_DRIVE0)
int f_delvolume(int drivenum)
```

Open/Close a file

```
F_FILE *f_open(const char *filename, const char *mode)
int f_close(F_FILE *filehandle)
```

Read, write, and related functions:

```
int f_getfreespace(int drivenum, F_SPACE *pspace)
long f_write(const void *buf, long size,long size_st, F_FILE *filehandle)
long f_read( void *buf, long size,long size_st, F_FILE *filehandle)
long f_seek(F_FILE *filehandle,long offset, long whence)
int f_eof(F_FILE *filehandle)
int f_rewind(F_FILE *filehandle)
int f_delete(const char *filename)
```

Directory functions:

```
int f_findfirst(const char *filename, F_FIND *find)
int f_findnext(F_FIND *find)
int f_chdir(const char *dirname)
int f_mkdir(const char *dirname)
```

File time functions

```
int f_settimedate(const char *filename, unsigned short ctime, unsigned short cdate)
int f_gettimedate(const char *filename, unsigned short *pctime, unsigned short *pcdate)
```

10.5.6.1.2 File Time and Date Stamps The EFFS supports file time and date stamps. There are a number of ways to obtain the current world time for an embedded system, including a Network Time Server (NTP), Real-time clock (RTC), and setting it manually. For simplicity the first two examples concentrate on file system calls. The third example includes methods to set the time and date through all the aforementioned methods. If no time or date is set, the file time stamp will be January 1, 1980.

10.5.6.1.3 File System Utils All of the NetBurner EFFS examples include a helpful utility file called `File↩SystemUtils.cpp`. This file provides an easy use interface for initializing, getting status, testing, reading and writing to a CF or SD/MMC card. To select between the types of cards, edit the header file `cardtype.h`. This file also demonstrates many of the commonly used EFFS function calls. These utility files are located in `<nburn_install>\examples\_common\EFFS\FAT` or `<nburn_install>\examples\↩_common\EFFS\STD`, depending on if you want to use an SD card (FAT), or the on chip file system (STD). The examples for EFFS that take advantage of these utility files can be found in `<nburn_↩install>\examples\EFFS\FAT` or `<nburn_install>\examples\EFFS\STD`, again, depending on which file system you wish to use.

10.5.6.1.4 EFFS FAT Example Programs

10.5.6.1.4.1 EFFS FAT Example applications for using the file system as simple storage, with FTP, HTTP, HTTP with Variable tags, multiple tasks, RAM drive, and application updates are located at↩ : `\nburn\examples\EFFS\Fat`

10.5.6.1.4.2 EFFS FAT Used In Security Applications Example applications using the file system in security applications are:

- HttpsDualCert to demonstrate how to have both a permanent compiled-in certificate and key, as well as one that can be loaded from an SD/MMD flash card using the EFFS FAT file system for a web server.
- SSL_pop3 to demonstrate how to implement retrieving email from a server that requires SSL, and then saves it to an SD/MMD flash card using the EFFS FAT file system.
- SendMailAttach demonstrates how to attach files from the EFFS FAT file system to an email and send it securely using SSL/TLS encryption.

10.5.6.1.5 Project Settings To create a project that uses the EFFS-FAT file system, you will need to complete a few extra steps in addition to your normal project setup.
For NBEclipse users:

- Add the EFFS-FAT library, libFatFile.a, to the linker path. This is done by right clicking the NBEclipse project in the "Project Explorer" -> C/C++ Build -> Settings -> GNU C/C++ Linker -> Libraries -> Add "FatFile" to the list of "Libraries (-l)".
- Import the source files found here into your project: `<nburn_install>examples\_common\EFFS\FAT`

For projects that use the command line tools, add the following lines to your makefile:

```
XTRALIB = $(NNDK_ROOT)/platform/$(PLATFORM)/original/lib/libFatFFile.a
include $(NNDK_ROOT)/examples/_common/EFFS/FAT/common.mak
```

10.5.7 EFFS-STD File System with On-chip Flash

10.5.7.1 Using the EFFS-STD File System with On-chip Flash

10.5.7.1.1 EFFS-FAT vs. EFFS-STD The previous sections of this guide have focused on the EFFS-FAT file system, which is a FAT32 file system used for external flash cards. The EFFS-STD file system uses the on-chip flash memory of your NetBurner device. It has a different set of libraries and functions than the EFFS-FAT file system. In most cases the differences between the EFFS-STD and EFFS-FAT are simply a change in functioncall prefix from `f_` to `fs_`. For example, the EFFS-FAT function call `f_open()` is `fs_open()` for EFFS-STD. A list of the documented functions can be found at EFFS-STD Flash File System.

10.5.7.1.2 Overview When using the on-chip flash you need to be aware that the flash will be shared between your application and the file system. You will need to specify the amount of space to be used by the file system, while making sure you leave enough room for your application. The amount of flash used by your application is displayed each time you compile. You want to use the compressed number, not the uncompressed. You certainly want to leave enough additional space so your application can grow.
The flash memory chip will be divided into sectors, typically 4k or 64k bytes in size. The EFFS-STD file system requires that you allocate a number of these sectors to be used by the file system. Please review the data sheet for the flash memory used on your NetBurner device so you are familiar with the architecture. The configuration settings are dependent on the specific flash chip you are using.

10.5.7.1.3 Examples

10.5.7.1.3.1 EFFS STD Examples The HTTP example located at `\nburn\examples\EFFS\Std` demonstrates HTTP access to the STD file system. Web pages can be served from the application itself, or overridden if the same file name exists in the file system.

10.5.7.1.3.2 EFFS STD With NetBurner's Security Library Example are located at `\nburn\examples\SSL`
HttpsUploadCert - This program will demonstrate how to upload certificates and keys to support SSL/TLS web page access and store them in the EFFS-STD file system.
SslVerifyPeerEffs - This program will demonstrate how to upload CA Lists to use in support of verify peer, and store them in the EFFS-STD file system.

10.5.7.1.4 Project Settings To create a project that uses the EFFS-STD file system, you will need to complete a few extra steps in addition to your normal project setup.
For NBEclipse users:

- Modify the `COMPCODEFLAGS` to match the new memory space of the application. The remaining space will be used for the file system. The format of the `COMPCODEFLAGS` setting is: `COMPCODEFLAGS <start address> <end address>`.
- Add the EFFS-STD library StdFFile.a to the linker path

- Import the source files found here into your project <nburn_install>\examples_common\EFFS\STD

For projects that use the command line tools:

- Add the following lines to your makefile:

```
XTRALIB = $(NNDK_ROOT)/platform/$(PLATFORM)/original/lib/libStdFFile.a
include $(NNDK_ROOT)/examples/_common/EFFS/STD/common.mak
```

10.5.7.1.5 Flash Memory Addresses The flash memory on your NetBurner device is used for the Boot Monitor, System Parameter Storage, User Parameter Storage, Application, and now the EFFS-STD file system. A table of memory sizes for NetBurner platforms at the time of this writing is shown below. The example column illustrates one possible configuration. You can modify the parameters to suit your requirements. The COMPCODE flag starting address specifies the starting memory location of your application. The end address specifies the end location of the application. You should not modify the starting address! The Boot Monitor, Configuration Record and User Parameters occupy the space between the start of flash memory address and the start of the application memory address. You will only need to modify the end address to represent the amount of memory allocated for the flash file system.

Platform	Total Size in Bytes	Start Address	End Address	Configuration Definition Example
MODM7AE70	2MB	0x00400000	0x005FFFFF	**Example for 512K flash file system** Application must begin at 0x0040600 COMPCODEFLAGS = 0x00406004 0x005A0000 `#define FLASH_SIZE ( 2*1024*1024 ) // Total flash size #define FS_SIZE ( 256*512 ) // Size of filesystem`

10.5.7.1.5.1 Configuration File for SAME70Q21 Please refer to the SAME70Q21.h file located at \nburn\examples_common\EFFS\STD\src\flashChip

10.6 File Descriptors

The NetBurner environment integrates the RTOS, TCP/IP stack, and other peripherals with a file I/O system based on file descriptors. A file descriptor can be described as a handle to a network socket, serial port, system peripheral, or any other object that can be read or written to. Most of the API functions pass a file descriptor as a parameter to such an object
By default there are a maximum of 255 file descriptors:

- 0 – 2 for stdin, stdout and stderr

- 3 – 4 for the first two UART serial ports, 0 and 1.
- 5 – 128 for TCP (32 in total)
- 129 – 250 for expansion (additional UARTs, TCP sockets, or custom)

The expansion file descriptor positions can be used for many things, including additional serial ports, such as an external UART, TCP ports, or even buffers.

10.6.1 Creating Custom I/O Drivers Using File Descriptors

The header file in \nburn\include\iointernal.h defines the programming interface functions to create a custom file descriptor.

- SetDataAvail()
- ClrDataAvail()
- SetWriteAvail()
- ClrWriteAvail()
- SetHaveError()
- ClrHaveError()

```
struct IoExpandStruct
{
    int ( * read )  ( int fd, char *buf, int nbytes );
    int ( * write ) ( int fd, const char *buf, int nbytes );
    int ( * close ) ( int fd );
    void *extra;
}

int  GetExtraFD( void *extra_data, struct IoExpandStruct *pFuncs );
void *GetExtraData( int fd );
void FreeExtraFd( int fd );
```

The TCP state call back, fd = socket has new data, fd $<$ 0 means error `typedef void ( tcp_read_notify_handler )( int tcp_fd );`
When data comes in or the TCP connection enters an error state, register a callback to handle the event `void RegisterTCPReadNotify( int tcp_fd, tcp_read_notify_handler *newhandler );`
The Set and Clr functions are used to update the state of your fd device for file I/O functions such as `select()`, `read()` and `write()`. The `IoExpandStruct` is used to declare function pointers for the `read()`, `write()` and `close()` functions that will be implemented in your application, as well as an "extra" void pointer that you can use for whatever you wish.
`GetExtraFD()` is the function that will return a fd for the object passed as the `IoExpandStruct`. The extra_data void pointer is optional, and can be used to pass data into your fd. You can read the extra_data value at any time with the `GetExtraData()` function. `FreeExtraFd()` will release the fd back to the pool of available fds. The `tcp_read_notify_handler` and `RegisterTCPReadNotify()` are callback functions. These functions will get called by the system if you define them for the corresponding TCP event.

10.6.2 Using File Descriptors to Pend on Multiple Events

Once you have created a file descriptor you can use the `select()` function call just as you would for network or serial file descriptors. In fact, you can pend on a mixture of them all at the same time.

10.7 HTML Processing

10.7.1 Introduction

The Web Server will process all content in a project's "html" folder. If you are using any of the html examples or the application wizard, the html folder will have been created for you. If adding html and web server functionality to a

project without a html folder, simply create one, and add the corresponding StartHttp() or StartHttps() function call to your application to start web server services.
You can add whatever content you wish to the html folder: HTML, CSS, java script, java, images, etc. The NetBurner tools will automatically compile and link them into the application image that you download into your NetBurner device. You can use whatever web development tools you wish to create content, or write content with a text editor. The content in the html folder can be static pages or take advantage of dynamic content using function callback and variable tags. Static content can also be delivered from the onboard EFFS-STD file system and the external EFFS-FAT file system used by flash cards.

Note

When using NBEclipse, you can view web content source by right-clicking on the file and selecting Open With -> Text Editor

The function `StartHttp()` with no parameters will default to port 80. To use a different port number add a parameter, such as `StartHttp(2000)`. Similarly, `StartHttps()` will default to port 443. The function can have up to 2 parameters, one for HTTPS and one for HTTP. `StartHttps(527)` changes the HTTPS port number. `StartHttps(442, 80)` enables both HTTPS and HTTP on the standard ports.

10.7.2 Dynamic Content

Dynamic content is generated at run-time on your NetBurner device. For example, you can a web page for an instrument in which the content changes to match the current state of the instrument, such as gauges, light indicators, and switches. Or you can provide a graph showing data by creating an image or using any of the advanced tool such as Bootstrap. There are many examples of this in the `\nburn\examples\web` folder.
Dynamic content can be created in a number of ways:

- Function callback tags in HTML code to call a C/C++ function in real time as the web page is being rendered, such as `CPPCALL`
- Variable tags in HTML code to insert application variable values in real time as the web page is being rendered, such as `VARIABLE`
- Languages such as Javascript that runs in the browser
- A HTTP GET request handler that will be called when a web page is requested from the web browser
- A HTTP POST handler that will be called when a web page FORM post data to the web browser
- Web Sockets for two-way communication
- Web content can be delivered from multiple sources: content compiled into the application, generated by application code, internal file system (EFFS-STD in on-chip memory), or an external file system (EFFS-FAT) such as microSD flash cards.

10.7.3 HTML Tags

You can think of web server operation as receiving a request from a web browser for a file, and responding by sending that file as a data stream back to the browser. One method to create dynamic content is to embed a "tag" in the HTML source code file so that when the web server encounters it while streaming the content to the web browser, it takes some type of action in real-time. The NetBurner web server uses a number of tags to accomplish real-time dynamic content.

Tag	Description
CPPCALL	Call a C++ function in the application without parameters
VARIABLE()	Call a C++ function with parameters
VARIABLE	Insert the value of an application variable

The tags are used inside a HTML comment so they do not interfere with standard HTML processing. For example, a function callback could be used to format and display client information from the web browser:
`Client Info: <!--CPPCALL webShowClientInfo -->`

Warning

The space characters in the HTML comment block are important. there should be no space between "!--" and CPPCALL, and there must be a space between the end of the function name and "-->": `<!--CPPCALL YourFunctionName -->`

10.7.3.1 Configuration System Tags

While the previously mentioned tags can be used to display and/or modify any type of system configuration or application variables, the CONFIG tags are another option and can greatly simplify the interface. The CONFIG tags are designed to operate on the variables in the Configuration Record, which holds system configuration variables as well as any application variables in the AppData leaf of the configuration tree structure.

Tag	Description
CONFIGVALUE	Display a configuration or application variable value.
CONFIGINPUT	Display a configuration or application variable value and create a HTML "<input>" field. The current variable value will be displayed.
CONFIGTABLE	Create a table of the object and all sub-objects

•

10.7.3.2 The CPPCALL Tag

When the web server encounters a CPPCALL tag, it executes the corresponding C++ function in the application code. The function is passed the file descriptor to the open TCP socket and a pointer to the URL of the web page. The function then has control and can do whatever is desired by the application: create web content, perform operations such as controlling hardware, change software options, etc.

Note

The FUNCTIONCALL tag is is a 2.x tools tag and deprecated. The difference is the CPPCALL does not require extern "C" declarations in application source code to prevent C++ name mangling.

The screen shot below shows the usage for CPPCALL, VARIABLE and VARIABLE(x, y) tags:

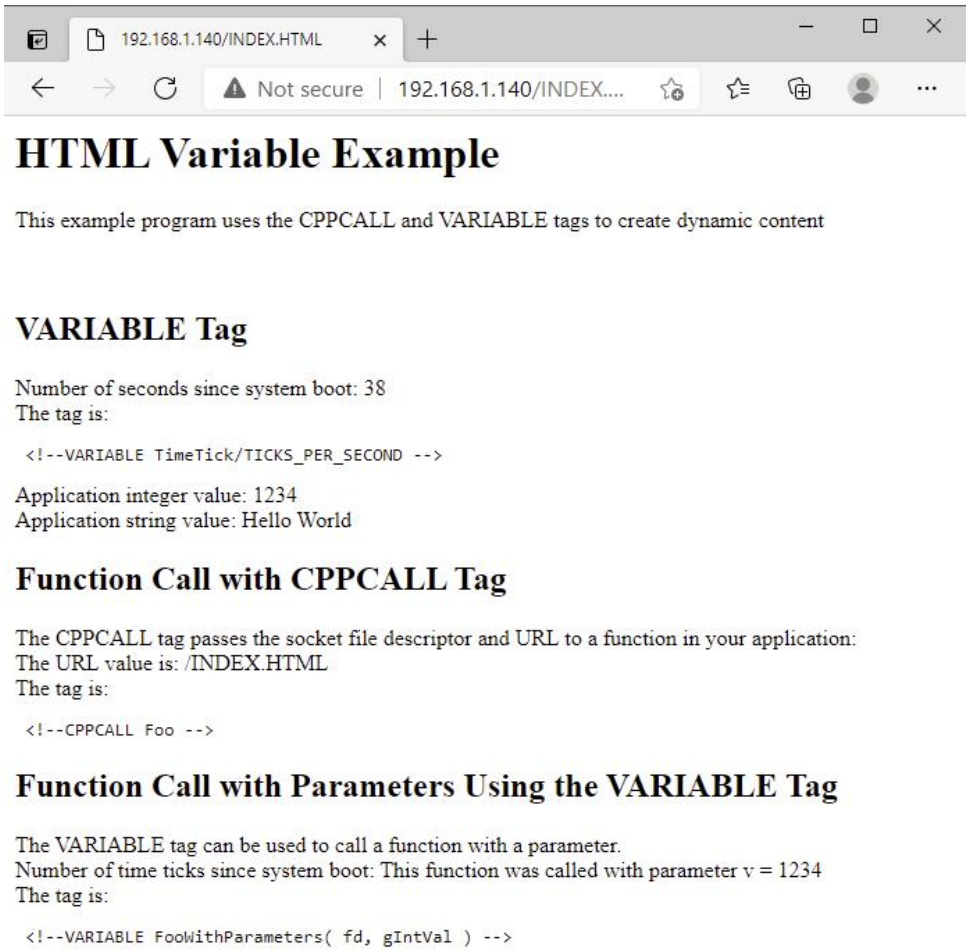


Figure 10.4 HTML Page with CPPCALL and VARIABLE Tags

The CPPCALL function signature is:

```
void YourFunctionName( int sock, PCSTR url )
{
   // add code here
}
```

For example, to display the IP address and port number of the client that made the web page request:
HTML Code:

```
<html>
<body>
    Client Address: <!--CPPCALL webShowClientInfo -->
</body>
</html>
```

The corresponding C++ function in the application:

```
void webShowClientInfo( int sock, PCSTR url )
{
    IPADDR localIP = GetSocketLocalAddr(socket);
    uint32_t remotePort = GetSocketRemotePort(socket);

    localIP.fdprint(socket);                        // Write IP address
    fdprintf(sock, ": %d<br>\r\n", remotePort);}  // Write port number
}
```

10.7.3.3 The VARIABLE Tag

The VARIABLE tag has two uses:

1. Simply inserting a variable or expression
2. Calling an application function with variable parameters (essentially a CPPCALL with function parameters)

To display a variable the format is: <!--VARIABLE <name> -->
Where <name> is the name of the application variable or an expression. For example, the system time tick variable, TimeTick can be displayed with <!--VARIABLE TimeTick -->
Or you can display the time in seconds with the equation: <!--VARIABLE TimeTick/TICKS_PER_SECOND -->
The VARIABLE tag is processed during the compilation of the application by parsing the text between "<!--VARIABLE" and the trailing " -->", then the NetBurner tools automatically convert it into a function call with a signature that contains a file descriptor and variable. For example, WriteHtmlVariable(fd, TimeTick/TICKS_PER_SECOND);
The following parameter types are available and defined in \nburn\include\htmlfiles.h:

```
void WriteHtmlVariable(int fd, char c);
void WriteHtmlVariable(int fd, int i);
void WriteHtmlVariable(int fd, short i);
void WriteHtmlVariable(int fd, long i);
void WriteHtmlVariable(int fd, uint8_t b);
void WriteHtmlVariable(int fd, uint16_t w);
void WriteHtmlVariable(int fd, unsigned long dw);
void WriteHtmlVariable(int fd, const char *);
void WriteHtmlVariable(int fd, MACADR ip);
```

10.7.3.4 The INCLUDE Tag and htmlvar.h Header File

The functions that are created from the VARIABLE tags are located in an auto-generated file named htmldata.cpp. Since these functions reference the variable names, there must be a way for the linker to resolve them. For example, to display TimeTick the application would need to include <nbrtos.h>, otherwise a linker error will occur.
Include files for the HTML code/files to enable linking can be handled two ways:

1. An include HTML tag, <INCLUDE headername.h> in the HTML code
2. Create a header file in the project with the name "htmlvar.h". The tools will automatically look for a file of this name if <INCLUDE> tags are not detected in the HTML source files.

Example of a HTML file using the INCLUDE tag with a user defined header file name:

```
<html>
<body>
<!--INCLUDE myIncludeFile.h -->
Value = <!--VARIABLE MyVar --><br>
</body>
</html>
```

Example of htmlvar.h header file that exposes two variables, the <nbrtos.h> header file, and a function that takes an integer parameter:

```
#ifndef HTMLVARS_H_
#define HTMLVARS_H_

#include <nbrtos.h>     // For access to TimeTick

extern int  gIntVal;
extern char gStrVal[];

const char *FooWithParameters(int fd, int v);

#endif /*HTMLVARS_H_*/
```

10.7.3.5 The VARIABLE() Tag With Parameters

If you need to specify a function callback but need to pass a parameter, the CPPCALL tag will not work because the function signature parameters are fixed as the socket file descriptor and URL. In this case we can use the VARIABLE tag to achieve the functionality of calling a function with a variable.
The include file (e.g. htmlvar.h) must specify the function definition in the format below. In this case we are passing an integer value 'v'. The first parameter must always be the socket file descriptor: `const char * myInt↩Function(int fd,int v);`
The HTML source code then uses the VARIABLE tag with the function definition. In this example we are passing the integer value of TimeTick. `<!--VARIABLE MyIntFunction(fd,TimeTick) -->`
When the application is compiled, the function definition will be created as: `WriteHtmlVariable( fd, My↩Function(fd,TimeTick) );`
This function returns an empty string, which will have no effect on the web page. An example of what a function might do is shown below:

```
const char *FooWithParameters(int fd, int v)
{
    fdprintf(fd, "This function was called with parameter v = %d\r\n", v);
    return "\0"; // Return a const char * here of zero length so it will
                    not print anything
}
```

10.7.3.6 Extending the VARIABLE Tag

The VARIABLE functionality can be extended to support user defined types, such as displaying a user defined structure or class. Let's say you have a structure you want to display on a web page called myStruct:

```
struct  my_struct {
   int      i;
   char     buf[80];
   uint32_t dVal;
} MY_STRUCT;

MY_STRUCT myStruct;
```

In your include file add the function definition: `void WriteHtmlVariable(int fd, MY_STRUCT my↩Struct);`
Now you can display it on the web page with the VARIABLE tag: `<!--VARIABLE myStruct -->`
Which will compile to: `WriteHtmlVariable( fd, myStruct );`
Note that you still have to write the implementation of the above function. The function below is the source code for the MAC address type already defined by the system:

```
void WriteHtmlVariable(int fd, MACADR ma)
{
   PBYTE lpb = ( PBYTE ) &ma;
   for (int I = 0; I < 5; i++)
   {
      fd_writehex( fd, lpb[i] );
      write(fd,":",1);
   }
   fd_writehex( fd, lpb[6] );
}
```

10.7.3.7 The CONFIGVALUE Tag

10.7.4 HTTP GET Request Handler

When you click on a link or enter an address in the URL field of a web browser, it sends a GET request to the web server. At its most basic level a GET request is the word "GET" and the requested file name. A GET handler is a mechanism by which an application can take control of the GET request instead of it being handled by the

normal web server processing. For example, if an application implemented a GET handler for a HTML page named "showstatus.html", the GET handler could create whatever type of status output it wanted instead of the web server delivering a file named "showstatus.html".
A GET handler is also useful for URL encoding. Two common methods for delivering data from a client (eg. web browser) to a web server are URL encoding and HTML forms using a POST submission. URL encoding involves sending data to the web server by adding it to the end of the URL. For example, an e-commerce application might store product information such as: http://www.store.com/orderform?type=order123. The data sent to the web server to be processed "type=order123" in the URL. Everything following the "?" character is ignored by the browser, so your application can store information there. Another advantage of this method is that the application can be stateful, meaning multiple users can access the same application and each user's session is maintained by the data encoding in the URL.
When a web browser requests something from a web server, such as an HTML page or image, it makes a GET request. The web server normally handles static web pages and dynamic web pages with the CPPCALL and VARIABLE tags, but your application can intercept the request and take control of the processing using a callback function object called CallBackFunctionPageHandler. When you declare the instance of the object you specify the name of the request to intercept and a pointer to the function in your application to process the request. For example, to take control of processing for a HTML page named setcookie.html:
To create a HTTP GET handler:

1. Create an application callback function to process the GET request
2. Create a CallBackFunctionPageHandler declaration to specify the URL mask, application callback function, and additional configuration parameters

A GET handler consists of a callback function to handle the actual processing, and a page handler declaration specify the conditions in which the callback function should be executed. The callback function sends an HTML header to identify the content, in this case it is HTML, determines if the system seconds timer is odd or even, then sends the appropriate HTML file as a response. Note that OddEven.html does not exist as a file. The GET request for OddEven.html is intercepted by the callback.
The callback has the function signature: 'int callbackGetOddEven(int sock, HTTP_Request &pHttpRequest)'
The page handler declaration has the signature:

```
CallBackFunctionPageHandler(const char *pUrl,
                            http_gethandlerfunc *pFunction,
                            HTTP_RequestTypes reqType = tGet,
                            int  accessGroup = 0,
                            bool beforeFiles = false)
```

```
/*---------------------------------------------------------------------------------------
 * Callback to intercept a GET request for URL OddEven.html. The purpose of this callback
 * example is to demonstrate how to add custom processing and send an existing HTML file.
 * The URL OddEven.html does not exist as a file. If the page is requested by a browser,
 * this callback function is executed, which does some processing and decides which web
 * page file to send to the browser based on the current system seconds timer value.
 *
 * - Sends a HTML header to the browser to specify the content that follows is HTML
 * - Reads the system seconds timer, and sends the file odd.html or even.html
 * - A return value of 1 tells the system the HTML request was handled by the callback
 *
 * Notes:
 * - HTTP_Request is a structure containing detailed information of the request
 *---------------------------------------------------------------------------------------*/
int callbackGetOddEven(int sock, HTTP_Request &pHttpRequest)
{
    // Send the web page header telling the browser we are sending HTML content
    SendHTMLHeader(sock);

    //----- Add custom processing here -----
    // Display message to serial terminal showing requested URL
    iprintf("Executing GET request callback function: callbackGetOddEven\r\n");
    iprintf("Request URL: %s, from: %I\r\n", pHttpRequest.pURL, pHttpRequest.client_IPaddr);

    if ( (Secs % 2) == 0)
        SendFileFragment("even.html", sock);
    else
        SendFileFragment("odd.html", sock);

    // Notify system we handled the GET request
    return 1;
```

```
}

/*
 * Declare the callback function. Parameters:
 *   OddEven.html       URL to intercept
 *   callbackGetOddEven Pointer to the callback function
 *   tGet               HTTP Request Type (ref HTTP_RequestTypes)
 *   0                  User Access Group, 0 = no password
 *   true               When to call (beforeFiles parameter):
 *                        true:  Always called for the specified URL
 *                        false: Only called if the if the requested URL cannot be satisfied elsewhere
 */
CallBackFunctionPageHandler getHandlerOddEven("OddEven.html", callbackGetOddEven, tGet, 0, true);
```

A key component is the `CallBackFunctionPageHandler` declaration. Lets look at each parameter:
`pURL:` is the URL name to match. In this case it is fully defined as "OddEven.html", but wildcards can also be used. Although not very useful in this case, it could have been "Odd∗", which would result in the callback being executed for anything beginning with "Odd".
`pFunction:` is a pointer to the function callback.
`tGet:` is the request type, in this case it is a GET, which has a request type of tGet.
`accessGroup:` is used for security/password access. Not the subject of this example, so a 0 for no access restriction is used.
`beforeFiles:` refers to the html content in the html folder of the project. If true, the callback will be executed without the web server first checking to see if a file by that name exists. If false, if a file by the pURL name is found, it will be sent to the client first.

10.7.5 HTTP Form POST Handler

You have probably encountered forms many times on the web, such as e-commerce and feedback forms. The format is typically some number of text fields, check boxes, radio buttons, combo boxes and a submit button. When you click on the submit button, the data from the form is sent to the web server as a HTTP POST. The web server then parses the data and takes appropriate action.
A form is declared in HTML by the `<form>` tag. User input is controlled by various `<input>` tags. For example, the web page below creates a form with a text field. NetBurner tags such as CPPCALL, VARIABLE and CONFIG tags can be used to populate input fields with their current value.
To create a HTTP POST form handler:

1. Create a HTML web page with a form
2. Create a corresponding application callback function to process the posted data
3. Create a HtmlPostVariableListCallback declaration to specify the matching callback mask and attach the application callback function

The example below creates a HTML page with four input fields, and a POST handler callback function to process them.

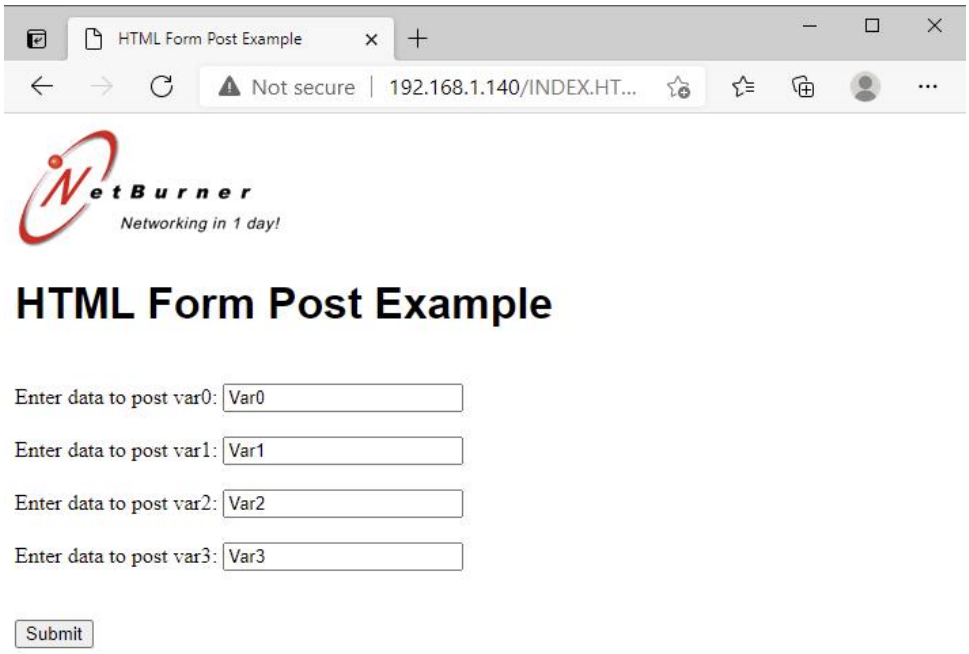


Figure 10.5 HTML form with four variables

HTML Code:

```
<html>
<title>HTML Post Example</title>
<body>
    <img src="images/netburner-logo.gif">
    <h1><font face=arial>HTML Form Post Example, Second Form</font></h1>
    <br>

    <form action="form2" method=post>
       Enter data to post var0: <input type="text" name="Var0" value="Var0"><br><br>
       Enter data to post var1: <input type="text" name="Var1" value="Var1"><br><br>
       Enter data to post var2: <input type="text" name="Var2" value="Var2"><br><br>
       Enter data to post var3: <input type="text" name="Var3" value="Var3"><br><br>

       <br><input type="submit" value="Submit"><br>
    </form>
</body>
</html>
```

Corresponding Application Source Code:

```
/*
 * Callback function for Form2. This function will be called once for each variable
 * in the form.
 */
void form2PostCallBack(int sock, PostEvents event, const char * pName, const char * pValue)
{
    // Received a call back with an event, check for event type
    switch (event)
    {
        // Called at the beginning of the post before any data is sent
        case eStartingPost:
            for (int i = 0; i < numVars; i++)
                strVar[i][0] = '\0';
            form2ParseError = false;
        break;

        // Called once for each variable in the form
        case eVariable:
            if (strcmp("Var0", pName) == 0)
            {
                strncpy(strVar[0], pValue, varLen - 1);
                iprintf("strVar0 set to: \"%s\"\r\n", strVar[0]);
            }
            else if (strcmp("Var1", pName) == 0)
```

```
            {
                strncpy(strVar[1], pValue, varLen - 1);
                iprintf("strVar1 set to: \"%s\"\r\n", strVar[1]);
            }
            else if (strcmp("Var2", pName) == 0)
            {
                strncpy(strVar[2], pValue, varLen - 1);
                iprintf("strVar2 set to: \"%s\"\r\n", strVar[2]);
            }
            else if (strcmp("Var3", pName) == 0)
            {
                strncpy(strVar[3], pValue, varLen - 1);
                iprintf("strVar3 set to: \"%s\"\r\n", strVar[3]);
            }
            else
            {
                iprintf("*** Error: Form variable name not found\r\n");
                form2ParseError = true;
            }

        break;

    //Called back with a file handle if the post had a file
    case eFile:
        break; //No file type here so we do nothing

    // Called back when the post is complete. You should send your response here.
    case eEndOfPost:
        {
                iprintf("Variable Summary:\r\n");
                for (int i = 0; i < numVars; i++)
                    iprintf("  Var%d = \"%s\" \r\n", i, strVar[i]);

                if (form2ParseError)
                    iprintf("*** Error: Form variable name not found\r\n");

            RedirectResponse(sock, "complete.html");
        }
        break;
    } //switch
}

// Create a global static post handling object that responds to the specified html page.
// A separate post handler can be created for each form in your application.
HtmlPostVariableListCallback postForm2("form2*", form2PostCallBack);
```

When a post is submitted the callback post handler will be executed a number times for the following events:

Event	Description
eStartingPost	Occurs once at the first event of a post
eVariable	Occurs once for each variable in the form
eFile	Occurs if a file is being posted
eEndOfPost	Occurs once at the end of post processing

In this example the following events will occur:

- eStartingPost is called first and enables the application to initialize the array.
- eVariable is called once for each variable (input field). In this example it will be called 4 times. The callback function must parse for the input field name on each call.
- eFile is called. Since we are not posting a file no action is taken.
- eEndOfPost is called last. In this example it is used to print a summary of the input field values

The HtmlPostVariableListCallback declaration is used to specify the URL name and mask, as well as a pointer to the post handler callback. Full URL names can be specified with and multiple callback functions, or a wildcard can be used and the post handler callback must parse the URL information to determine the correct action.

10.8 Interrupt Handling

10.8.1 Overview

An interrupt is a hardware or software request to interrupt the currently executing code and jump to a routine/code to process the event. After the interrupt processing is complete, the execution of non-interrupt code will resume. The code that processes an interrupt is commonly called an Interrupt Service Routine (ISR) or Interrupt Handler. Although it varies by platform, NetBurner devices commonly handle interrupts from sources such as timers, serial port, Ethernet ports, external hardware inputs, and software traps.
As mentioned, interrupt handling varies by processor hardware. In some cases, such as ColdFire, NetBurner provides macros to make the more involved ISR entry and exit much easier. ARM platforms tend to be much simpler, requiring only the Interrupt Service Routine function.

10.8.2 Recommended Interrupt Functions

The three recommended interrupt functions that work on all NetBuner platforms are listed below. There are also lower level functions for situations that require fine tuning. The PinIO object was created to make the configuration of a processor pin easier than having to configure all the required processor registers.
Configure a pin to handle an external interrupt by specifying the pin number as a PinIO object, the polarity (high/low), and the Interrupt Service Routine that will be called to process the interrupt:

```
bool SetPinIrq( PinIO pin, int polarity, PinIrq_t func);
```

Enable the interrupt:

```
void EnableIrq( PinIO pin );
```

Disable the interrupt:

```
void DisableIrq( PinIO pin );
```

10.8.3 Functions That Cannot Be Called in an ISR

Warning

> The code within an ISR can be called at any time, therefore certain RTOS and I/O functions cannot be called from within the ISR.

10.8.3.1 All nbrtos init and pend functions (all OsxxPendNoWait functions are okay):

- OSxxPend
- OSCritEnter
- OSChangePrio
- OSTaskDelete
- OSLock
- OSUnlock
- OSTaskCreate
- OSTimeDly

10.8.3.2 I/O functions:

- write
- writeall
- read
- printf
- fprintf
- fdprintf
- iprintf
- scanf
- gets
- puts

10.8.3.3 Memory management functions:

- malloc
- free
- new
- delete

10.8.4 MODM7AE70 and SBE70LC Platforms (SAME70)

Any GPIO pin on the SAME70 processor can be used as an interrupt source. Pin IRQs are inputs on General Purpose I/O (GPIO) ports. The Cortex M7 architecture is structured in a way that all pins on the same port share the same GPIO priority. In other words, there is one IRQ per port.

10.8.4.1 Configure the interrupt request (IRQ) priority for a PIO pin

Configure a port priority by specifying a PinIO object.

10.8.4.1.1 Synopsis

```
bool SetPortIrqPriority( PinIO pin, int priority);
```

10.8.4.1.2 Description The PIO controller on the SAME70 configures the IRQ priority by PIO ports. **Therefore, configuring the priority of an IRQ for a specific pin will change the IRQ priority for the entire PIO port**. SetPinIrq() will set the pin IRQ with the priority that is configured with SetPortIrqPriority(). Therefore, SetPortIrqPriority() should be called before SetPinIrq().

Name	Description
pin	A PinIO object that is to be configured as an IRQ Pin.
priority	Priority of the interrupt. A priority value of 1 being the highest priority IRQ, and 7 being the lowest.

Returns true if successfully if the PIO port's IRQ priority was successfully configured, otherwise false.

10.8.4.2 Configure the interrupt request (IRQ) priority for a PIO port

Configure a port priority by specifying a port number.

10.8.4.2.1 Synopsis

```
bool SetPortIrqPriority( int portNum, int priority);
```

10.8.4.2.2 Description The PIO controller on the SAME70 configures the IRQ priority by PIO ports. Therefore, configuring the priority of an IRQ for a specific pin will change the IRQ priority for the entire PIO port. SetPinIrq() will set the pin IRQ with the priority that is configured with SetPortIrqPriority(). Therefore, SetPortIrqPriority() should be called before SetPinIrq().

Name	Description
portNum	PIO port number. Where PIOA = 0, and PIOE = 4.
priority	Priority of the interrupt. A priority value of 1 being the highest priority IRQ, and 7 being the lowest.

Returns true if successfully if the PIO port's IRQ priority was successfully configured, otherwise false.

10.8.4.3 Set up an interrupt request (IRQ) pin

10.8.4.3.1 Synopsis

```
bool SetPinIrq( PinIO pin, int polarity, PinIrq_t func);
```

10.8.4.3.2 Description Set up a PinIO object pin with polarity and pointer to an ISR function.

Name	Description
pin	A PinIO object that is to be configured as an IRQ Pin.
polarity	Polarity and edge vs. level sensitivity configuration. 2: high level sensitive 1: positive edge sensitive 0: (either) edge sensitive -1: negative edge sensitive -2: low level sensitive
func	A function pointer of type PinIrq_t to register as the service handler (ISR) for the given PinIO.

Returns true if PinIO was successfully registered for IRQs, otherwise false.

10.8.4.4 Enable the interrupt for the given PinIO

10.8.4.4.1 Synopsis

```
void EnableIrq( PinIO pin );
```

10.8.4.4.2 Description Enable the interrupt for the given PinIO

Name	Description
pin	The PinIO object to enable interrupts for.

10.8.4.5 Disable the interrupt for the given PinIO

10.8.4.5.1 Synopsis

```
void DisableIrq( PinIO pin );
```

10.8.4.5.2 Description

Disable the interrupt for the given PinIO

Name	Description
pin	The PinIO object to disable interrupts for.

10.8.5 MOD54415, MOD54417, NANO54415 and SB800EX Platforms (MCF5441x)

The MCF5441x processor requires more setup and configuration than ARM platforms. To simplify implementation, the NetBurner platform implements the following functions:

- An INTERRUPT() Macro to provide the Interrupt Service Routine entry and exit.
- The SetIntC() function to configure the Interrupt Controller hardware.

Each are described in the following sections.

10.8.5.1 INTERRUPT() Macro

MCF5441x platforms use the INTERRUPT() macro as described below, which handles the ISR entry and exit coding. The header file to include is cfinter.h, located in: "\nburn\arch\coldfire\include\cfinter.h". ColdFire interrupts have values from 1 to 7, with 7 being the highest priority.

Note

Once you use the INTERRUPT macro to define the interrupt function, you will need to use the SetIntc(), set interrupt controller function, to set up your interrupt controller variables to point the processor's interrupt vector at that function. Please refer to the appropriate NXP hardware reference manual, located in "\nburn\docs\nxp" for your specific interrupt source number and controller number (some ColdFire's have more than one interrupt controller).

10.8.5.1.1 Header Files

```
#include <nbrtos.h>
#include <cfinter.h>
```

10.8.5.1.2 Synopsis

```
INTERRUPT( FunctionName, sr_value )
```

10.8.5.1.3 Description

The INTERRUPT macro sets up an interrupt function and the code block to save and restore the CPU registers. In addition, this macro tells the RTOS that an interrupt is occurring.

Note

Level 7 interrupts are unmaskable, which means they will interrupt any other processing, including the level 7 ISR itself. Therefore, any level 7 ISR used in an application must ensure it can complete its processing before another level 7 interrupt occurs.

Name	Description
FunctionName	The name of the Interrupt Service Routine, to be used with the SetIntC() function. This is the name of a C/C++ function in an application created to process the interrupt.

Name	Description
SR_Value	The ColdFire Status Register (SR) value that the processor will have during the interrupt. This is used as a mask to specify which, if any higher priority interrupts can interrupt the ISR. The mask bits define the current interrupt level. Interrupt requests are inhibited for all priority levels less than or equal to current level, except edge-sensitive level 7 requests, which cannot be masked.For example, if a level 5 interrupt is currently executing, specifying a value of 0x2500 will allow a IRQ level 6 or higher to interrupt.

10.8.5.1.4 Parameters

10.8.5.1.5 Status Register Values The status register is 32-bits. The mask bits are 24-26 and range from 000 to 111. The eight permitted SR register values are shown below. Bit 29 is the supervisor bit, which should always be set.

- 0x2000 - Allows all interrupts
- 0x2100 - Blocks all interrupts below level 2
- 0x2200 - Blocks all interrupts below level 3
- 0x2300 - Blocks all interrupts below level 4
- 0x2400 - Blocks all interrupts below level 5
- 0x2500 - Blocks all interrupts below level 6
- 0x2600 - Blocks all interrupts below level 7
- 0x2700 - Blocks all interrupts below level 7

10.8.5.2 SetIncC() function (Set Interrupt Controller)

MOD54415, MOD54417, NANO54415 and SB800EX SetIncC() functions are located in Hardware Abstraction Layer (HAL) cpp file for the particular platform. For example, for the MOD5441x it is located in "\nburn\platform\↩ MOD5441x\hal.cpp". The SetIntC() function must be declared as an extern in your application .cpp file, it is not available in a header file. For example:

```
void SetIntc(int intcnum, long func, int vector, int level)
```

Parameters:

Name	Description
intcnum	Interrupt Controller Number. For MCF5441x processors there are two: 0 and 1.
func	Pointer to ISR function name.
vector	Interrupt Vector Number. There are 256 exception vectors; the first 64 are defined for the core and the remaining 192 are device-specific peripheral interrupt vectors. See Chapter 17, “Interrupt Controller Modules (INTC)” for details on the device-specific interrupt sources, and table 17.2.9.1 Interrupt Sources. For example, the UART 0 vector number is 26, and the PIT2 timer is 15.
level	Interrupt level. Range is from 0 (disabled) to 7, with 7 being the highest level/priority. For example, a level 6 interrupt request can interrupt a level 1 Interrupt Service Routine.

10.8.5.3 Example, OneWire Interrupt Configuration

```
extern void SetIntc(int intc, long func, int vector, int level);

// Interrupt Service Routine
void OWMasterIsr()
{
    // ISR code...
}

// INTERRUPT macro to set vector ISR function and mask to level 6
```

```
INTERRUPT( ow_master_int_routine, 0x2600 )
{
    OWMasterIsr();
}


// Initialization call to use in UserMain() of application. The purpose of this example is only
// to demonstrate the use of SetIntC().
void OWInit()
{
    // Initialize 1-Wire module registers (MCF5441x Reference Manual Chapter 36.3: 1-Wire Memory
      Map/Register Definitions)
    sim1.ow.cr  = 0;
    sim1.ow.div = 0x7C; // Divides the peripheral bus clock to run at 1 MHz
    sim1.ow.cmd = 0;
    sim1.ow.ier = OW_IER_ERBF;

    // 63 is the interrupt source number for the 1-Wire module, and we're using IRQ level 3
    SetIntc(0, ( long ) &ow_master_int_routine, 63, 3 /* IRQ 3 */);

    ow_struct.state = OK;
}
```

10.8.6 SOMRT1061 Platform

10.9 JSON Lexer

10.9.1 Using the JSON Lexer Interface

Note

This application note applies to NNDK tools version 3.3.9 and later. The many additional features added will not build in earlier tools versions.

The Json Lexer interface is in json_lexer.h. The two object types that are of interest are:

1. ParsedJsonDataSet: represents a complete JSON object parsed and stored internally.
2. JsonRef: represents a positional reference of location inside a ParsedJsonDataSet. It can be used as a pointer to a variable or sub object inside a ParsedJsonDataSet object to make decoding the hierarchy much simpler and easier to understand. A JsonRef is not limited to just pointing at internal objects; it can point to any JSON parsed token/type.

The ParsedJsonDataSet has two decoding interfaces. The previous interface prior to 3.3.9 should be considered deprecated. If you have used this class in the past this code will still work, but we have added a new much clearer easier to understand interface based on the JsonRef object and the associated overloaded operators. This interface should be used for all decoding going forward.
Before we can practice decoding JSON, we need to get the JSON content into a ParsedJsonDataSet object. There are multiple methods to accomplish this:

1. Load from a text blob as demonstrated in the example: `\nburn\examples\JsonLexer\ParseTest`. Calling the function `ParsedJsonDataSet pjd((const char *)jdata, jlen, true);` will construct an object from plain text.
2. Load from an external web page as demonstrated in the examples:

- `\nburn\examples\JSON\GetJsonFromServer`

- \nburn\examples\JSON\PostAndGetJsonExternalServer
- \nburn\examples\WebClient\EarthQuake

1. Load from some type of external interface, web socket, serial port, etc.

- `ParsedJsonDataSet pjds; // declare the object`
- `pjds.ReadFrom(int fd); // read the content from a file descriptor`

Once you have a complete ParsedJsonDataSet, you can operate on it and decode the JSON content.
For example, we have a simple JSON structure:

```
{ "Anumber": 1234,
  "AString":"ImaString",
  "AnObject":{"Item1":1,
              "Item2":"TheItem2"
             },
"AnArray" : [1,2,3,4]
}
```

Using one of the three methods listed above will convert the JSON content into a ParsedJsonDataSet object named pjds. Internally, the ParsedJsonDataSet toekenizes the JSON it is loaded with and stores the results in minimized form in NetBurner poolBuffers. None of these internals really matter to the user.
To extract values from the parsed data set:

```
int i = pjds("ANumber");        // Read an integer

// Strings can be accessed as a NBString object or const char *.
// The const char * is only valid until the dataset is destroyed or cleaned up.
NBString s = pjds("AString");
const char * pStr = pjs("AString");

// JsonRef can be used as a pointer to reference interior variables and objects
// in the data set and access them in the same manor.
JsonRef jr = pjds("AnObject");
int i1 = jr("Item1");
NBString s2 = jr("Item2");

// Arrays can be indexed using the array operator
for(int i = 0; I < 4; i++)
   printf("Item[%d] = %d\n", i, pjds("AnArray")[i]);
```

In general, you can use ("name") to find an element and [i] to index in an array. You can then assign the result to the desired object or variable.
ParsedJsonDataSet types:

```
operator  bool ()    const {return PermissiveCurrentBool(); };
operator  float ()   const {return (float)CurrentNumber(); };
operator  double ()  const {return CurrentNumber(); };
operator  uint8_t()  const {return (uint8_t)CurrentNumber(); };
operator  int()      const {return (int)CurrentNumber(); };
operator  uint16_t() const {return (uint16_t)CurrentNumber(); };
operator  uint32_t() const {return (uint32_t)CurrentNumber(); };
operator  int8_t()   const {return (int8_t)CurrentNumber(); };
operator  int16_t()  const {return (int16_t)CurrentNumber(); };
operator  int32_t()  const {return (uint32_t)CurrentNumber(); };
operator  time_t()   const {return (time_t)CurrentNumber(); };
operator const char *() const {return CurrentString(); };
operator NBString()  const {return (NBString)CurrentString(); };
```

There are corresponding check functions as well in the form of: `bool IsValid();` to determine if the dereferenced ParsedJsonDataSet element or JsonRef point to something valid.
Continuing from the example above:

```
pjds("AnArray")[3].IsValid();   // returns true, a valid decode
pjs("AnArray")[4].IsValid();    // returns false as there are only array elements 0 - 3
```

Additonal Check Functions:

```
IsNumber()
IsObject()
IsString()
IsBool()
IsNull()
IsArray()
```

These functions are demonstrated very well in: `\nburn\examples\JsonLexer\ParseTest`. The example `\nburn\examples\webclient\Earthquake` uses these functions to extract various fields from inside a very large complicated JSON blob and is an interesting example to see the simplification these methods provide.

10.9.2 Earthquake Example

Note

If you wish to use any of this code, do not copy from this document. Get the latest code from the example source: `\nburn\examples\webclient\Earthquake`

10.9.2.1 Source Code

```
/*NB_REVISION*/

/*NB_COPYRIGHT*/
#include <init.h>
#include <stdio.h>
#include <ctype.h>
#include <buffers.h>
#include <json_lexer.h>
#include <webclient/http_funcs.h>
#include <nbtime.h>   // Include for NTP functions

const char *url = "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_day.geojson";

const char *AppName = "EarthQuake";

void UserMain(void *pd)
{
    init();                                     // Initialize network stack
    // Enable system diagnostics. Probably should remove for production code.
    EnableSystemDiagnostics();
    WaitForActiveNetwork(TICKS_PER_SECOND * 5);   // Wait for DHCP address

    iprintf("Application started\n");

    bool bTimeSet = SetTimeNTPFromPool();

    if(bTimeSet)
    {
        printf("Time Set\r\n");
    }

    ParsedJsonDataSet JsonResult;
    bool result = DoGet(url, JsonResult);


    if (result)
    {
        // JsonResult.PrintObject(true);
        int32_t nQuakes = JsonResult("metadata")("count");
        printf("In the last Day we have had %ld Earthquakes > 4.5\r\n", nQuakes);

        for (int i = 0; i < nQuakes; i++)
        {   // A JsonRef is a pointer into a ParsedJsonDataSet object varible or object to simplify access.
            // The following code creates a pointer into the event array to capture the properties object.
            JsonRef OneQuakeProperties = JsonResult("features")[i]("properties");
            if (OneQuakeProperties.Valid())
            {
                double magnitude = OneQuakeProperties("mag");
                NBString place_name = OneQuakeProperties("place");
                printf("Magnitude :%2.1f at %s", magnitude, place_name.c_str());

                if (bTimeSet)
                {
                    time_t when = OneQuakeProperties("time");
                    when /= 1000;
                    time_t now = time(NULL);
                    time_t delta = (now - when);
                    int sec = delta % 60;
                    int mins = (delta / 60) % 60;
                    int hour = (delta / 3600);
                    printf(" %02d:%02d:%02d ago", hour, mins, sec);
                }
                printf("\r\n");

            } //Array object look up is valid
            else
            {
                printf("Parse error failed to find array element %d\r\n", i);
            }

        }  // for
    }  // valid result
    else
    {
        iprintf("Failed to contact server\r\n");
    }

    while (1)
    {
```

```
        OSTimeDly(TICKS_PER_SECOND * 1);
    }
}
```

10.9.2.2 Earthquake Summary Serial Output

This is the summary output from the example:
In the last Day we have had 17 Earthquakes > 4.5 Magnitude :4.5 at Fiji region 01:44:12 ago Magnitude :4.6 at 128 km W of San Juan, Peru 10:06:35 ago Magnitude :4.5 at 7 km NW of Papayal, Peru 10:32:50 ago Magnitude :4.9 at 56 km SSE of Sand Point, Alaska 10:47:22 ago Magnitude :4.9 at 22 km SSE of Padong, Philippines 11:05:28 ago Magnitude :5.1 at 70 km WNW of San Antonio de los Cobres, Argentina 12:25:58 ago Magnitude :5.3 at 116 km SSE of Kushiro, Japan 13:12:09 ago Magnitude :4.6 at 112 km SSE of Kushiro, Japan 13:13:00 ago Magnitude :4.8 at 129 km WNW of Pangai, Tonga 13:49:43 ago Magnitude :4.9 at 256 km ESE of Ust'-Kamchatsk Staryy, Russia 20:15:25 ago Magnitude :4.6 at 120 km NNE of Tobelo, Indonesia 20:23:53 ago Magnitude :4.9 at 57 km ESE of Koseda, Japan 21:28:56 ago Magnitude :4.8 at 24 km NE of La Paz, Philippines 22:38:44 ago Magnitude :4.5 at northern Qinghai, China 23:10:02 ago Magnitude :4.6 at 8 km S of Padong, Philippines 23:13:24 ago Magnitude :4.5 at Fiji region 23:19:04 ago Magnitude :5.1 at 108 km SSE of Hihifo, Tonga 23:39:10 ago

10.9.2.3 JSON Data Parsed From Web Site

The raw JSON content is shown below. This is a good example of how a very complex JSON blob can be easily parsed with the ParsedJsonDataSet object. To view this data when running the example, uncomment the line: `//JsonResult.PrintObject(true);` in the source code.


```
{
    "type": "FeatureCollection",
    "metadata": {
        "generated": 1666815488000,
        "url": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_day.geojson",
        "title": "USGS Magnitude 4.5+ Earthquakes, Past Day",
        "status": 200,
        "api": "1.10.3",
        "count": 17
    },
    "features": [
        {
            "type": "Feature",
            "properties": {
                "mag": 4.5,
                "place": "Fiji region",
                "time": 1666809240464,
                "updated": 1666814388040,
                "tz": null,
                "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikie",
                "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikie.geojson",
                "felt": null,
                "cdi": null,
                "mmi": null,
                "alert": null,
                "status": "reviewed",
                "tsunami": 0,
                "sig": 312,
                "net": "us",
                "code": "7000ikie",
                "ids": ",us7000ikie,",
                "sources": ",us,",
                "types": ",origin,phase-data,",
                "nst": 60,
                "dmin": 3.053,
                "rms": 1,
                "gap": 74,
                "magType": "mb",
                "type": "earthquake",
                "title": "M 4.5 - Fiji region"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -178.3577,
                    -20.3848,
                    551.981
                ]
            },
            "id": "us7000ikie"
        },
        {
            "type": "Feature",
            "properties": {
                "mag": 4.6,
```

```
            "place": "128 km W of San Juan, Peru",
            "time": 1666779097415,
            "updated": 1666779906040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikgd",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikgd.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 326,
            "net": "us",
            "code": "7000ikgd",
            "ids": ",us7000ikgd,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 36,
            "dmin": 3.395,
            "rms": 0.54,
            "gap": 174,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.6 - 128 km W of San Juan, Peru"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -76.3603,
                -15.3705,
                10
            ]
        },
        "id": "us7000ikgd"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.5,
            "place": "7 km NW of Papayal, Peru",
            "time": 1666777522132,
            "updated": 1666779183255,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikg9",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikg9.geojson",
            "felt": 1,
            "cdi": 2.7,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 312,
            "net": "us",
            "code": "7000ikg9",
            "ids": ",us7000ikg9,",
            "sources": ",us,",
            "types": ",dyfi,origin,phase-data,",
            "nst": 33,
            "dmin": 0.862,
            "rms": 0.85,
            "gap": 165,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.5 - 7 km NW of Papayal, Peru"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -80.7874,
                -4.027,
                44.425
            ]
        },
        "id": "us7000ikg9"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.9,
            "place": "56 km SSE of Sand Point, Alaska",
            "time": 1666776650887,
            "updated": 1666803216746,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikg1",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikg1.geojson",
            "felt": 2,
```

```
            "cdi": 3.6,
            "mmi": 3.716,
            "alert": null,
            "status": "reviewed",
            "tsunami": 1,
            "sig": 370,
            "net": "us",
            "code": "7000ikg1",
            "ids": ",us7000ikg1,at00rkct3e,ak022dqn87rq,",
            "sources": ",us,at,ak,",
            "types": ",dyfi,impact-link,moment-tensor,origin,phase-data,shakemap,",
            "nst": 232,
            "dmin": 0.369,
            "rms": 0.6,
            "gap": 127,
            "magType": "mww",
            "type": "earthquake",
            "title": "M 4.9 - 56 km SSE of Sand Point, Alaska"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -160.2254,
                54.8564,
                41.855
            ]
        },
        "id": "us7000ikg1"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.9,
            "place": "22 km SSE of Padong, Philippines",
            "time": 1666775564622,
            "updated": 1666776869040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikfz",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikfz.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 369,
            "net": "us",
            "code": "7000ikfz",
            "ids": ",us7000ikfz,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 73,
            "dmin": 4.915,
            "rms": 0.54,
            "gap": 117,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.9 - 22 km SSE of Padong, Philippines"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                120.8605,
                17.8819,
                36.251
            ]
        },
        "id": "us7000ikfz"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 5.1,
            "place": "70 km WNW of San Antonio de los Cobres, Argentina",
            "time": 1666770734080,
            "updated": 1666775309650,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikfn",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikfn.geojson",
            "felt": 1,
            "cdi": 2.7,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 400,
            "net": "us",
```

```
            "code": "7000ikfn",
            "ids": ",us7000ikfn,",
            "sources": ",us,",
            "types": ",dyfi,origin,phase-data,",
            "nst": 55,
            "dmin": 1.573,
            "rms": 1,
            "gap": 40,
            "magType": "mww",
            "type": "earthquake",
            "title": "M 5.1 - 70 km WNW of San Antonio de los Cobres, Argentina"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -66.9953,
                -24.0942,
                185.017
            ]
        },
        "id": "us7000ikfn"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 5.3,
            "place": "116 km SSE of Kushiro, Japan",
            "time": 1666767963104,
            "updated": 1666790561040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikfi",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikfi.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 432,
            "net": "us",
            "code": "7000ikfi",
            "ids": ",us7000ikfi,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 90,
            "dmin": 1.178,
            "rms": 1.04,
            "gap": 99,
            "magType": "mww",
            "type": "earthquake",
            "title": "M 5.3 - 116 km SSE of Kushiro, Japan"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                144.7352,
                41.9619,
                35.062
            ]
        },
        "id": "us7000ikfi"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.6,
            "place": "112 km SSE of Kushiro, Japan",
            "time": 1666767912312,
            "updated": 1666769828040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikfl",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikfl.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 326,
            "net": "us",
            "code": "7000ikfl",
            "ids": ",us7000ikfl,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 27,
            "dmin": 1.163,
            "rms": 0.38,
```

```
            "gap": 176,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.6 - 112 km SSE of Kushiro, Japan"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                144.7166,
                41.9912,
                37.754
            ]
        },
        "id": "us7000ikfl"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.8,
            "place": "129 km WNW of Pangai, Tonga",
            "time": 1666765709390,
            "updated": 1666767075040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikff",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikff.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 354,
            "net": "us",
            "code": "7000ikff",
            "ids": ",us7000ikff,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 32,
            "dmin": 1.175,
            "rms": 1.12,
            "gap": 64,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.8 - 129 km WNW of Pangai, Tonga"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -175.5083,
                -19.3888,
                235.932
            ]
        },
        "id": "us7000ikff"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.9,
            "place": "256 km ESE of Ust'-Kamchatsk Staryy, Russia",
            "time": 1666742567939,
            "updated": 1666743756040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikdk",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikdk.geojson",
            "felt": null,
            "cdi": null,
            "mmi": 3.339,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 369,
            "net": "us",
            "code": "7000ikdk",
            "ids": ",us7000ikdk,ak022dqhpbfw,",
            "sources": ",us,ak,",
            "types": ",origin,phase-data,shakemap,",
            "nst": 69,
            "dmin": 5.174,
            "rms": 0.38,
            "gap": 83,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.9 - 256 km ESE of Ust'-Kamchatsk Staryy, Russia"
        },
        "geometry": {
            "type": "Point",
```

```
            "coordinates": [
                166.3788,
                55.5213,
                10
            ]
        },
        "id": "us7000ikdk"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.6,
            "place": "120 km NNE of Tobelo, Indonesia",
            "time": 1666742059732,
            "updated": 1666743214040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikde",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikde.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 326,
            "net": "us",
            "code": "7000ikde",
            "ids": ",us7000ikde,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 37,
            "dmin": 2.209,
            "rms": 0.9,
            "gap": 117,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.6 - 120 km NNE of Tobelo, Indonesia"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                128.3655,
                2.756,
                212.676
            ]
        },
        "id": "us7000ikde"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.9,
            "place": "57 km ESE of Koseda, Japan",
            "time": 1666738156180,
            "updated": 1666741703040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikcv",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikcv.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 369,
            "net": "us",
            "code": "7000ikcv",
            "ids": ",us7000ikcv,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 65,
            "dmin": 1.465,
            "rms": 0.66,
            "gap": 122,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.9 - 57 km ESE of Koseda, Japan"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                131.2036,
                30.1903,
                32.26
            ]
        },
        "id": "us7000ikcv"
```

```
        },
        {
            "type": "Feature",
            "properties": {
                "mag": 4.8,
                "place": "24 km NE of La Paz, Philippines",
                "time": 1666733968988,
                "updated": 1666735447870,
                "tz": null,
                "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikch",
                "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikch.geojson",
                "felt": 1,
                "cdi": 3.8,
                "mmi": null,
                "alert": null,
                "status": "reviewed",
                "tsunami": 0,
                "sig": 355,
                "net": "us",
                "code": "7000ikch",
                "ids": ",us7000ikch,",
                "sources": ",us,",
                "types": ",dyfi,origin,phase-data,",
                "nst": 73,
                "dmin": 4.996,
                "rms": 0.56,
                "gap": 117,
                "magType": "mb",
                "type": "earthquake",
                "title": "M 4.8 - 24 km NE of La Paz, Philippines"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    120.8722,
                    17.8003,
                    35.567
                ]
            },
            "id": "us7000ikch"
        },
        {
            "type": "Feature",
            "properties": {
                "mag": 4.5,
                "place": "northern Qinghai, China",
                "time": 1666732090914,
                "updated": 1666733158040,
                "tz": null,
                "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikc3",
                "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikc3.geojson",
                "felt": null,
                "cdi": null,
                "mmi": null,
                "alert": null,
                "status": "reviewed",
                "tsunami": 0,
                "sig": 312,
                "net": "us",
                "code": "7000ikc3",
                "ids": ",us7000ikc3,",
                "sources": ",us,",
                "types": ",origin,phase-data,",
                "nst": 61,
                "dmin": 10.77,
                "rms": 0.67,
                "gap": 64,
                "magType": "mb",
                "type": "earthquake",
                "title": "M 4.5 - northern Qinghai, China"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    92.2784,
                    37.7294,
                    10
                ]
            },
            "id": "us7000ikc3"
        },
        {
            "type": "Feature",
            "properties": {
                "mag": 4.6,
                "place": "8 km S of Padong, Philippines",
                "time": 1666731888656,
```

```
            "updated": 1666735766443,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikc6",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikc6.geojson",
            "felt": 0,
            "cdi": 1,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 326,
            "net": "us",
            "code": "7000ikc6",
            "ids": ",us7000ikc6,",
            "sources": ",us,",
            "types": ",dyfi,origin,phase-data,",
            "nst": 54,
            "dmin": 4.828,
            "rms": 0.78,
            "gap": 127,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.6 - 8 km S of Padong, Philippines"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                120.7462,
                17.975,
                48.754
            ]
        },
        "id": "us7000ikc6"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 4.5,
            "place": "Fiji region",
            "time": 1666731548904,
            "updated": 1666734071040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikc1",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikc1.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
            "alert": null,
            "status": "reviewed",
            "tsunami": 0,
            "sig": 312,
            "net": "us",
            "code": "7000ikc1",
            "ids": ",us7000ikc1,",
            "sources": ",us,",
            "types": ",origin,phase-data,",
            "nst": 26,
            "dmin": 3.012,
            "rms": 0.4,
            "gap": 124,
            "magType": "mb",
            "type": "earthquake",
            "title": "M 4.5 - Fiji region"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -178.3447,
                -20.5028,
                590.211
            ]
        },
        "id": "us7000ikc1"
    },
    {
        "type": "Feature",
        "properties": {
            "mag": 5.1,
            "place": "108 km SSE of Hihifo, Tonga",
            "time": 1666730342483,
            "updated": 1666733344040,
            "tz": null,
            "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000ikc5",
            "detail": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/us7000ikc5.geojson",
            "felt": null,
            "cdi": null,
            "mmi": null,
```

```
                "alert": null,
                "status": "reviewed",
                "tsunami": 0,
                "sig": 400,
                "net": "us",
                "code": "7000ikc5",
                "ids": ",us7000ikc5,",
                "sources": ",us,",
                "types": ",origin,phase-data,",
                "nst": 52,
                "dmin": 1.882,
                "rms": 1.07,
                "gap": 69,
                "magType": "mb",
                "type": "earthquake",
                "title": "M 5.1 - 108 km SSE of Hihifo, Tonga"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -173.386,
                    -16.8501,
                    19.916
                ]
            },
            "id": "us7000ikc5"
        }
    ],
    "bbox": [
        -178.3577,
        -24.0942,
        10,
        166.3788,
        55.5213,
        590.211
    ]
}
```


10.10 NetBurner RTOS

The NetBurner Real-Time Operating System (NBRTOS) a full featured preemptive multitasking real-time operating system, supporting multiple tasks, semaphores, mail boxes, FIFOs, queues, and critical sections just to name a few. As part of the NetBurner development package, the RTOS is pre-configured and running. The task you will normally start with is named `UserMain()`.
There are 255 priority levels (1 through 255). The lower the number, the higher the priority. Some of these tasks are reserved by the system, such as the idle task. You can specify a priority when the task is created, and change the priority later if you wish. A priority level can only be used by one task at a time. Be sure to check the return values when creating a task or changing a task priority to verify that the operation was successful.

10.10.1 What is a Preemptive RTOS?

10.10.2 Preemptive Operation, Priorities and Blocking

In a preemptive operating system the highest priority task ready to run will always run. This is an extremely important point, so we will mention it again: the highest priority task ready to run will always run. This is different than a Windows or Unix system that employs a round-robin in which each task will run based on its time slice. If you create a high priority task that can always run and never blocks, then no lower priority tasks will ever run. Lower priority tasks can only run when a higher priority task blocks on a resource or blocking function call.
For example, there are two tasks A and B. Task A has priority 50 and Task B has priority 51. Since Task A is of higher priority, it will always run (and Task B will never run) unless it calls a blocking function. Task B will then run for as long as Task A blocks; this could be 1 second due to a call to `OSTimeDly(TICKS_PER_SECOND)`, until a shared resource is available due to a call to OSSemPend(), or until data is available from a network connection due to a `select()` or `read()`. It both tasks are in a blocking state, then the RTOS idle task will run.

10.10.2.1 RTOS Blocking Functions

Generally any function that pends on a resource or creates a time delay:

- OSTimeDly()
- OSSemPend()
- OSMboxPend()
- OSQPend()
- OSFlagPendAny()
- OSFlagPendAll()

10.10.2.2 I/O Functions That Can Block

Generally any function that does a read operation or pends on a file descriptor

- select()
- read(), including variants with timeouts
- write(), blocks until at least one char can be written
- gets()
- getchar()
- fgets()

10.10.2.3 Network Functions That Can Block

Generally calls that will pend on an incoming connection or received data

- accept()
- Creation of a UDPPacket packet object when initialized to receive data

10.10.2.4 Functions That Can Enable a Task To Be "Ready To Run"

Generally functions that post to pending objects

- OSMboxPost()
- OSQPost()
- OSSemPost()

10.10.2.5 System Task Priorities

The number and type of system tasks and priorities used by the system will depend on your platform and the system features used by your application. For example, if your application calls StartHTTP( ) to enable web services, then a system task will be created that handles web server requests. The system task priority levels are defined in \nburn\nbrtos\include\constants.h.

10.10.3 Task Creation

Whether you use the Application Wizard in NBEclipse to create a new application, or start with one of the example programs, you will notice that the point at which your application takes control is a function named `UserMain()`. `UserMain()` is a task created by the system. The first few lines will consist of system initialization functions such as `init()` and `StartHttp()`.
Additional tasks are created with the `OSTaskCreatewName()` or `OSSimpleTaskCreatewName()` functions. The following is an example program that demonstrates the use of each function. When using `OSTaskCreaewName()` you specify all parameters:

- Task function name
- Optional parameter to pass to the task, NULL if not used
- Top of the task stack
- Bottom of the task stack
- Task priority
- Task name

A significant benefit of the full function call is that it returns the status of the task creation process. For example, one possible error is that you cannot create a second task at the same priority as any other task.
The 'OSSimpleTaskCreatewName()' implementation lets you specify just the task function, priority, and name.

```
/*
 *  This example demonstrates how to use two of the task creation
 *  functions:
 *  - OSSimpleTaskCreatewName()
 *  - OSTaskCreatewName()
 *
  */
#include <init.h>
#include <stdlib.h>
#include <nbrtos.h>
#include <system.h>
#include <utils.h>

const char *AppName = "OSTaskCreate Example";

// Allocate stack space
uint32_t   TaskAllParamsStack[USER_TASK_STK_SIZE];

/*-------------------------------------------------------------------------
 * Task created with OSTaskCreatewName() using all function parameters.
 * The void *pd parameter can be cast to any type of data you wish to send to
 * the task, or NULL if not used.
 *-------------------------------------------------------------------------*/
void TaskAllParams(void *pd)
{
    uint32_t loopCount = 0;
    uint32_t delayTime = (uint32_t)pd;

    iprintf("TaskAllParams delay time set to: %ld seconds\r\n", delayTime);

    while (1)
    {
        iprintf("TaskAllParams, %ld\r\n", loopCount);
        loopCount++;
        OSTimeDly(TICKS_PER_SECOND * delayTime);
    }
}

/*-------------------------------------------------------------------------
 * Task created with OSSimpleTaskCreatewName(). The primary difference from the
 * full version is that it allocates the task stack for you.
 *-------------------------------------------------------------------------*/
void TaskSimple(void *pd)
{
    uint32_t loopCount = 0;
    uint32_t delayTime = 6;

    iprintf("TaskSimple delay time set to: %ld seconds\r\n", delayTime);

    while (1)
    {
        iprintf("TaskSimple, %ld\r\n", loopCount);
        loopCount++;
        OSTimeDly(TICKS_PER_SECOND * delayTime);
    }
```

```
}

/*-------------------------------------------------------------------------
 *  UserMain
 *-------------------------------------------------------------------------*/
void UserMain(void *pd)
{
    uint32_t delayTime = 3;
    int returnCode;

    init();                                         // Initialize network stack
    WaitForActiveNetwork(TICKS_PER_SECOND * 5);     // Wait for DHCP address

    iprintf("Creating TaskAllParams....");
    returnCode = OSTaskCreatewName( TaskAllParams,                          // Task function name
                                    (void  *)delayTime,                     // Optional parameter to
      pass to the task function
                                    &TaskAllParamsStack[USER_TASK_STK_SIZE],    // Top of stack
                                    TaskAllParamsStack,                     // Bottom of stack
                                    MAIN_PRIO - 1,                          // Task priority
                                    "TaskAllParams" );                      // Task name
    if (returnCode == OS_NO_ERR)
        iprintf("Task creation successful\r\n");
    else
        iprintf("*** Error: status = %d\r\n", returnCode);


    iprintf("Creating TaskSimple\r\n");
    OSSimpleTaskCreatewName(TaskSimple, MAIN_PRIO - 2, "TaskSimple");

    while (1)
    {
        OSTimeDly(TICKS_PER_SECOND * 1);
    }
}
```

10.10.4 Protecting Shared Data

The following RTOS mechanisms can be used to protect shared data resources. They are listed in a decreasing order of severity as regarding system latency (all pend and post functions are at the same level).

OSSemPend() OSSemPost()	Protects an area of memory or resource. A task calls OSSemPend, which will block until the resource is available. OSSemPost releases the resource.
OSMboxPend() OSMboxPost()	Same as semaphore, except a pointer variable is passed as the “message”. A task or ISR can post a message, but only a task can pend on a message. Both the posting task and pending task must agree on what the pointer points to.
OSQPend() OSQPost()	A Queue is basically an array of mailboxes. It is used to post one or more messages. When you initialize a Queue, you must specify the maximum number of messages. The first message posted to the queue will be the first message extracted from the queue (FIFO).
OSFifoPend() OSFifoPost() OSFifoPostFirst() OSFifoPendNoWait()	A FIFO is similar to a queue, but is specifically designed to pass pointers to OS_FIFO structures. The first parameter of the structure must be a (void $*$) element, which is used by the operating system to create a linked list of FIFOs. When initializing a FIFO, you do not specify the maximum number of entries as with a queue. Instead, your application has the ability (and responsibility) to allocate memory (static or dynamic) in which to store the structures. This can be done statically by declaring global variables, or dynamically by allocating memory from the heap. As with a queue, the first message posted to the FIFO will be the first message extracted from the queue.

OSCritEnter OSCritExit OSCritObj	This is a counted critical section that restricts access to resources to one task at a time, sometimes called a "mutex". For example, you have a linked list that is maintained by 3 separate tasks. If one task is manipulating the list, you could first call OSCirtEnter for that object (the list). If any other task tries to manipulate the list, it will block at the OSCritEnter until the task that previously called OSCritEnter, calls OSCrit↩ Exit. Note that the number of enter calls must match number of exit calls. OSCritObj is a C++ implementation that uses scoping to automatically call the enter and exit functions. See example below.
OSLock() OSUnlock() OSLockObj	Disables other tasks, but not interrupts. Increments for each OSLock, decrements for each OSUnlock. The C++ object OSLockObj was created to assist in making sure that an unlock is called for each lock. When an OSLockObj is created, the constructor calls OSLock( ). When the object goes out of scope, OSUnlock( ) is automatically called by the destructor.
USER_ENTER_CRITICAL() USER_EXIT_↩ CRITICAL()	Macro that disables other tasks and interrupts. Increments count on enter, decrements on exit.

How do you decide which type of mechanism to use? Some guidelines are:

- If you need some type of signal, but do not need to pass any data, use a Semaphore. A semaphore is a single 32-bit integer that increments and decrements for each pend or post.
- If you want to pass a single 32-bit number, you can use a Mailbox or Queue. Most applications use the 32-bit number as the data, but it could also be a pointer to a structure or object. A queue is like an array of mailboxes. You declare the number of queue entries a compile time.
- If you want to pass a structure or object, then use a FIFO. You may be wondering how a FIFO differs from a Queue. The difference is that a Queue has a predefined number of entries. The FIFO implementation uses a linked list, so the only limit to the number of entries is available memory. Using a FIFO is not as simple as any of the other mechanisms, because your application must implement some type of memory management to allocate and deallocate the FIFO objects. This is usually done by managing a predeclared array of objects, or through dynamic memory allocation. We encourage all embedded designers to avoid dynamic memory allocation if at all possible, since in any embedded system memory fragmentation could eventually occur and the call to allocate a new object could fail. If you create an array of objects at compile time you will always be guaranteed the maximum number can exist.

10.10.5 Semaphore

A semaphore is a protected variable that is used to control access to shared system resources (such as memory or serial ports), to signal the occurrence of events and task synchronization. A task can request a semaphore and wait until the resource or event takes place (called pending). A task can also post to a semaphore to indicate it no longer needs a resource, or to signal an event has taken place. To create a semaphore you declare one of type OS_SEM and initialize with `OSSemInit()`: `OS_SEM` MySemaphore; `OSSemInit( &MySemaphore, 0 ); // set initial value to 0`
Your application tasks can now use the post and pend functions on the semaphore: `OSSemPost( &My↩ Semaphore ); // post to a semaphore OSSemPend( &MySemaphore, 0 ); // pend on a semaphore`
The second parameter in the `OSSemPend()` function specifies the number of time ticks to wait. A value of 0 waits forever. A good way to express a wait value is to use the TICKS_PER_SECOND definition provided by the RTOS: `OSSemPend( &MySemaphore, TICKS_PER_SECOND * 5)` to wait 5 seconds.

10.10.6 Queues

A message queue is an object that enables tasks and interrupt service routines to pend and post pointer sized messages. The pointer values typically point to some type of object or structure that contains the actual message or data.

10.10.7 FIFO

A FIFO is similar to a queue, but is specifically designed to pass pointers to OS_FIFO structures. The first parameter of the structure must be a (void $*$) element, which is used by the operating system to create a linked list of FIFOs. When initializing a FIFO, you do not specify the maximum number of entries as with a queue. Instead, your application has the ability (and responsibility) to allocate memory (static or dynamic) in which to store the structures. This can be done statically by declaring global variables, or dynamically by allocating memory from the heap. As with a queue, the first message posted to the FIFO will be the first message extracted from the queue.

10.10.8 Critical Section

`OSCritEnter()`, `OSCritExit()` and an `OSCritObj` enable an application to use counted critical sections that restrict resource access to one task at a time (also called a "mutex"). For example, you have a linked list that is maintained by 3 separate tasks. If one task is manipulating the list, you call `OSCritEnter()` before modifying the list. If any other task tries to access the list, the task will block at the OSCritEnter() call until the task that previously called `OSCritEnter()`, calls `OSCritExit()`. Since this is a counting critical section implementation, the number of enter calls must match number of exit calls for each task.
`OSCritObj` is a C++ implementation that uses scoping to automatically call the enter and exit functions so you do not need to manually match each enter with an ext. In comparison with `OsLock()`, `OsCritEnter()` does not restrict task swapping unless two tasks want to access the same resource. `OsLock()` prevents all task swapping.

10.10.9 OS Flags

OSFlags enables a function or task to pend on multiple flags or events, in contrast to a OSSemaphore which can pend on only a single event. The OSFlag implementation is essentially a 32-bit bitmap in which each bit position represents a "flag". You create a `OSFlag` object with `OSFlagCreate()`, then set, clean and read the flags with the appropriate function. There are a number of functions used to monitor or pend on the flags, and provide the ability to pend on any one or more of the flags being set, or pending on all of flags being set at one time.
Flag Functions

- `OSFlagSet()` Set the bits asserted with bits_to_set
- `OSFlagState()` Return the current value of the flags
- `OSFlagClear()` Clear the bits asserted in bits_to_clr
- `OSFlagPendAll()` Wait until all of the flags indicated by mask are set
- `OSFlagPendNoWait()` Check (but do not wait) if all of the flags indicated by the mask are set
- `OSFlagPendAny()` Wait until any of the flags indicated by the bit mask are set

- `OSFlagPendAnyNoWait()` Check (but do not wait) if any of the flags indicated by the mask are set

10.11 Network Protocols

This section describes the NetBurner TCP and UDP protocols.
Topic Links:

- TCP vs UDP
- TCP Server
- TCP Client
- UDP Class
- UDP Sockets

10.11.1 TCP vs UDP

A very common question that arises when designing a network application is whether to use TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). There are a few guidelines and features that can determine which would be the preferred protocol.

TCP is a point to point stream based protocol. It is used in applications where reliability and data sequencing is needed: acknowledgments, error detection, re-transmission of data, and will guarantee the data received will be sequenced in the same order as it was sent.

UDP is a connectionless protocol that does not guarantee delivery or data packet sequence (although each segment is numbered). It is a send and forget protocol that does not use acknowledgments. A common comparison is that TCP is similar to a phone call and UDP is similar to a post card. With TCP, you connect to a specific destination phone number. When that person answers, they say "hello", you say "Hi, my name is Bob", and then the conversation continues with each side speaking and responding (in a well-behaved conversation!). With UDP you essentially transmit a datagram, like writing on a post card, and send it without verifying it was received.

When choosing between TCP and UDP, some major concerns are the overhead it takes to establish a TCP connection, speed and the reliability of data transmission. For example, SNMP uses UDP. SNMP is used to monitor networks, and many messages are sent and received for status updates and alerts. If TCP were used, the overhead of establishing, maintaining and closing a connection for each message and each host would create a lot of unnecessary traffic. A second example of when UDP is a better choice is when an application handles its own reliability at the application layer. Using TCP in this instance would be redundant.

As mentioned previously, TCP is a stream based protocol and UDP is a datagram based protocol. Let's take an example of an application in which a NetBurner device takes A/D readings and sends them to another network device or host PC. Using UDP, each output operation (i.e. creating and sending a UDP packet) results in exactly one IP packet being created and sent. The result of taking and sending 10 A/D readings is that the host will receive 10 individual packets, each containing one reading.
The host PC can then easily identify each reading, although each reading will have to be sent with a sequence number so that the reading order can be recreated. If TCP is used with a single continuous network connection (i.e. the connection is not closed and reestablished for each reading), you do not have control over how many readings are send with each IP packet. You would need to add start message and stop message identifiers to separate the data from each reading.

Some applications that use TCP are: HTTP, FTP, Telnet and SMTP. Some applications that use UDP are: DHCP, BOOTP, SNMP and DNS.

10.11.1.1 TCP

TCP is used to create a reliable byte stream connection between two network hosts. The host that listens for incoming connections is referred to as the server, and the host that initiates a connection the client. Although TCP and UDP both use IP, TCP sends information as a stream of data. There are no record markers to delimit the data. For example, if a server is sending analog-to-digital (A/D) readings to a client, the client will see a stream of digits; TCP will not automatically insert delimiters to allow the client to determine where one measurement ends and the next begins. To the client, the stream may look like: `98273129323424`. Even if the client knew each reading was 4 digits, it would not know where one ended and the next began. Four parameters are required for a TCP connection: source IP address, source port number, destination IP address, and destination port number.

In contrast to TCP, UDP (covered in the next chapter) is an unreliable, datagram-oriented connectionless protocol. Delivery is not guaranteed, but each output operation creates and sends one UDP datagram. In the above A/D example, each reading (or some number of multiple readings) could be sent as a single datagram and the client could then process one datagram at a time.

10.11.2 TCP Server

A TCP server is basically an application that creates a listening socket, and then listens on the socket for incoming connections. When an incoming connection request is received, tit must be accepted before communication can begin. A web server is an example of a TCP server. A web server listens on port number 80 for incoming connections. Once a connection is established, the web browser sends a GET request to the web server, which then sends the requested information and terminates the connection.

To connect to a TCP server you must specify a port number. A port number is a 16-bit value. Since you must know the port number before connecting, many port numbers have been defined for common protocols, and are called well-known port numbers. Some of these values are shown below:

```
FTP     21
Telnet  23
SMTP    25
DNS     53
TFTP    69
HTTP    80
POP3    110
NTP     123
```

10.11.2.1 NetBurner TCP Server Basics

When creating a TCP server with the NetBurner you will use the following functions:

- `listen()`: Open a listening socket on a specific port number
- `accept()`: Accept a connection request on a listening port
- I/O functions to send and receive data such as `read()`, `write()`, `fdprintf()`, `writestring()`, etc.
- `close()`: Close a network socket

10.11.2.2 Simple TCP Server Example

This example is located in `\nburn\examples\TCP`. It will listen on port 23 for incoming connections, send a sign on message to the client when a connection is made, and display all received data to the debug serial port. A telnet program on a host PC can be used to connect to the server.

```
#include <predef.h>
#include <stdio.h>
#include <tcp.h>
#include <nbrtos.h>
#include <iosys.h>
#include <init.h>
#include <fdprintf.h>

const char *AppName = "Simple TCP Server Example";

#define TCP_LISTEN_PORT 23   // Telnet port number
#define RX_BUFSIZE (4096)

char RXBuffer[RX_BUFSIZE];

// Allocate task stack for TCP listen task
uint32_t   TcpServerTaskStack[USER_TASK_STK_SIZE];

/*-------------------------------------------------------------------
 * TCP Server Task
 *-------------------------------------------------------------------*/
void TcpServerTask(void * pd)
{
    int listenPort = (int) pd;

    // Set up the listening TCP socket
    int fdListen = listen(INADDR_ANY, listenPort, 5);

    if (fdListen > 0)
    {
        IPADDR      clientAddress;
        uint16_t    clientPort;

        while(1)
        {
            /* The accept() function will block until a TCP client requests a connection. Once a client
             * connection is accepting, the file descriptor fdAccept is used to read/write to it.
             */
            iprintf( "Waiting for connection on port %d...\n", listenPort );
            int32_t fdAccept = accept(fdListen, &clientAddress, &clientPort, 0);
            iprintf("Connected to: %I\r\n", GetSocketRemoteAddr(fdAccept));

            writestring(fdAccept, "Welcome to the NetBurner TCP Server\r\n");
            fdprintf(fdAccept, "You are connected to IP Address %I:%d\r\n", GetSocketRemoteAddr(fdAccept),
      GetSocketRemotePort(fdAccept) );

            while (fdAccept > 0)
            {
                /* Loop while connection is valid. The read() function will return 0 or a negative number if
      the
                 * client closes the connection, so we test the return value in the loop. Note: you can also
      use
                 * ReadWithTimout() in place of read to enable the connection to terminate after a period of
      inactivity.
                */
                int n = 0;
                do
                {
                    n = read( fdAccept, RXBuffer, RX_BUFSIZE );
                    RXBuffer[n] = '\0';
                    iprintf( "Read %d bytes: %s\n", n, RXBuffer );
                } while ( n > 0 );

                iprintf("Closing client connection: %I\r\n", GetSocketRemoteAddr(fdAccept) );
                close(fdAccept);
                fdAccept = 0;
            }
        } // while(1)
    } // while listen
}


/*-------------------------------------------------------------------
 User Main
 -------------------------------------------------------------------*/
void UserMain(void * pd)
{
    init();

    // Create TCP Server task
    OSTaskCreatewName( TcpServerTask,
                       (void  *)TCP_LISTEN_PORT,
                       &TcpServerTaskStack[USER_TASK_STK_SIZE] ,
                       TcpServerTaskStack,
                       MAIN_PRIO - 1,   // higher priority than UserMain
```

```
                        "TCP Server" );

    while (1)
    {
        OSTimeDly( TICKS_PER_SECOND * 5 );
    }
}
```

This is an extremely simple example designed to illustrate how the `accept()` and `listen()` calls can be used. It only listens to a single port number, and processes a single connection at a time. Any information sent from the Client will be displayed in the MTTTY window. The application does not have the capability to terminate the incoming connection.

There are `#define` options for the TCP listen port number and the incoming TCP buffer storage array size. `RXBuffer[]` is then declared and will hold the received data. The `listen()` function call sets up a socket to listen for an incoming connection from any IP address on port number 23, the telnet port number.

If the `listen()` succeeds in creating a listening socket, we then enter a second while loop. The application will block at the `accept()` function call until an incoming connection request is received, such as when the telnet program on a PC attempts to connect. When the connection is established, the `accept()` function returns and the sign-on message is sent to the telnet application.

We now enter the do loop: `while(n > 0)`. The `read()` function will block until data is received or an error occurs such as the client terminating the connection. When data is sent from the telnet application, the `read()` function will return with the data in the `RXBuffer[]` array. The application will stay in this while loop until the connection is terminated by the telnet client (or you reset the NetBurner device). If the connection is broken by the telnet client, the application will then loop back to the `accept()` function call and wait for another incoming connection.

10.11.2.3 Advanced TCP Server for Multiple Connections

In the previous example the TCP server processed only a single incoming connection. The `select()` function has the ability to pend on multiple file descriptors, which can used for TCP or serial connections. The example below demonstrates how the TCP Server can be written using `select()`.

```
/*-------------------------------------------------------------------------
    This example creates a TCP server that listens on the specified TCP port
    number and can handle multiple TCP connections simultaneously (10 in this
    example). The select() function is a great way method to pend and process
    multiple connections.

    An easy way to test the example is to use multiple Telnet sessions to create
    simultaneous connections to the NetBurner device. Status messages are sent
    out stdio to the debug serial port, and to the client TCP connections.
*-------------------------------------------------------------------------*/
#include <predef.h>
#include <constants.h>
#include <utils.h>
#include <system.h>
#include <iosys.h>
#include <stdio.h>
#include <ctype.h>
#include <tcp.h>
#include <init.h>
#include <ipshow.h>

const char* AppName = "TCP Multiple Sockets Example";

#define listenPort      (23)         // TCP port number to listen on
#define maxConnections  (10)         // Max number of file descriptors/connections
#define readBufferSize  (1024)       // Connection read buffer size

int32_t fdArray[maxConnections];     // Array of TCP file descriptors

/*-------------------------------------------------------------------------
 * User Main
 *-------------------------------------------------------------------------*/
```

```
void UserMain( void* pd )
{
    init();
    WaitForActiveNetwork(TICKS_PER_SECOND * 10);
    showIpAddresses();

     // Listen for incoming TCP connections. You only need to call listen() one time.
     // Any incoming IP address is allowed, queue up to 5 connection requests at one time
    int32_t fdListen = listen( INADDR_ANY, listenPort, 5 );
    iprintf( "Listening for incoming connections on port %d\r\n", listenPort );

    while ( 1 )
    {
        // Declare file descriptor sets for select()
        fd_set readFds;
        fd_set errorFds;

        // Init the fd sets
        FD_ZERO( &readFds );
        FD_ZERO( &errorFds );

        // Configure the fd sets so select() knows what to process. In this case any fd data to be read, or
      an error
        for ( int32_t i = 0; i < maxConnections; i++ )
        {
            if ( fdArray[i] )   // The fd in the array will be > 0 if open and valid, so reset the file
      descriptor sets
            {
                FD_SET( fdArray[i], &readFds );
                FD_SET( fdArray[i], &errorFds );
            }
        }

        // select() should also process the listen fd
        FD_SET( fdListen, &readFds );
        FD_SET( fdListen, &errorFds );

        // select() will block until any fd has data to be read, or has an error. When select() returns,
        // readFds and/or errorFds variables will have been modified to reflect the events.
        select( FD_SETSIZE, &readFds, ( fd_set * )0, &errorFds, 0 );

        // If the listen fd has a connection request, accept it.
        if ( FD_ISSET( fdListen, &readFds ) )
        {
            IPADDR      clientIp;
            uint16_t    clientPort;

            int fdAccept = accept( fdListen, &clientIp, &clientPort, 0 );

            // If accept() returned, find an open fd array slot
            if ( fdAccept > 0 )
            {
                for ( int32_t i = 0; i < maxConnections; i++ )
                {
                    if ( fdArray[i] == 0 )
                    {
                        fdArray[i] = fdAccept;
                        writestring( fdAccept, "Welcome to the NetBurner Multi-Socket TCP Server! 'Q' to
      quit." );
                        iprintf( "Added connection on fd[%d] = %d, Client IP: %I:%d\r\n",
                                i, fdAccept, GetSocketRemoteAddr(fdAccept), GetSocketRemotePort(fdAccept) );
                        fdAccept = 0;
                        break;
                    }
                }
            }

            // If no array positions are open, close the connection
            if ( fdAccept )
            {
                writestring( fdAccept, "I am sorry, but the server is full\r\n" );
                iprintf("Server Full\r\n");
                close( fdAccept );
            }
        }

        // If the listen fd has an error, close it and reopen
        if ( FD_ISSET( fdListen, &errorFds ) )
        {
            close( fdListen );
            fdListen = listen( INADDR_ANY, listenPort, 5 );
        }

        // Process each fd array element and check it against readFds and errorFds.
        for ( int32_t i = 0; i < maxConnections; i++ )
        {
            if ( fdArray[i] )
```

```
            {
                // Check for data to be read
                if ( FD_ISSET( fdArray[i], &readFds ) )
                {
                    char buffer[readBufferSize];
                    int rv = read( fdArray[i], buffer, readBufferSize );
                    if ( rv > 0 )
                    {
                        buffer[rv] = 0;
                        if ( buffer[0] == 'Q' )
                        {
                            iprintf( "Closing connection fd[%d]\r\n", i );
                            writestring( fdArray[i], "Bye\r\n" );
                            close( fdArray[i] );
                            fdArray[i] = 0;
                        }
                        else
                        {
                            iprintf( "Read \"%s\" from fd[%d]\r\n", buffer, i );
                            sniprintf( buffer, readBufferSize, "Server read %d byte(s)\r\n", rv );
                            writestring( fdArray[i], buffer );
                        }
                    }
                    else
                    {
                        iprintf( "Read Error on fd[%d]\r\n", fdArray[i] );
                        FD_SET( fdArray[i], &errorFds );
                    }
                } // data available to read

                // Check for errors
                if ( FD_ISSET( fdArray[i], &errorFds ) )
                {
                    iprintf( "Error on fd[%d], closing connection\r\n", i );
                    close( fdArray[i] );
                    fdArray[i] = 0;
                }
            }   // if fd is valid
        }   // process each connection in the array
    } // while (1)
}  // UserMain
```

10.11.3 TCP Client

The previous section demonstrated how a TCP Server waits for incoming connections from a TCP client. This section will demonstrate how a TCP client initiates a connection using the `connect()` function. Features of this example:

- A web page interface is created with input fields for the destination IP address, destination port number, and the message to send to the server
- Demonstrates how to use HTML forms to submit user data
- The web page interface will use the `connect()` function to create an outgoing connection to a TCP Server, send the message, and close the socket

A screen shot of the running application is shown below:

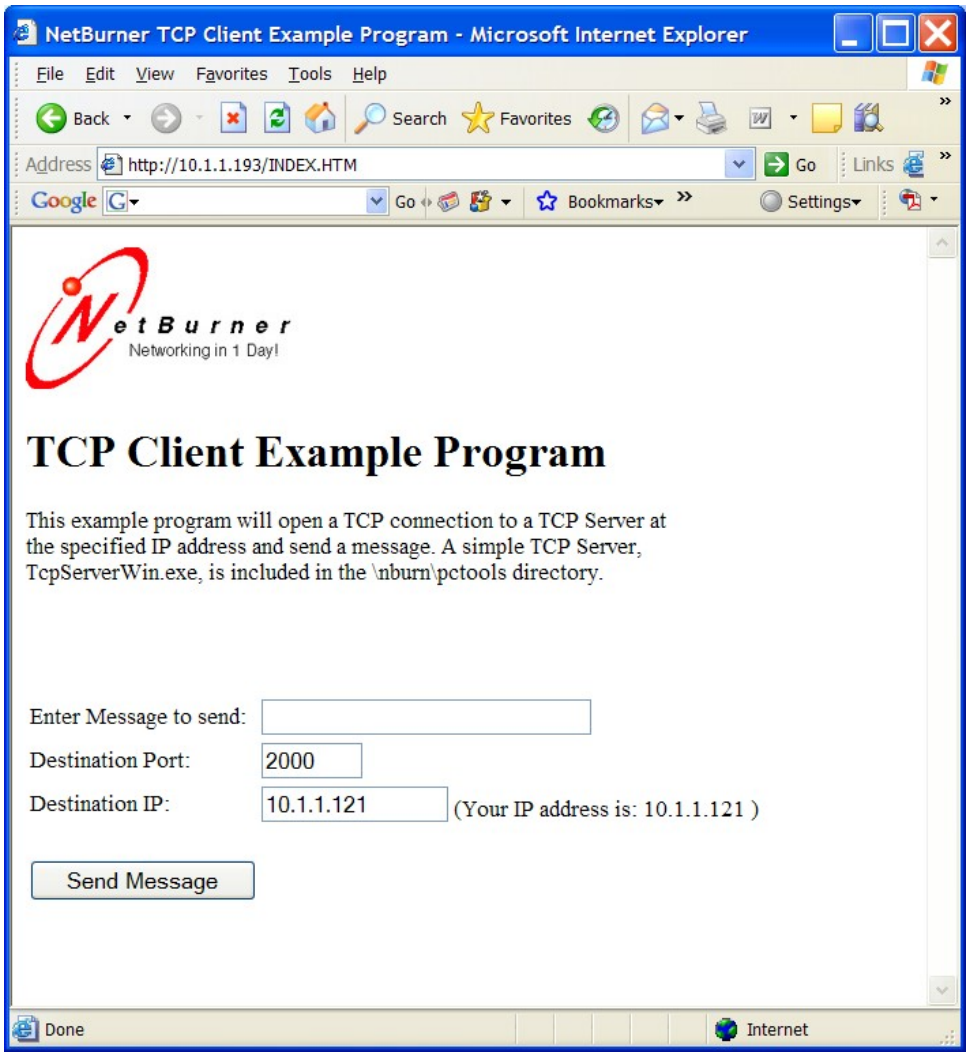


Figure 10.6 TCP Client Web Interface

10.11.3.1 Application Source Code

The application consists of the following source files:

File Name	Description
main.cpp	Application initialization, starts the web server
clientweb.h	Header file for clientweb.cpp
clientweb.cpp	Handles the user interface interaction, makes the outgoing connection
index.html	HTML code for the user interface

10.11.3.1.1 main.cpp This is a very simple source file that initializes the system. All the action occurs in clientweb.cpp when a user posts web form data. The purpose of the `RegisterPost()` function is to use our own POST handler instead of just calling the system POST handler.

```
/*-------------------------------------------------------------------
  This program demonstrates how to create a TCP Client. All interaction is
  through the web page interface, which enables you to type a message as
  well as the IP address and port number of the destination TCP server.
  You should be able to use any TCP server. If you do not have one,
  we have a windows example TCP server in the \nburn\pctools\TCP\TcpServerWin
  directory (as well as source code).
 -------------------------------------------------------------------*/
#include <predef.h>
#include <stdio.h>
#include <startnet.h>
#include <init.h>
#include <ipshow.h>
#include "clientweb.h"

void UserMain(void * pd)
{
    init();                                 // Initialize system
    StartHttp();                            // Start web server
    WaitForActiveNetwork(TICKS_PER_SECOND * 10);
    showIpAddresses();                      // Display IP information to serial port
    RegisterPost();                         // Register custom HTML Post function

    while (1)
    {
        OSTimeDly( TICKS_PER_SECOND );
    }
}
```

10.11.3.1.2 clientweb.h The only function we need to call outside of clientweb.cpp is `RegisterPost()`

```
#ifndef _NB_CLIENTWEB_H
#define _NB_CLIENTWEB_H

void RegisterPost();

#endif
```

10.11.3.1.3 index.html Web page implements a form to send data to the TCP server using a web POST. Inside the HTML `<form>` tags there is a table for formatting, with three HTML `<input>` fields. The message input field is straight HTML and any data entered in the text box will be sent to the server as data in the POST. The second and third input fields make use of the NetBurner CPPCALL function callback tag to execute functions that provide default values in the input fields when the page is displayed.

```
<html>
<head>
<title>NetBurner TCP Client Example Program</title>
</head>

<img src="logo.jpg">
<br>
<h1>TCP Client Example Program</h1>
This example program will open a TCP connection to a TCP Server at
the specified IP address and send a message. A simple TCP Server,
TcpServerWin.exe, is included in the \nburn\pctools directory.

<form ACTION="nothing.html" METHOD=POST><br>

<table>
  <tr>
    <td>Enter Message to send: </td>
    <td><input NAME="tfMessage" TYPE="text" SIZE="30"></td>
  </tr>

  <tr>
    <td>Destination Port: </td>
    <td><input NAME="tfDestPortNum" <!--CPPCALL WebDestPort --> TYPE="text" SIZE="6"></td>
  </tr>

  <tr>
    <td>Destination IP: </td>
    <td><input NAME="tfDestIpAddr" <!--CPPCALL WebDestIp --> TYPE="text" SIZE="15">
    (Your IP address is: <!--CPPCALL WebShowClientIp --> )
    </td>
  </tr>

  <tr>
    <td> <br></td>
  </tr>

  <tr>
    <td><input NAME="SendMessage" TYPE="submit" VALUE="Send Message"></td>
```

```
  </tr>
  <br><br>
</table>

</form>
</body>
</html>
```

10.11.3.1.4 clientweb.cpp This code module handles the dynamic content and web server interface. When the web page is displayed, current values for the port numbers and IP addresses are filled in as default values.

```
/*-------------------------------------------------------------------
 * This code implements the web page entries for the message,
 * destination IP address and destination port number. When the
 * web page first loads it will automatically fill in the IP
 * address from the source requesting the web page, because in
 * most cases it will also be the address of the TCP Server.
 * The web page is a form, and when a user presses the submit
 * button the SendMsg() function will open a TCP connection to
 * the server, send the message, and close the connection.
 * Any error messages will be sent to stdout and can be viewed
 * with MTTTY.
 *
 * A TCP server program must already be listening at the specified
 * IP address and port number for the message to be sent. A simple
 * TCP Server called TcpServerWin.exe is located in the
 * \nburn\pctools directory of your NetBurner installation.
 -------------------------------------------------------------------*/
#include <predef.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <http.h>
#include <iosys.h>
#include <utils.h>
#include <tcp.h>
#include <string.h>
#include <fdprintf.h>
#include "clientweb.h"

#define APP_VERSION "Version 1.2 9/1/2018"
#define POST_BUFSIZE (4096)         // form post buffer size

int    gDestPort;
IPADDR gDestIp;

/*-------------------------------------------------------------------
 Sends a message to the specified host.
 -------------------------------------------------------------------*/
void SendMsg( IPADDR destIp, int destPort, char *msg )
{
    iprintf( "Connecting to: %I:%d\r\n", destIp, destPort );
    int fd = connect( destIp, destPort, TICKS_PER_SECOND * 5 );
    if (fd < 0)
    {
        iprintf("Error: Connection failed\r\n");
    }
    else
    {
        iprintf( "Attempting to write: \"%s\" \r\n", msg );
        int n = write( fd, msg, strlen(msg) );
        iprintf( "Wrote %d bytes\r\n", n );
        close( fd );
    }
}


/*-------------------------------------------------------------------
 * Show destination port number on web page
 * -------------------------------------------------------------------*/
void WebDestPort(int sock, PCSTR url)
{
    if ( gDestPort == 0 )       // If no dest port is specified, use a default
        gDestPort = 2000;
    fdprintf(sock, "VALUE=\"%d\" ", gDestPort);
}

/*-------------------------------------------------------------------
 Show destination ip address on web page
 -------------------------------------------------------------------*/
void WebDestIp(int sock, PCSTR url)
{
    if ( gDestIp.IsNull() )     // If no dest ip address has been entered, use the one that requested the
```

```
      web page.
        fdprintf( sock, "VALUE=\"%I\" ", GetSocketRemoteAddr(sock) );
    else
        fdprintf( sock, "VALUE=\"%I\" ", gDestIp );
}

/*-------------------------------------------------------------------
 * Show destination ip address on web page
 *-------------------------------------------------------------------*/
void WebShowClientIp(int sock, PCSTR url)
{
    fdprintf( sock, "%I", GetSocketRemoteAddr(sock));
}

/*-------------------------------------------------------------------
 * Process HTTP Post
 *-------------------------------------------------------------------*/
int MyDoPost(int sock, char * url, char * pData, char * rxBuffer)
{
    int max_chars = 40;
    // The buffer is 4096 bytes, make it static so that it does
    // not take up task stack space, but instead uses global space.
    static char buf[POST_BUFSIZE];

    iprintf( "Post Data: %s\r\n", pData );  // print all data sent from app

    // The SendMessage button on the web page initiates a form POST to send the message to the TCP server.
    if ( ExtractPostData( "SendMessage", pData, buf, max_chars) > 0 )
    {
        iprintf( "Processing SendMessage Post\r\n" );

        if (ExtractPostData("tfDestPortNum", pData, buf, max_chars) == -1)
            iprintf("Error reading post data: tfDestPortNum\r\n");
        else
            gDestPort = (int)atoi(buf);

        if (ExtractPostData("tfDestIpAddr", pData, buf, max_chars) == -1)
            iprintf( "Error reading post data: tfDestIpAddr\r\n");
        else
            gDestIp = AsciiToIp( buf );

        if ( ExtractPostData( "tfMessage", pData, buf, max_chars ) < 0 )
            iprintf("Error reading post data: tfMessage\r\n");
        else
            SendMsg( gDestIp, gDestPort, buf );
    }

    // We have to respond to the post with a new HTML page. In this case we
    // will redirect so the browser will go to that URL for the response.
    RedirectResponse( sock, "index.htm" );

    return 0;
}

/*-------------------------------------------------------------------
 Specify the function to call when a user clicks Submit on the
 web page form.
 -------------------------------------------------------------------*/
void RegisterPost()
{
    SetNewPostHandler( MyDoPost );
}
```

10.11.3.1.5 TCP Client Program Operation The application boots and executes the initialization functions in main.cpp. From that point forward all interaction occurs through the web interface. When a client web browser connects to the device's web server, the CPPCALL functions `WebDestPort()` and `WebDestIp()` fill in the current values in the input text fields. A port number of 2000 is specified as the default port. The `WebDestIp()` function will default to the last value entered by the user, or if no value has been entered yet, it will enter the IP address of the host running the web browser. It extracts the IP address from the TCP connection request.

The message to send it is just a temporary string displayed in the web browser. To send a message, the user clicks on the Send Message submit button. This is a HTML command that sends a web POST to the web server containing all the information contained in the form. This is where the `MyDoPost()` function in clientweb.↩ cpp comes into play. It parses the POST data, assigns data tot he runtime variables, extracts the message data, and calls `SendMsg()`. `SendMsg()` in clientweb.cpp makes a `connect()` call using the passed message and the values of the destination port number and IP address. Once the connection is made, `SendMsg()` calls `write()` to send the message followed by `close()` to terminate the connection.

Note

The "%I" parameter in the `iprintf()` functions will display an IP addresses for an IPADDR object in dotted notation (eg. 192.168.1.1). An IPADDR object can hold an IPv4 or IPv6 address. For IPADDR4 objects, use "%hI".

10.11.4 UDP Class

The NetBurner API provides two mechanisms to implement UDP: sockets and a C++ class. The choice of which to use is Dependent of which method you find most comfortable using. There is not a performance difference.

10.11.4.1 Receiving UDP Packets Using the UDP Class

To receive UDP Packets:

- Create an OS_FIFO to hold incoming UDP packets
- Register a UDP FIFO with `RegisterUDPFifo()` to listen for and store incoming packets. If listening on more than one port number, a separate OS_FIFO can be used for each port, or one OS_FIFO can be used to listen on multiple ports by calling `RegisterUDPFifo()` for each port number and specifying the same OS_FIFO.
- Call a UDP constructor such as: `UDPPacket upkt(&fifo, timeout)`, which will block until a packet is received, then accept and store the packet.
- Call the `Validate()` member function to verify the packet
- Call the `GetDataBuffer()` member function to obtain the data in the packet

10.11.4.2 Sending a UDP Packet Using the UDP Class

To send a UDP packet:

- Declare an instance of a UDP object
- Specify the source and destination port numbers
- Add data to the packet using the class member functions
- Call the `Send()` member function to send the packet

Note

Unlike TCP, there is no connection between the client and sever, so I/O functions such as read() and write() cannot be used. When sending, a UDP packet must be created for each transmission.

10.11.4.3 UDP Send/Receive Example with UDP Class

Note

You can use the NetBurner UDPTerminal utility on Windows platforms to send/receive UDP packets on your PC.

```
/*
 *  This example program will receive a UDP packet from another device or
 *  host computer, and then send a response. To run the example, connect a serial
 *  port from your PC to the debug serial port on your NetBurner device and run a
 *  terminal program such as MTTTY. On the PC, run the NetBurner UDP Terminal
 *  (be sure to set the IP address and port numbers to match). You will then
 *  be able to type characters in the UDP Terminal and see them in MTTTY,
 *  and vice versa.
 *
 *  You will be prompted for the port number to send/receive data and the
 *  destination IP address of the other device or host. Note that the application
 *  uses the same port number to send and receive data, but you can use any other
 *  port number you wish.
 *
 *  The application will create a thread to receive packets and display them
 *  on the debug port, while the main task will take any data you type in
 *  to the MTTTY terminal and send it as a UDP packet to the destination IP
 *  address.
 */

#include <init.h>
#include <stdlib.h>
#include <system.h>
#include <udp.h>
#include <utils.h>

const char *AppName = "UDP Send/Receive Example";

// Allocate stack space for the listen task
uint32_t   UdpReceiveTaskStack[USER_TASK_STK_SIZE];

/*
 *  This UDP task will wait for incoming packets on the specified port number,
 *  which is passed as a OSTaskCreate() void * parameter.
 */
void UdpReceiveTask(void *pd)
{
    static OS_FIFO fifo;   // Create a FIFO for the UDP packet and initialize it
    int listenPort = (int)pd;

    iprintf("Listening on UDP port: %d\r\n", listenPort);

    // Register the OS_FIFO. Received packets will be stored in the OS_FIFO.
    RegisterUDPFifo(listenPort, &fifo);

    while (1)
    {
        // Construct a UDP packet object using the previously declared FIFO.
        // The UDP constructor will block until a packet has been received.
        // The second parameter is a timeout value (time in ticks). A value of 0 will
        // wait forever.
        UDPPacket upkt(&fifo, 0 * TICKS_PER_SECOND);

        // Did we get a valid packet, or just time out?
        if (upkt.Validate())
        {
            uint16_t len = upkt.GetDataSize();
            iprintf("Received UDP packet with %d bytes from: %I\r\n", (int)len, upkt.GetSourceAddress());
            ShowData(upkt.GetDataBuffer(), len);   // hex dump function
            iprintf("\r\n");
        }
    }
}

/*
 *  UserMain
 */
void UserMain(void *pd)
{
    int     portNumber;
    IPADDR  destIpAddress;
    char    buffer[80];

    init();                                         // Initialize network stack
    WaitForActiveNetwork(TICKS_PER_SECOND * 5);     // Wait for DHCP address

    iprintf("Application: %s\r\nNNDK Revision: %s\r\n", AppName, GetReleaseTag());
```

```
    iprintf("Enter the UDP port number (will be used for send & receive): ");
    gets(buffer);
    portNumber = atoi(buffer);

    iprintf("\r\nEnter the destination IP Address: ");
    gets(buffer);
    destIpAddress = AsciiToIp(buffer);

    iprintf("Listening/Sending on UDP Port %d, Sending to IP address: %I\r\n", portNumber, destIpAddress);

    // Create UDP listen task
    OSTaskCreatewName( UdpReceiveTask,
                       (void  *)portNumber,
                       &UdpReceiveTaskStack[USER_TASK_STK_SIZE] ,
                       UdpReceiveTaskStack,
                       MAIN_PRIO - 1,   // higher priority than UserMain
                       "UDP Receive" );

    // while loop to process user input and send as a UDP packet
    iprintf("Enter data and hit return to send.\r\n");
    while (1)
    {
        gets(buffer);

        UDPPacket pkt;
        pkt.SetSourcePort(portNumber);
        pkt.SetDestinationPort(portNumber);
        pkt.AddData(buffer);
        pkt.AddDataByte(0);
        pkt.Send(destIpAddress);
        iprintf("\r\n");

        iprintf("Sent \"%s\" to %I:%d\r\n", buffer, destIpAddress, portNumber);
    }
}
```

10.11.4.3.1 Sending Packets In the while loop of `UserMain()` you can see that a UDPPacket object named "pkt" is created. Member functions are then called to specify the source port number, destination port number, add the message data, add a null string terminator, and finally to send the packet. In practice it is a good idea to choose random source port numbers.

10.11.4.3.2 Receiving Packets `UserMain()` creates a task named `UdpReceiveTask()`. The task declares an OS_FIFO to use to store incoming packets. Note that it is static so the FIFO is located in global variable space. A UDPPacket object constructor is then called for upkt. It will block until a packet is received. When one does arrive it is verified with the `Validate()` member function. If it is a valid packet the `ShowData()` utility function is used to display the data in hex format to the debug serial port.

10.11.5 UDP Sockets

The NetBurner API provides two mechanisms to implement UDP: sockets and a C++ class. The choice of which to use is Dependant of which method you find most comfortable using. There is not a performance difference.

10.11.5.1 Receiving UDP Packets using UDP Sockets

To receive UDP Packets:

- Use the `CreateRxUdpSocket()` function to open a listening socket. It will return a file descriptor.
- Use the `recvfrom()` function to receive packets. The industry standard behavior of this function is to block forever until a packet is received. If you want to allow your application to have better control, you can wrap the `recvfrom()` function inside a `select()` function using the UDP file descriptor and a timeout.

10.11.5.2 Sending UDP Packets using UDP Sockets

To send UDP Packets:

- Use the `CreateTxUdpSocket()` function to open a socket. It will return a file descriptor.
- Use the `sendto()` function to send packets.

10.11.5.3 UDP Send/Receive Example Using Sockets

```
/*
 *  This application will send/receive UDP packets with another host on a network,
 *  such as a PC. Use the MTTTY serial port program to access the menu and
 *  prompts to specify the destination IP address and port number.
 *
 *  NetBurner also supplies an API for handling UDP as a C++ Class using UDPPacket.
 *
 *  For an external UDP host you can use the NetBurner Java example, or the
 *  NetBurner UDP terminal program.
 */

#include <init.h>
#include <stdlib.h>
#include <string.h>
#include <system.h>
#include <udp.h>
#include <utils.h>

const char *AppName = "UDP Sockets Example";

// Allocate stack space for the listen task
uint32_t   UdpReceiveTaskStack[USER_TASK_STK_SIZE];


/*
 *  This task will wait for incoming UDP packets and process them.
 */
void UdpReceiveTask(void *pd)
{
    int listenPort = (int)pd;

    iprintf("UdpReceiveTask monitoring port %d\r\n", listenPort);

    // Create a UDP socket for receiving
    int udpFd = CreateRxUdpSocket(listenPort);
    if (udpFd <= 0)
    {
        iprintf("Error Creating UDP Listen Socket: %d\r\n", udpFd);
        while (1)
        {
            OSTimeDly(TICKS_PER_SECOND);
        }
    }
    else
    {
        iprintf("Listening for UDP packets on port %d\r\n", listenPort);
    }

    while (1)
    {
        IPADDR sourceIpAddress;     // UDP packet source IP address
        uint16_t localPort;         // Port number UDP packet was sent to
        uint16_t sourcePort;        // UDP packet source port number
        char buffer[80];

        int len = recvfrom(udpFd, (uint8_t *)buffer, 80, &sourceIpAddress, &localPort, &sourcePort);
        buffer[len] = '\0';

        iprintf("\r\nReceived a UDP packet with %d bytes from : %I\r\n%s\r\n", len, sourceIpAddress,
      buffer);
    }
}

/*
 *  UserMain
 */
void UserMain(void *pd)
{
    int     portNumber;
    IPADDR  destIpAddress;
```

```
    char    buffer[80];

    init();                                         // Initialize network stack
    WaitForActiveNetwork(TICKS_PER_SECOND * 5);     // Wait for DHCP address

    iprintf("Application: %s\r\nNNDK Revision: %s\r\n", AppName, GetReleaseTag());

    // Get destination IP address
    iprintf("Enter the UDP Server destination IP address: ");
    gets(buffer);
    destIpAddress = AsciiToIp(buffer);
    iprintf("\r\n");

    // Get the port number. This application uses the same port number for send and receive
    iprintf("Enter the source/destination port number: ");
    gets(buffer);
    portNumber = atoi(buffer);
    iprintf("\r\n");

    // Create a UDP socket for sending
    int udpFd = CreateTxUdpSocket(destIpAddress, portNumber, portNumber);
    if (udpFd <= 0)
    {
        iprintf("Error Creating UDP Socket: %d\r\n", udpFd);
        while (1)
        {
            OSTimeDly(TICKS_PER_SECOND);
        }
    }
    else
    {
        iprintf("Sending/Receiving with host %I: %d\r\n", destIpAddress, portNumber);
    }

    // Create UDP receive task. The priority is higher than UserMain() so packets get processed as they are
      received
    OSTaskCreatewName( UdpReceiveTask,
                       (void  *)portNumber,
                       &UdpReceiveTaskStack[USER_TASK_STK_SIZE] ,
                       UdpReceiveTaskStack,
                       MAIN_PRIO - 1,   // higher priority than UserMain
                       "UDP Receive" );

    iprintf("Enter data and hit enter to send.\r\n");

    while (1)           // Loop forever displaying UDP data
    {
        gets(buffer);
        iprintf("\r\n");
        iprintf("Sending \"%s\" using UDP to %I : %d\r\n", buffer, destIpAddress, portNumber);

        sendto(udpFd, (uint8_t *)buffer, strlen(buffer), destIpAddress, portNumber);
        iprintf("\r\n");
    }
}
```

10.12 SSL/TLS Programming Guide

10.12.1 SSL/TLS Overview

The NetBurner SSL/TLS library makes SSL/TLS as easy as possible. However, if this is your first experience with SSL/TLS, some outside reading on how public key cryptography works is extremely helpful.

When you use SSL/TLS to connect to http://www.amazon.com (for example) with a normal web browser, you will not need to know anything about certificates. This is because Amazon purchased a certificate from Verisign and your browser vendor preinstalled Verisign, as an entity that can sign trusted certificates.

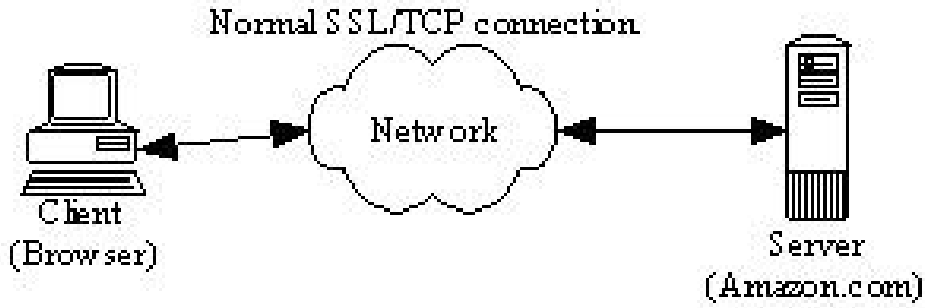


Figure 10.7 A Normal SSL/TLS Connection

Above, is a picture of a perfectly normal TCP or SSL/TLS connection; the client (most often a browser) has connected through the network to a server. If we do not have any entities doing bad things on our network then there is no need for SSL/TLS. However, if the data we are sending is worth stealing, we might have a very different network picture (below).

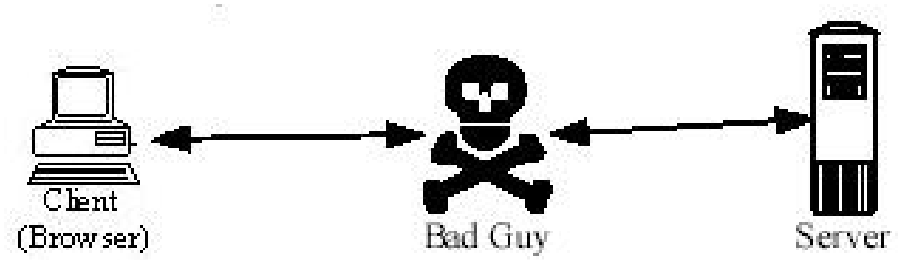


Figure 10.8 An Unsecure Connection

If our connection is routed through a third party (a normal TCP connection), we have no guarantee that this third party is not a "bad guy" trying to steal or modify our data. The SSL/TLS protocol was designed to eliminate this man in the middle attack. SSL/TLS is designed not only to make sure that the data we send over the network is hidden from snooping eyes, but it is also designed to make sure we are connected to the proper server without any "bad guys" in the middle. This verification is done with Public Key (PK) Cryptography and a hierarchy of trust.
Why do we trust a doctor when we go to the emergency room? We trust the doctor we have never met because we trust the hospital to employ qualified doctors. The hospital vouches for his skills and we trust the hospital. His medical school also vouches for him by giving him a diploma with his name and the schools seal or signature. We trust the school, we trust the hospital, and thus we trust the doctor.
SSL/TLS works in a very similar way. When a client connects to the server sends the client a certificate. This certificate has three major elements:

- A name (i.e. who is this server)
- A public key (e.g. think of an open padlock)
- A signature (by a trusted third party that vouches for the name and the public key)

A doctor's diploma is very similar; it also has three major elements: The doctor's name, the type of degree and the medical school (that vouches for the doctor). For example, Bob and George both graduate from Harvard Medical School. They both have Harvard diplomas. However, the diplomas are unique to each doctor. The diplomas are not interchangeable. Bob's diploma would be of no use to George and vice versa. This illustrates the first key point about SSL/TLS.

Key Point #1: Each and every SSL/TLS server must have a unique certificate. Note: You cannot reuse an SSL/TLS server certificate. The certificates are distinguished by the "common name" or "CN" on the certificate.

If you went into a new doctor's office and saw a diploma from Harvard medical school, you would feel comfortable with the doctor's skills. You trust Harvard and Harvard will vouch for this doctor. If instead, the diploma were from the Medical School of Zaire, you would probably be more skeptical. We do not have the same inherent trust of this school as we did with Harvard.

In SSL/TLS as a client, we have to decide who we will trust to sign our certificates. This list of trusted certificate authorities must be explicitly configured into the client. When the web browser or OS was installed on your PC, it probably installed a list of trusted certificate authorities. With the NetBurner SSL/TLS library, we have to explicitly decide whom we are going to trust to sign server certificates. This leads to key point #2.

Key Point #2: An SSL/TLS client must be pre-configured with a list of Certificate Authorities (CAs) that it will trust to sign server certificates. This list can be common across all the clients and does not have to be unique.

10.12.1.1 What do I need to do to make SSL/TLS work?

- You must create or choose a certificate authority. Note: If you create a certificate authority, you will also have to create a set of public/private keys for this authority.
- You must create public/private keys and a certificate for each SSL/TLS server and have the certificate signed by the certificate authority you have chosen.
- You must configure the clients with the list of certificate authorities it should trust.

10.12.1.2 How do I find or create a certificate authority?

Using the medical school example, you can go to Harvard pay lots of $$$ and get a diploma that is trusted by everyone. You can also choose to start your own medical school and issue diplomas. Almost everyone in the world would trust a Harvard diploma. Almost no one will trust a "Bob's Medical School" diploma, unless you spend the time convincing them that it is a quality medical school. In the end, you will likely only be able to convince your family, and then only for non-life threatening needs.

SSL/TLS certificates are a lot like medical schools; you can go and purchase server certificates. To see what a certificate looks like, open your web browser (e.g. Internet Explorer), and connect to https://www.NetBurner.com (notice the s on the end of https). On Internet Explorer's menu, choose File then Properties. Now, click the Certificates button, and look at all the tabs shown in this section.

10.12.1.3 How do I know whom my browser trusts?

On your (Internet Explorer) browser's menu - choose Tools then Internet Options. Open the Content tab, click the Certificates button, and open the Trusted Root Certificate Authorities tab. Add Verisign or Thawte and every browser in the world will trust your certificate and your server.

If you want to save some money and create your own certificate authority then you can do so. However, none of the clients will accept your certificate until you convince them to add "Bob's Certificate Authority" to their list of trusted certificate authorities. If, the users using the embedded SSL/TLS system you are deploying are all in one business entity, then it is relatively simple to add your own certificate authority to the list of trusted authorities. If you are responsible for both the client and server end of the connection, it is even easier; you can configure the clients to accept a single server authority - yours.

SSL/TLS is based on Public Key Cryptography (PK) and a little bit of background on PK is necessary in order to deploy a secure SSL/TLS solution. Public Key Cryptography is different from Symmetric Key Cryptography. In PK, the keys used for encryption are broken into two parts, much like a padlock (the public part) and a key (the private part). If you give someone an open padlock and a steel box, they can put things into the box, close the lid,

and lock the lock. Unless they have the key to the lock, they cannot open the box. They can be confident that if they mail you the box, none of the mailmen along the way can look inside. Only the person who holds the (private) key to the padlock can open the box. For additional information on Public Key Cryptography, please read the Cryptography FAQ (http://isc.faqs.org/faqs/cryptography-faq).

When the SSL/TLS client connects to a server, the server sends back a certificate with a public key (open padlock). This certificate also includes the name of the server and a signature vouching for both the public key and the name. If any part of the certificate is changed, the signature will compute to be invalid.

So, if we have a "bad guy" in the middle, he can watch the padlock going from the server to the client. But, when the client puts his secret information into the box and locks it, the "bad guy" cannot see inside. He only knows that the client sent something in the box to the server. The secrets in the box are safe from the prying eyes of the "bad guy". This safety only exists if the server has done a good job of protecting the private key. If the "bad guy" sneaks into the server room, logs on the server console, and makes a copy of the private key, he can intercept all of the traffic. He can also change the content at will. This leads to key point #3.

Key Point #3: When using Public Key Cryptography (as SSL/TLS does), the system is only as secure as the security of the private key. Since a server needs access to the private key to unlock the data from the client, the private key must exist on the server.

Key Point #3 Corollary: If the private key exists on the server, then the system is only as secure as the physical security of the server. If the server is not physically secure, then someone (i.e. the "bad guy") can attach an emulator or other hardware to the server and read out the private key.

For example, suppose the "bad guy" wants to intercept your credit card number when you send it to Amazon to order a book. We have already shown that he cannot read the data unless he has Amazon's private key. However, he has one other option - he can pretend to be Amazon and offer his own certificate to you, the client. If this certificate is properly signed by a Certificate Authority the client trusts, then client will accept the connection. If any "Certificate Authority" in the list of trusted authorities is compromised, then the system is insecure. If the "bad guy" has the ability to add a new "Certificate Authority" to the client, then he can completely compromise the system. This leads to key point #4.

Key Point #4: If the ability to add a "Certificate Authority" to the client's list of trusted authorities is not secure, then system is not secure.

Key Point #4 Corollary: If the list of trusted "Certificate Authorities" exists on the client, then the system is only as secure as the physical security of the client. If the client is not physically secure, then someone (i.e. the "bad guy") can attach an emulator or other hardware to the client and add a "trusted" authority.

These last two key points imply that it is not possible to build a system that is more secure than the physical security of the device being secured.

Important: All the cryptography in the world will not help if someone can gain access to your computer and hide a bug inside the keyboard; or even easier, add or modify a system file to record your keystrokes and periodically send them over the internet to some nefarious foe.

10.12.1.4 Recommended Reading

For an excellent overview of computer security:

- Secrets and Lies by Bruce Schneier (ISBN 0-471-25311-1)

For a detailed review of cryptography:

- Applied Cryptography by Bruce Schneier (ISBN 0-471-11709-9)

For a detailed description of the SSL/TLS protocol:

- SSL and TLS by Eric Rescorla (ISBN 0-201-61598-3)

For a reference on the math and methods in cryptography (this is a heavy duty book):

- Handbook of Applied Cryptography by Menezes, Oorschot and Vanstone (ISBN 0-8493-8523-7)

10.12.2 Onboard Self Signed Certificate Creation

To help developers tackle the growing burden of implementing and managing security on embedded and IoT devices, NetBurner now offers the ability for onboard self-signed certificate generation. This means that with very little effort (a single line of code, in fact), NetBurner modules acting as a server and running NNDK 3.x can automatically generate a self-signed certificate that can be used in secure SSL/TLS communications.

This feature is designed to allow engineers to have maximum control over the certificates that are used by their devices and can be modified in several different ways. Currently, the default behavior is as follows:
1) If a certificate is manually loaded on the module prior to the initialization of the SSL/TLS server, that certificate will be used, and the device will generate nothing.
2) If a TLS connection is attempted, either through SSL_connect(), SSL_accept(), or StartHttps(), and the device does not have a certificate loaded already, one will be generated automatically.
3) If a certificate has been generated previously and a new certificate is manually loaded on the device, the newly loaded certificate will be used in place of the automatically generated one. This will take place once the device has gone through a power cycle. The automatically-generated certificate will still be available on the module.
4) If `ENABLE_AUTOCERT_REGEN` is defined (found in predef.h), then the module will periodically check to see if the auautomatically-generated cert has expired, and if it has, it will automatically generate a new one. How often it is checked is defined by `AUTO_CERT_GEN_CHECK` (also in predef.h), and defaults to once per minute.

Certificates generated with this functionality are good for one year. These certificates live in non-volatile memory and will survive if the device is powered off. The auto-generated certificate creation system has been extended substantially since it was first introduced. The following examples are available to demonstrate the capabilities of the system, including several examples for advanced users who need more control over how the certificates are generated. The are located in `<nndk_install>\examples\SSL\SslOnboardCertGeneration`:

- `Simple`: Shows how to initialize the system to enable onboard certificate generation.
- `Advanced`: Shows how to manually call SSL_CreateNewSelfSignedCert() with properly formatted alt names.
- `CompiledCa`: Shows how to create a custom self signed cert function that uses a compiled in Certificate Authority to sign the generated certificate.

  When generating the certificate, the random number generator will need to be properly seeded. This is done by analyzing serial and network traffic. Pinging the device or visiting the device's config page from a browser will speed this process up.

  By default, the certificate is generated using ECC with SECP384R1. This can be changed to an RSA key by undefining `ENABLE_ECCKEY_CREATE` in `nburn\libraries\crypto\platform\<platform name>\user_settings.h`, and then rebuilding your application. To specify what curve or RSA key length is used in certificate generation, define `DEFAULT_KEY_TYPE` as one of the values defined by `SslKeyType_t`.

10.12.3 Creating Self-Signed Certificates

The NetBurner SSL/TLS library provides some open source tools for the generation and maintenance of SSL/↩ TLS self-signed keys and certificates. These tools are based on OpenSSL and subject to OpenSSL License. The NetBurner SSL/TLS library code subject to the standard NetBurner License (located by default in `\Nburn\docs`).

- `openssl.exe`: Creates certificates and keys

- compfile: Creates .cpp files from certificates and keys

There is a readme.txt file in \nburn\CreateCerts that provides information on .bat and .sh files used to make the creation process easier. Please refer to the script files if you want more information on the OpenSSL options used to create the certificates and keys. Certificates should use the PEM format.

10.12.3.1 Creating a Certificate Authority (CA) Certificate and Key

This step creates a CA you can use to sign certificates. **Important:** You should only have to do this step once. The key file created in this step should be protected as the security of all your certificates depend on it. You have two choices to protect this key file. **Note:** If you want the key to not be protected by a pass phrase then leave the -des off the genrsa command.

- Open a command prompt in \nburn\CreateCerts\ECDSA or \nburn\CreateCerts\RSA, depending on the type of certificate you want to create.
- Run makeca to create CA.crt and CA.key, which will be used by other script files to sign server and client certificates and keys.
- You will be prompted to answer some questions. If you require more detailed information please refer to the OpenSSL website and documentation.

10.12.3.2 Creating a Server Certificate and Key for Your Device

You will need to create a certificate and key for each device you deploy, and the Common Name in the certificate must match each of your device's name. For example, if I have 2 devices and am using DNS, then I could name them Device1 and Device2, and the certificate would need the CN to match. If I did not have DNS, I could use their IP addresses as the CN. However, if their IP address changes, the certificate would no longer be valid because the CN would not match.

- Open a command prompt in \nburn\CreateCerts\ECDSA or \nburn\CreateCerts\RSA, depending on the type of certificate you want to create.
- Run 'makeserver' to create device.crt and device.key. The script will also invoke compfile to create cert.cpp and key.cpp in case you desire to use a compiled-in cert and key.
- This certificate will be signed with the CA certificate created previously.

10.12.3.3 Converting a Certificate and Key to Code with <tt>compfile</tt>

The compfile utility converts .crt and .key files to C++ source code files that can be built into your application (note this implies that each device needs its own application image). It is already part of in the batch and shell files mentioned previously. This section is included in the event you wish to run it separately against your .crt and .key files.

- Open a command prompt in \nburn\CreateCerts\ECDSA or \nburn\CreateCerts\RSA, depending on the type of certificate you are converting.
- Execute the commands: `compfile device.key comp_key comp_key_len key.cpp` `compfile device.crt comp_cert comp_cert_len cert.cpp`

Note

comp_key, comp_key_len, comp_cert, and comp_cert_len all refer to global variables available in the application that represent the key data, the length of the key data, the cert data, and the length of the cert data respectively.

10.12.3.4 Adding the Module to your Code Set

Take the `key.cpp` and `cert.cpp` files previously created and import it into your project directory. If you are using command line tools, copy it to your project directory and add it to your makefile.

10.12.3.5 Using a Recognized Certificate Authority

If you are going to have your certificates signed by an external entity, they will need a Certificate Request file. **Note:** The common name you enter in this step must match the deployed DNS name or IP Address of the Server it will be used on.

- Open a command prompt/DOS window
- Navigate to the directory that you want to house your device files
- To make a Device Certificate Request file, execute the command (and press the Enter key when finished)↩ : `openssl genrsa -out Server.key 1024 openssl req -new -key Server.key -out Server.csr`
- Send this `Server.csr` to the CA that will create your certificate.

Warning: If you lose the `Server.key` file associated with this particular device, then you will not be able to use the certificate file they send back.

10.12.3.6 Testing Your Certificates

The batch and shell files `checkcert` and `checkkey` can be found in `\nburn\CreateCerts\ECDSA` or `\nburn\CreateCerts\RSA`. For example: `checkcert device.crt` `checkkey device.key`

10.12.4 Creating Certificate Authority Lists

10.12.4.1 Creating a CA List file

To create a CA List that will hold all of the CA certificates you are willing to accept, you can either compile the certificate data into your project, or you can load it dynamically from a flash drive.

To use the compiled in data, create a header file in your project directory that will hold the certificate data. In this file, create a const char array for each certificate that is initialized as a string literal with each corresponding CA certificates' data. Make sure to take note of the length of the array, as it will passed into `SSL_connect()` as function parameters. The example `nburn\examples\SSL\SslClientVerifyPeerBasic` demonstrates this method.

To use an SD card or onchip flash to store the certificate data, please see the example `nburn\examples\SSL\Ssl↩ ClientVerifyPeerEffs` for further guidance. In that example, we initialize the SSL/TLS system with a call to `SslInit()` before we start the HTTPS server so that we might add the certificates with calls to `SSL_Add↩ CertToClientCaList()` for each certificate that we want to load. In this examplewe do this before the calls to `SSL_connect()` so that we can add multiple certificates stored on the SD card or onchip flash.

10.13 SSH Programming Guide

10.13.1 SSH Overview

The Secure Shell Protocol (SSH) is a secure network protocol that has been around since 1995. It was designed to facilitate operating network services over an encrypted connection, and is still widely used today.
SSH has been a part of NetBurner's offerings for quite awhile now, starting with NNDK version 2.0. Up until recently, we utilized Dropbear to provide the SSH capability to our library. As of NNDK 3.3.6, however, we have migrated to wolfSSH. This change provides our library with several benefits. First, it provides better security with an updated list of ciphers compared to the version of Dropbear we were utilizing previously. It also allowed us to provide additional features, such as the capability to support running an SSH client on our devices, where we had been unable to do so before. Finally, because it shares the same underlying library with our SSL/TLS library, wolfCrypt, we were able to reduce the amount of code required to run both libraries simultaneously.

10.13.1.1 Supported Key Types, Sizes, and Formats

The SSH library currently supports both ECC and RSA keys, though we recommend ECC. The default key size for RSA is 2024, but support for a 4096 key can be enabled by defining ENABLE_RSA_4K in `<nndk_install>\nbrtos\include\predef.h`. ECC key sizes can range from 160 to 512.
Both ECC and RSA keys can be presented in either PEM or ASN1 formats.

10.13.1.2 Migrating from NNDK 3.3.5 and earlier to 3.3.6 or later

We've made every effort to make sure that the updates to the SSH library would be backward compatible. However, in some cases change was unavoidable, and in others, developers would be better served to update their applications to use newer versions of existing functions.
When building applications that use the SSH libraries in NBEclipse, the following include paths need to be added to projects:

```
${NNDK_ROOT}/libraries/include/ssh
${NNDK_ROOT}/libraries/include/crypto
```

To do this, first right click on your project in NBEclipse, and select the properties option at the bottom of the menu. The following window will be displayed.

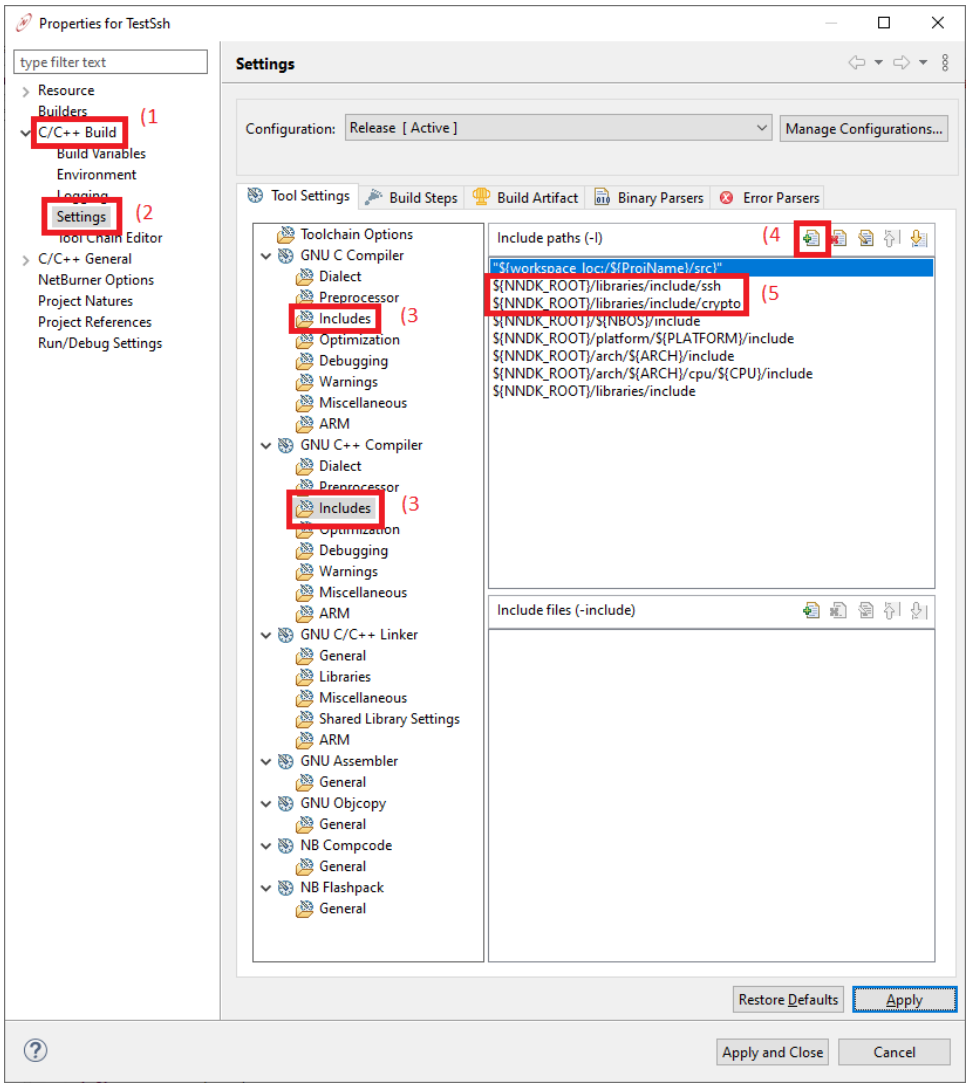


Figure 10.9 Adding the SSH Include Paths

From the left side menu, select "C/C++ Build" (1) followed by "Settings" (2). In the main window, you'll want to select the "Includes" (3) folder under both the "GNU C Compiler" and "GNU C++ Compiler". Once here, hit the icon with the green plus on it (4), and then add the paths so that they are listed above (5). Finally, hit "Apply", and then clean the project. It should find the required files during the build and complete successfully.

10.13.1.2.1 Removed and Updated Functions The following functions have been removed from our API. They were specific to the Dropbear library and no longer make sense to have in our current API.

- `SshConvertDecodedOpenSSLKey` This function converted a PEM formatted SSL key to the Dropbear format. The new API supports both PEM and ASN1 formatted keys, and we no longer support the Dropbear format.
- `SshSetchansessionrequest` This function set a user callback routine for handling server side channel requests. These requests are handled internally by wolfSSH.
- `SshSetTaskPriority` This function set the task priority for the SSH session. Where each SSH session had its own task previously, both SSH and SSL/TLS sessions are now managed by the Crypto↩ Server object. This object runs a task of priority SECURITY_TASK_PRIO, which is defined in `<nndk↩ _install>\nbrtos\include\constants.h`.

The following functions and function pointers have updated versions that we encourage developers to take advantage of.

- `sshUserAuthenticateFn` An updated version of this function pointer, sshUserAuthenticateWithTypeFn, has been added to the API. Instead of just taking a pointer to the username and password as its parameters, it now additionally takes a parameter of AuthType that specifies if the authentication value passed in is a password or a key.
- `SshSetUserAuthenticate` This function should be replaced with SshSetUserAuthenticateWithType(), and takes as it's parameter a function pointer of type sshUserAuthenticateWithTypeFn.
- `SshGetUserAuthenticate` This function should be replaced with SshGetUserAuthenticateWithType(), and returns a function pointer of type sshUserAuthenticateWithTypeFn, if it has been set with a call to SshSetUserAuthenticateWithType().

10.13.1.3 Onboard Generated Keys

As is the case with our updated SSL/TLS library, SSH can now take advantage of onboard generated keys. This means that SSH keys can be automatically generated on the device if required. The mechanism for this is exactly the same, and in fact, the SSH key is the same key that is generated as part of generating an SSL/TLS certificate. To understand how to use the functionality, we encourage developers to look at the appropriate examples, found in `<nndk_install>\examples\SSL\SslOnboardCertGeneration`.
Auto-regeneration of this key can be enabled by defining `ENABLE_AUTOCERT_REGEN`, found in `<nndk_install>\nbrtos\include\predef.h`.

10.13.1.4 Order of Key Use

There are several methods in which a key could be installed on the device. In order of consideration during runtime, these are:

- Set a function callback with SshSetUserGetKey() that will return the server key via the passed in parameters.
- Using a compiled in key, and overriding the weak function GetPrivateKeyPEM().
- Using an onboard generated key.

10.13.1.5 User Authorization

To assist with storing and managing user authentication data, we've provided a simple class, UserAuthManager. This class provided the methods required to store, retrieve, and compare user credentials for authentication and authorization purposes. Two callbacks signatures are provided, LoadAuthRecordsFn and SaveAuthRecordsFn, that can be used to load and save user authorization into the UserAuthManager object. This allows the developer to use any type of storage device or location to save the data.
Passwords and keys are hashed when added to the manager, so there is no possibility for leaking plain text data that could potentially compromise the security of the system. Our provided example, `sshServerUserAuth`, demonstrates this using the devices User Param space to save the data as a JSON blob.

10.14 Web Server

Topic Links:

- Basic Web Server
- Dynamic Web Content

10.14.1 Basic Web Server

The Application Wizard creates a main.cpp file, and also a directory in your project named "html" with a file named index.html:

```
<html>
<body>
The main page for the AppWizard project.
</body>
```

```
</html>
```

In NBEclipse you can open a HTML file in text edit mode by right-clicking on the file and selecting the text editor option:

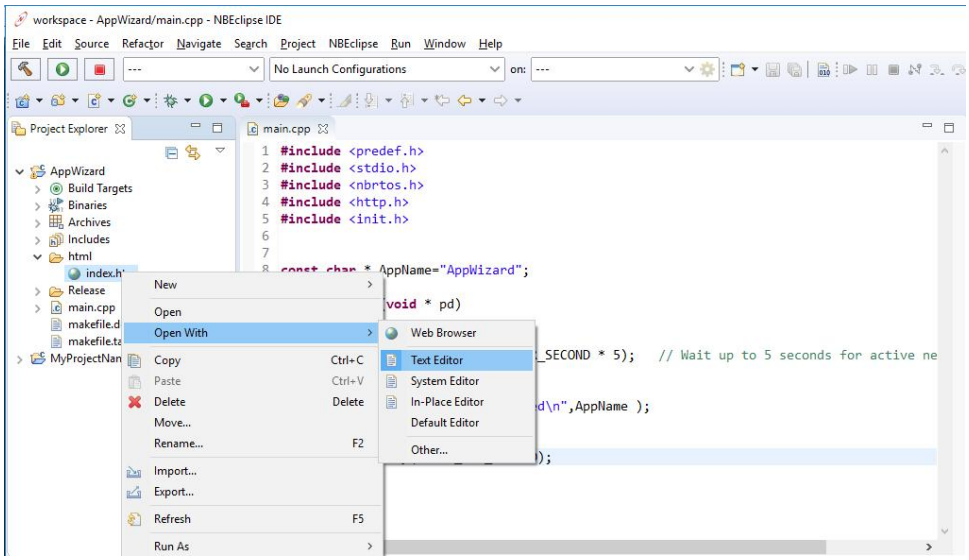


Figure 10.10 Open with text editor

The project html folder is where you can all the files you need for your application, including images, javascript, html files, etc. You can also add folders. For example adding a folder for images is very common. For example, if you created a folder in the html directory named images and added the file logo.jpg, you could add it to your index page by adding the img tag:

```
<html>
<body>
The main page for the AppWizard project.
<img src = images/logo.jpg>
</body>
</html>
```

10.14.2 Dynamic Web Content

Dynamic web content (web content created at run-time) can be created using the following NetBurner HTML tags:

- A C++ callback function CPPCALL:
  `<!--CPPCALL YourCppFunction -->`

  When the web server delivers the web page it will call your C++ function.

- The VARIABLE tag:
  `<!--VARIABLE YourVariable -->`

  . When the web server delivery the web page it will insert the specified variable.

- A C callback function FUNCTIONCALL. This tag is included for those porting from older tool sets. FUNCTIONCALL is a C only tag, and is deprecated.

10.14.2.1 Dynamic Content Using the CPPCALL Tag

To add a C++ callback we can add a function to main.cpp such as:

```
void webHelloWorld( int sock, PCSTR url )
{
    writestring( sock, "Hello World" );
}
```

And modify index.html to add the tag:

```
<html>
<body>
The main page for the AppWizard project.
<!--CPPCALL webHelloWorld -->
</body>
</html>
```

When a web client requests the index.html page, the web server will begin steaming the static content. When it gets to the CPPCALL, it will call the webHelloWorld() function, which will send "Hello World" to the client using the open TCP socket. When the function returns, the web server will continue sending the index.html web page. A web page can have any number of CPPCALL and VARIABLE tags. The parameters passed to the CPPCALL function are the open TCP socket handle, and a pointer to a constant string containing the entire URL of the client web browser. In this way your function can parse the URL if that is important to your application. For example, anything after a '?' character is application specific.

10.14.2.2 Displaying Variables Using the VARIABLE Tag

Variables in an application can be displayed on a web page using the VARIABLE tag. This can be useful for displaying information such as time, an IP address, temperature, etc. The format of the tag is:

```
<!--VARIABLE <name> -->
```

Where "name" is the name of the application variable or an expression. For example, the system time tick variable `TimeTick` can be displayed with

```
<!--VARIABLE TimeTick -->
```

Or you can display the time in seconds with the equation:

```
<!--VARIABLE TimeTick/TICKS_PER_SECOND -->
```

The VARIABLE tag is processed during the compilation of the application by parsing the text between `<!--VARIABLE` and the trailing `-->` and converting it into a function call such as: `WriteHtmlVariable( fd, TimeTick/TICKS_PER_SECOND );` In this function `fd` is a file descriptor to the current TCP connection to the web client.

Variable types are handled with C++, but you do not need to know anything about C++ to use this feature.
The parameter types are defined by the function definitions located in `\nburn\include\htmlfiles.h`:

```
void WriteHtmlVariable(int fd, char c);
void WriteHtmlVariable(int fd, int i);
void WriteHtmlVariable(int fd, short i);
void WriteHtmlVariable(int fd, long i);
void WriteHtmlVariable(int fd, BYTE b);
void WriteHtmlVariable(int fd, WORD w);
void WriteHtmlVariable(int fd, unsigned long dw);
void WriteHtmlVariable(int fd, const char *);
void WriteHtmlVariable(int fd, MACADR ip);
```

In addition, we have included a class named `IPCAST()` that takes a 32-bit value and converts it into an IP address format (e.g. 192.168.1.2). This example will display the IP address in dotted notation, rather than a 32-bit integer:

```
IP address: <!--VARIABLE IPCAST(IpAddress) !-->
```

10.14.2.3 Linking to Variables Using the INCLUDE Tag and htmlvar.h Header File

When an application with web content is built by the NetBurner tools, a file named htmldata.cpp is created. When using the VARIABLE tag for dynamic data, the application must be able to link to the variable. This can be achieved in either of two ways:

- Create a header file in your project named `htmlvar.h`.
- Use the INCLUDE HTML tag in the .html folder.

  An example of a htmlvar.h file to display a variable named Temperature, and a VARIABLE function callback named webMyVarFunction() to display an integer:

```
#ifndef HTMLVARS_H_
#define HTMLVARS_H_

#include <startnet.h>

extern int Temperature;

const char *webMyVarFunction(int fd, int v);

#endif // HTMLVARS_H_
```

10.14.2.4 Function Callbacks With Variables Using the VARIABLE Tag

If you need to use a function that can take a parameter, the CPPCALL tag will not work because the parameters are fixed as the socket fd and URL. However, the VARIABLE tag be used for both the function call and variable parameter. The htmlvar.h include file must specify the function definition in one of the formats described in the previous section. the format below. For example, to pass an integer variable webMyInteger in a callback function named webMyFunction:

```
const char * webMyVarFunction(int fd, int webMyInteger);
```

The HTML file calls the function with:

```
<!--VARIABLE MyFunction(fd, webMyInteger) -->
```

When the application is compiled the resultant function call will be:

```
WriteHtmlVariable( fd, MyFunction(fd, webMyInteger) );
```

In your .cpp source code the function could look something like this:

```
const char * webMyVarFunction(int fd, int v)
{
   char buffer[255];
   siprintf( buffer, "MyFunction() was called with v = %d\r\n", v );
   writestring( fd, buffer );

   return "\0";     // Return a const char * of zero length so it will not print to the HTML page
}
```

10.14.2.5 Extending the VARIABLE tag for User Defined Types

The VARIABLE functionality can be extended to support user defined types, such as a user defined structure or class. Lets say you have a structure you want to display on a web page called MySTruct:

```
struct  my_struct {
   int      i;
   char     buf[80];
   uint32_t dVal;
} MY_STRUCT;

MY_STRUCT MyStruct;
```

In your include file add the function definition: `void WriteHtmlVariable(int fd, MY_STRUCT MyStruct);` You then need to implement the function to display the data. For example:

```
void WriteHtmlVariable(int fd, MY_STRUCT MyStruct)
{
    fdprintf(fd, "int: %d<br>", i);
    fdprintf(fd, "buf: %s<br>", buf);
    fdprintf(fd, "uint32_t: %ld<br>", dVal);
}
```

You can then display it on the web page with the VARIABLE tag:

```
<!--VARIABLE MyStruct -->
```

Chapter 11

System Diagnostics

11.1 Introduction

System Diagnostics can be viewed from a device's Configuration Server web page (port 20034) by calling the EnableSystemDiagnostics() function in your application. Diagnostic information includes items such as system definitions, variables, TCP stack status, and RTOS state information.

11.2 Configuration Server Page with Diagnostics Enabled

When system diagnostics are enabled, an additional button will appear on the configuration server web page in the lower right corner:

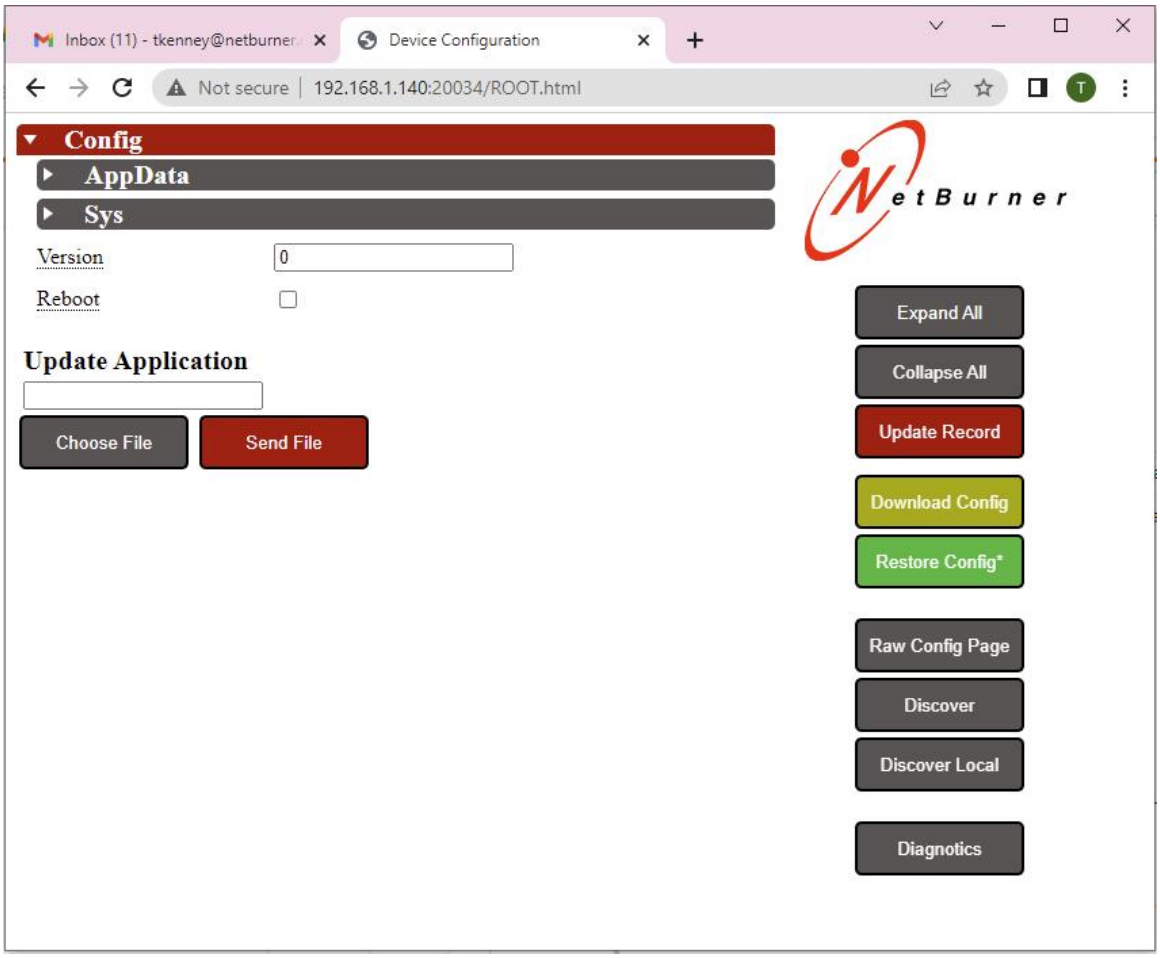


Figure 11.1 Configuration Server With Diagnostics Button

11.3 System Configuration Web Page Display

Clicking on the Diagnostics button will display system information. An example of the raw data is shown below:


```
{
"Active":0 day 01:18:04,
"Buffers":401,
"Config":{"ConfigUsed":749,
"ConfigMaxSpaceAvailible":10240
}
,
"EXtraIO":120,
"FreeMem":7577316,
"My Int":4677,
"My String":"My String",
"NNDKVer":"3.3.8",
"System Macros":{
"_DEBUG":"not defined",
"TICKS_PER_SECOND":20,
"TICK_IRQ_LEVEL":5,
"SERIAL_IRQ_LEVEL":3,
"TCP_SOCKET_OFFSET":5,
"TCP_SOCKET_STRUCTS":128,
"OS_MAX_IRQ":"not defined",
"NBRTOS_STACKCHECK":"not defined",
"NBRTOS_STACKOVERFLOW":"not defined",
"NBRTOS_TASKLIST":"not defined",
"NBRTOS_TASK_LOG":"not defined",
"NBRTOS_TIME":"not defined",
"BUFFER_DIAG":"not defined",
"BUFFER_DIAG_LOG":"not defined",
```

```
"BUFFER_SANITY":"not defined",
"_DEBUG_PRINT":"not defined",
"ENABLE_SMARTTRAP":"defined",
"MULTIHOME":"not defined",
"IPV6":"defined",
"IPV6_COUNTERS":"defined",
"AUTOIP":"defined",
"ALLOW_CUSTOM_NET_DO_RX":"not defined",
"TCP_NOCOPY_TX":"defined",
"UDP_FRAGMENTS":"not defined",
"GATHER_RANDOM":"defined",
"NB_SSL_SUPPORTED":"defined",
"SSL_V3_DISABLED":"not defined",
"SSL_TLS_SUPPORT":"defined",
"WEB_CLIENT_SSL_SUPPORT":"defined",
"SSL_DEFAULT_MAX_SESSION_AGE_TICKS":"defined",
"TLS_CACHE_PEER_CERT_VALIDATIONS":"defined",
"NB_SSL_CLIENT_CERTIFICATE_CHECKING_ENABLED":"not defined",
"NB_SSH_SUPPORTED":"not defined",
"NB_ENABLE_USER_QSPI":"defined",
"SUPPORT_LEGACY_FIND":"defined",
"NO_SYMETRIC_ROUTING":"not defined",
"FEC_ISR_ERROR_COUNTERS":"not defined"
},
"TCP":{"FreeSockets":124,
"Sockets":{"Socket129":{"State":"ESTABLISED","Flags":"0x81","myIP":"192.168.1.140","theirIP":"192.168.1.26","myPort":20034,"th
      "myWindow":871968320,"TxBuffDepth":0, "RxBuffDepth": 0,"lastRxTime":93681},
"Socket130":{"State":"ESTABLISED","Flags":"0x80","myIP":"192.168.1.140","theirIP":"192.168.1.26","myPort":20034,"theirPort":59
      "myWindow":2744612803,"TxBuffDepth":0, "RxBuffDepth": 0,"lastRxTime":93681},
"Socket131":{"State":"LISTEN","Flags":"0x00","myIP":"::","theirIP":"::","myPort":80,"theirPort":0,"max_listen":5,"cur_listen"
      "myWindow":0,"TxBuffDepth":0, "RxBuffDepth": 0,"lastRxTime":0},
"Socket132":{"State":"LISTEN","Flags":"0x00","myIP":"::","theirIP":"::","myPort":20034,"theirPort":0,"max_listen":5,"cur_liste
      "myWindow":0,"TxBuffDepth":0, "RxBuffDepth": 0,"lastRxTime":93681}}},
"Tasks":{
"Task38":{"Name":"Enet", "Prio":38,"State":"Fifo","Time":40,
      "Stack":["70005b26","7002d6d4","70005f30","7000ed1c","000004a4"]},
"Task44":{"Name":"Config Server", "Prio":44,"State":"Running","Time":600,
      "Stack":["7002d54a","700061be","70023f82","700249ec","70020de4","7002178e","700218e8","7003143e","700213e6","000004a4"]
"Task45":{"Name":"HTTP", "Prio":45,"State":"Semaphore","Time":19,
      "Stack":["70005b26","7002d6d4","70005e48","7000465a","70004694","700245e4","000004a4"]},
"Task50":{"Name":"Main", "Prio":50,"State":"Timer","Time":20,
      "Stack":["70005b26","7002d6d4","70005b96","7001e05a","000004a4"]},
"Task63":{"Name":"Idle", "Prio":63,"State":"Ready","Time":"FOREVER", "Stack":["7002c424","000004a4"]}
},
"Ticks":93681,
"UpTime":4684
}
```


If you are using Firefox or have the Pretty Print extension in Chrome, the JSON data will be formatted and much easier to read. If the output you see looks identical to the previous raw information, it means the browser you are using does not support JSON data formatting. The image below shows a snippet of a formatted file displayed in Firefox:

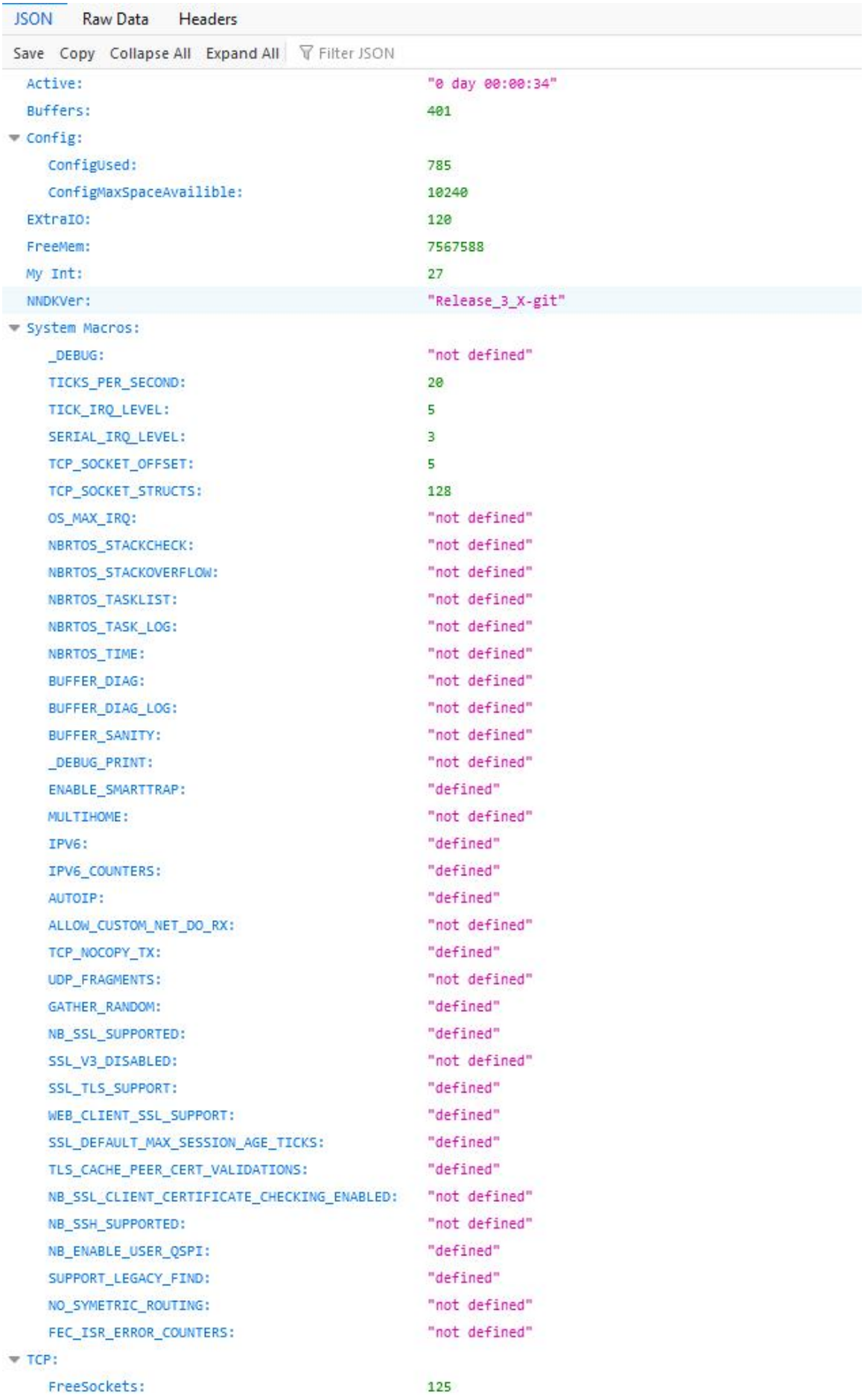


Figure 11.2 Formatted JSON displayed with Firefox

11.4 Adding Your Own Diagnostic Information

You can add your own diagnostic information using the MyVarMon() functions. Please refer to the System Diagnostics example for more details. In the example output in the previous section, the variables below were added to the standard system output:

```
"My Int":4677,
"My String":"My String",
```

The application code to implement the display from the example is:

```
// Variables to monitor
int myInt;
char myString[80];

// Declarations to add variables to Diagnostic Web page
static DiagVarMon MyIntMon("My Int", myInt);
static DiagVarMon MyStrMon("My String", myString);
```

Chapter 12

EFFS Multiple Partitions

This example program demonstrates how to create a second partition in the Application Flash, so that a guaranteed amount can be reserved for NNDK System usage, while still allowing general Application usage as well.

Chapter 13

Example Applications

13.1 Example Applications Main Page

Included in your NNDK software installation folder are over 200 examples to aid you in your product development and learning the NetBurner platform, demonstrating functionality such as:

- Web pages, Web servers and Web clients
- TCP client and server
- TLS client and server, certificate, auto cert generation and management
- UDP client and server
- Device configuration
- Real-Time operating system such as semaphore, tasks, queues and critical sections
- JSON
- File system
- VLAN
- Web sockets
- Hardware board locking to secure application code
- Hardware peripherals such as I2C, SPI, UART, USART, 1-Wire, GPIO, ADC, DAC, PWM, Timers
- PPP
- FTP
- SSH
- MQTT
- ACME certificate support

These are just some of the extensive topics covered. There are a few ways to browse the examples:

1. The HTML and PDF documents
2. Browsing the folder structure in `nburn\examples`
3. When using NBEclipse, you can browse and import examples

There are two main categories of examples:

1. The `\nburn\examples` folder contains examples that can be run on all platforms.

2. The \nburn\examples\PlatformSpecific folder contains examples that are specific to a NetBurner product, such as hardware peripherals: I2C, SPI, 1-Wire, ADC, DAC, etc.

Links to the example sections are below, or you can use the tree control structure in the online docs.

- AES
- Board Lock
- Clear Configuration Flash
- Clear User Parameter Flash
- Configuration
- DNS Device Name
- DHCP
- DNS Client
- Embedded Flash File System (EFFS)
- Ethernet
- Exception Try/Catch
- External IRQ
- Extra FD Circular Buffer
- FileDescriptor
- FTP
- GDB Debugger
- General Purpose I/O
- IP Address Object (IPADDR)
- IPv6
- JSON
- malloc
- MQTT Examples
- Multicast
- Multihome
- NB Approve Shutdown
- NBString Class
- NetBios Name
- Network Time Server (NTP) Client
- NTP & Real-Time Clock
- Overload Directory & System Files
- PLATFORM SPECIFIC
- PPP
- Profiler

- RTOS
- Save to User Parameter Flash
- Serial
- Serial Webserver
- SHA1 Digest
- Show Network Interfaces
- SOCKS5 Client
- SPI
- SSH
- SSL/TLS
- Stack Protection
- Syslog
- System Diagnostics
- TCP
- Telnet Command
- TFTP - Trivial File Transfer Protocol
- Time Functions
- Timers
- UDP
- VLAN
- Web Server
- Web Client
- Web Sockets
- Wifi

13.2 AES

A basic description of AES can be obtained from Wikipedia, and if you are new to AES, you should definitely do some research to understand the implementation. The description below is from Wikipedia:
In cryptography, a block cipher is a symmetric key cipher which operates on fixed-length groups of bits, termed blocks, with an unvarying transformation. When encrypting, a block cipher might take a (for example) 16-bit block of plaintext as input, and output a corresponding 16-bit block of ciphertext. The exact transformation is controlled using a second input - the secret key. Decryption is similar: the decryption algorithm takes, in this example, a 16-bit block of ciphertext together with the secret key, and yields the original 16-bit block of plaintext.
To encrypt messages longer than the block size, a mode of operation is used. Block ciphers can be contrasted with stream ciphers; a stream cipher operates on individual digits one at a time, and the transformation varies during the encryption. The distinction between the two types is not always clear-cut: a block cipher, when used in certain modes of operation, acts effectively as a stream cipher.
This example demonstrates how to use the AES key, encryption, and decryption function calls on a 16 byte block of data. Since AES is a block cipher, all data must be encrypted in 16 byte blocks. If you have more than 16 bytes to transfer, you must break the data into 16 byte blocks, encrypt them (you can use the same key for all blocks), send them, then decrypt them in 16 byte blocks on the other end.

13.3 Board Lock

This set of examples implements a scheme to lock an application to a specific device (board). The application will only run on a device with a specific MAC address. The implementation uses a secret message that is run through a MD5 hash function. The secret message is padded to a multiple of 64 bytes, and the MD5 context is saved.
The board is then signed by combining the partial MD5 context with the board's MAC address to generate a 16 byte digest specific to both the board and secret message. The 16 byte digest is then stored in flash memory. In this example it is stored in the User Parameters area, but you can choose a different location if you wish.
To verify the application is authorized to run on the device, you recompute the digest and compare it to the stored value.
Procedure to Create a Secret Message:

- Go to the keyblob project.
- Edit the text: `const char * YourSecretSigningText = "This should be your company secret message";`
- Compile and run the application on your NetBurner device.
- Capture the last message, your keyblob should be :

```
const MD5_CTX YourCompanySecret = {{2106921824u,3945495657u,2391356351u,2780313164u},{512u,0u},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};
```

  -Be sure to save the message someplace safe. This is your company's secret message.
- Copy this message/text to the signboard project.
- Compile and run the signboard project on the target board to be permitted/locked. This application computes and stores a signature in UserParam space.
- Now copy the company secret into the checkboard project `main.cpp`.
- Compile and run the application. It will check to see if the application is authorized to run on the device.

Things to modify or enhance:

- The sign function and check function do not need to be in separate programs. They could be in the same program with the sign function hidden by some secret command.
- In most real or significant apps your code will want to use the UserParam space for storing additional information,. so you need to modify the sign and check functions to store and retrieve the 16 byte digest from your storage structures.
- Key Blob
- Sign Board
- Check Lock

13.3.1 Key Blob

Create a MD5 context to save a a key to lock an application to a device.

13.3.2 Sign Board

Sign an application to lock it to a specific device (board) using a key and the MAC address of the device.

13.3.3 Check Lock

Check to verify a locked device is authorized to run the application.

13.4 Clear Configuration Flash

Simple program used to clear the NetBurner 3.0 configuration flash area. Once the flash area is erased, resetting the NetBurner device will re-initialize the configuration parameters. Note: on 5441x based devices this program will not clear the configuration parameters used by the Alternate Boot Monitor.

13.5 Clear User Parameter Flash

Small program to clear the User Parameter area in flash memory. In practice, your application should have the ability to reset any non-volatile settings, rather than a full erase. This utility is useful if switching between examples/applications that use User Parameter storage in different ways, without having to create code to do a default reset.

13.6 Configuration

Examples to demonstrate how to:

- Customize the configuration server
- Set configuration parameters via web page and through the application code

Examples:

- Application Data
- Config and System Params
- Config Web Interfaces
- Detached Config Object

13.6.1 Application Data

The examples in this directory show how to use the configuration system to create, modify, and store persistent data values that can be used in user applications.

- BasicConfigVariables - Shows how to create, modify and store basic config variable types with the config system
- ConfigClass - Expands on BasicConfigVariables, and shows how to create new class types that inherit from base config objects, and how they can be combined together to create complex objects
- NestingConfigObjects - Expands on ConfigClass, and shows how config objects can be grouped together to create a nested hierarchy of data

Examples:

- Basic Config Variables
- Config Class
- Nesting Config Objects

13.6.2 Basic Config Variables

The config system stores system-defined boot and interface information, such as default BaudRate and current IP settings, and also provides a system for the user to define and store their own persistent variables.
In this example, we've created some very basic config objects to represent the basic data types that are created when the device first boots, and whose values persist after the device has been power cycled. The currently available config objects that can be used are as follows:

- `config_obj` // base object, used as a container for other objects

- config_bool // bool
- config_int // int
- config_double // double
- config_string // string
- config_chooser // multiple choice option
- config_pass // password
- config_IPADDR4 // IP4 Address
- config_IPADDR6 // IP6 Address
- config_MACADR // MAC Address

13.6.3 Config Class

This program shows the basics of using NetBurner's config system. The config system stores system-defined boot and interface information, such as default BaudRate and current IP settings, and also provides a system for the user to define and store their own persistent variables.
In this example, which expands on the BasicConfigVariable example, we create some custom classes that inherit from an empty config object type. We then use these classes to group additional config data together. The objects of these types are created when the device first boots, and their values persist after the device has been power cycled. The currently available config objects that can be used are as follows:

- config_obj // base object, used as a container for other objects
- config_bool // bool
- config_int // int
- config_double // double
- config_string // string
- config_chooser // multiple choice option
- config_pass // password
- config_IPADDR4 // IP4 Address
- config_IPADDR6 // IP6 Address
- config_MACADR // MAC Address

13.6.4 Nesting Config Objects

This program shows the basics of using NetBurner's config system, specifically, how config objects can be nested inside each other. The config system stores boot and interface information, such as default BaudRate and current IP settings, and also provides a system for the user to define and store their own persistent variables.
In this example, we have expanded on the ConfigClass example, and created an additional config class that is used to nest several instances of our previously defined class. This nesting allows for control over the structure and hierarchy of the data when it is saved in the flash as a JSON blob. The currently available config objects that can be used are as follows:

- config_obj // base object, used as a container for other objects
- config_bool // bool
- config_int // int
- config_double // double

- config_string // string
- config_chooser // multiple choice option
- config_pass // password
- config_IPADDR4 // IP4 Address
- config_IPADDR6 // IP6 Address
- config_MACADR // MAC Address

13.6.5 Config and System Params

The examples in this directory show how to access and modify system values through the configuration system.

- ShowConfig - Shows how to view the entire JSON config blob
- ModifyBoot - Shows how to view and modify the Boot system values through the global config variable monitor_config
- ModifyEthernet - Shows how to view and modify the interface system values (default is typically Ethernet0) through the config system

Examples

- Modify Boot
- Modify Ethernet Interface
- Show Configuration

13.6.6 Modify Boot

The config system stores system-defined boot and interface information, such as default BaudRate and current IP settings, and also provides a system for the user to define and store their own persistent variables.
This program shows how to view and modify the values of the "Boot" config record.

13.6.7 Modify Ethernet Interface

The config system stores boot and interface information, such as default BaudRate and current IP settings, and also provides a system for the user to define and store their own persistent variables.
This program shows how to view and modify the values of the Ethernet interface in the config record.

13.6.8 Show Configuration

The configuration system stores settings controlling how the device boots, as well as network and serial interface settings, in a JSON format. This example demonstrates how to:

- Read the system configuration and display on the serial port
- Display the amount of memory used by the configuration settings and total available space

13.6.9 Config Web Interfaces

The system configuration web page interface is located on network port number 20034. These example demonstrate how to provide the same configuration interface as part of the application web server (default port numbers are 80 or 443 for SSL/TLS), so that the configuration web pages are consistent with the rest of the application web interface.
Examples:

- Basic Web Config
- Basic Web Config with SSL/TLS

- Config Tags
- Custom Web Config
- Custom Web System Config

13.6.10 Basic Web Config

The system configuration web page interface is located on network port number 20034. This example demonstrates how to provide the same configuration interface as part of the application web server on port 80 (or 443 for SSL/↩ TLS), so that the configuration web pages are consistent with the rest of the application web interface.
The web page is dynamically generated, and provides access to view and/or modify boot and interface settings, such as default serial port baud rates and IP addresses, as well as application specific data.
The code used to generate and process the web form is in `html/index.html`. This example does not use any external JavaScript libraries, and all of the form generation code is located at the bottom of index.html. For a more advanced example that utilized Bootstrap and jQuery, please see the CustomWebConfig example.
Currently available config objects that can be used are as follows:

- `config_obj` // base object, used as a container for other objects
- `config_bool` // boolean
- `config_int` // integer
- `config_double` // double float
- `config_string` // string
- `config_chooser` // multiple choice option
- `config_pass` // password
- `config_IPADDR4` // IP4 Address
- `config_IPADDR6` // IP6 Address
- `config_MACADR` // MAC Address

Note that because we are only processing the config object, we are able to avoid writing a custom POST handler that is used in the other examples. If you include more to your web interface than is provided here, that will need to be incorporated as well.

13.6.11 Basic Web Config with SSL/TLS

This example is identical to the BasicWebConfig example with the addition of SSL/TLS. Please refer to that documentation for a description of how the configuration server operates.
Certificates can be handled a number of ways: auto-generated, self-signed, compiled into the application, or uploaded to a file system. To simplify things, this example provides files for a compiled certificate and key, named ServerCert.cpp and ServerKey.cpp.
To generate your own self-signed certificates, use the script files in the `\nburn\CreateCerts` folder:

- makeca.bat to generate a Certificate Authority certificate and key
- makeserver.bat to generate the HTTPS server certificate and key

For the common name (CN), use the IP address of your device, then copy the generated .cpp files to your project's src folder.
Please refer to the `\nburn\examples\ssl` folder for more advanced SSL/TLS certificate handling.

13.6.12 Config Tags

This program demonstrates the use of the following dynamic web content tags that can be used to display and modify configuration variables:

- CONFIGVALUE
- CONFIGINPUT
- CONFIGTABLE

The example shows how to just get the raw data from the board, as well as how it can be styled using Bootstrap and CSS. While VARIABLE and CPPCALL HTML tags can be used to present the same information, the CONFIG tags simplify things by combining the data with HTML code to create input fields, tables and value displays.

13.6.13 Custom Web Config

This example is an advanced version BasicWebConfig example that implements the content with JavaScript and jQuery. It also demonstrates how to store and display custom application data in the AppData section.
Configuration objects are created when the device first boots, then they are stored so they are persist after the device has been power cycled. The web interface adds custom styling beyond the basic server presentation to view and set the values. The currently available configuration objects are:

- `config_obj` // base object, used as a container for other objects
- `config_bool` // bool
- `config_int` // int
- `config_double` // double
- `config_string` // string
- `config_chooser` // multiple choice option
- `config_pass` // password
- `config_IPADDR4` // IP4 Address
- `config_IPADDR6` // IP6 Address
- `config_MACADR` // MAC Address

Note that because we are only processing the config object, we are able to avoid writing a custom POST handler that is used in the other examples .If you include more to your web interface than is provided here, that will need to be incorporated as well.

13.6.14 Custom Web System Config

The majority of examples for creating a custom configuration interface create a configuration page in the application code so that it will be part of the application's web interface. This enables the configuration interface to be consistent with the rest of the web pages. In contrast, this example will demonstrate how to change the device's system configuration server that appears on network port number 20034. The change can be as simple as branding with your logo, or modification to the web page content.
You can combine both the custom application and custom system configuration to brand your product
The logo and html files are converted to .cpp files and compiled into the application. The files involved are located in `\nburn\nbrtos\source`:

- LOGO.gif, the logo displayed on the configuraiton page.
- ROOT.html, the javascript root configuration page.
- RAW.html, Low level HTML only page with links to configuration and the `UI.html` page that displays the entire JSON object.
- UI.html, displays the entire JSON object.

To customize, follow the overload procedure in which you copy and modify the desired files into your project's overload directory. For example, to change just the logo, create a folder in the overload directory where the file is located: `\nbrots\source`, and copy your LOGO.gif file there. The majority of customizations will only need to change the logo, but the html overrides are there to suppor any type of customizaton an application requres. A sample LOGO.gif file is provided for you in this application's src folder. Copy this gif to the `override\nbrtos\source` folder to see the modification.
Running the Example: The example is based on SimpleHtml and provides a basic web page on port 80. Viewing the system configuration page on port 20034 will show the custom modifications. A LOGO.gif file is already provided to demonstrate the use of a custom logo, so you can build and run the example as-is before adding your own customizations.

13.6.15 Detached Config Object

This example demonstrates how to create JSON configuration objects (data or structures) in a "detached" mode, meaning they do not appear in the web page configuration screen, nor are they saved, but are exposed via individual web pages. In Detached mode they are created at boot time and can be modified and used by the application.
The example will:

- Create a number of configuration object classes
- Use the classes to create Detached configuration objects
- For reference, shows how they would be exposed in the normal configuration tree
- Expose those objects as web pages containing the JSON information

For example, if your device IP address is 10.1.1.10, opening a web browser and typing "10.1.1.10/set.json" will display that object and it's data:

```
ctrl_set
  Hset
      D   0
      I   0.0001
      P   1
  Vset
      D   0
      I   0.0001
      P   1
  rev 0
```

13.7 DNS Device Name

The device name is stored in the device's flash configuration. There are two ways to give your device a name:

1. Use the configuration web interface, which is your device's `<device ip address>:20034`.
2. You can change the configuration flash setting in your application. You will then need to reboot the device for the name to take effect.

This example demonstrates how to change the device name in an application.
Additional information on DNS naming
When a network device (such as the NetBurner module) requests an IP address from a DHCP Server, the device sends it MAC address and an optional device name (this is supported by your NetBurner device). The DHCP Server supplies the dynamic IP address and maintains a table containing the MAC address, IP address and device name for each device that has accepted a DHCP lease.
The purpose of a DNS server is to convert a name into an IP address. To do this it must get the information from your DHCP Server. On windows server platforms that run both a DHCP server and DNS server, this can be configured in a dialog box. On Linux/Unix it may be more complicated. Basically, every time the DHCP server gives out a new dynamic IP address it updates the DNS server. If you are not using DHCP, and are assigning static IP addresses, then you must update the DNS server with the entries manually (i.e. name and IP address).
If you do not have a DNS server, you can modify the lmhosts file on your PC to add the IP address and name of each network device you want to address by name. You would need to do this on every PC that wants to use the name.
Windows has a proprietary protocol called WINS (only Microsoft products can use it). Windows machines can have a name, and the protocol enables windows machines to talk to one another. This is achieved by having one windows PC maintain the table of MAC address, IP address and names.

13.8 DHCP

DHCP Examples:

- DHCP Client - Change IP Address
- DHCP Client - Test DHCP
- DHCP Server (DHCPD)
- DHCP Client - Change IP Address Via Webpage

13.8.1 DHCP Client - Change IP Address

Demonstrate how to view IP address and change between DHCP and static IPv4 IP settings.
To change from DHCP to Static:

- Read the Static settings from the configuration record
- Set address mode to "Static"
- save to the configuration flash
- Option to reboot or change runtime settings

To change from Static to DHCP:

- You can leeave the static settings intact if you wish, it will not affect DHCP
- Set address mode to "DHCP" or "DHCP w Fallback"
- Save to the configuration flash
- Option to reboot or change runtime settings

13.8.2 DHCP Client - Test DHCP

13.8.2.1 Example of advanced DHCP functionality

Most applications can simply use the GetDhcpAddress() function call to automatically handle obtaining a DHCP address and configuration. If you need finer control, you can create a DHCP Object and manage the DHCP functions from within your application. This example shows how you can create a DHCP object and use a pointer to the object to monitor and control DHCP services.
It also will show you the internal workings of DHCP by displaying UDP packet information and DHCP variables on stdio, which is UART0 by default (view with MTTTY). It is not the intent of the example to be a tutorial on DHCP. For detailed DHCP information please refer to RFC 2131.
For the purposes of education/demonstration, this application will access the C++ private class variables of a DHCP object. This is NOT required for end applications, it is just used here for education.
To obtain access to private class variables in the DHCP object, you will need to enable debugging in the `\nburn\system\dhcpc.cpp` file by removing the comments from the following line:
`#define DHCP_DEBUG 1`

Warning

> COMMENT OUT THE `#define DHCP_DEBUG 1` when you are done with this example, it should not be enabled in normal applications.

This example will:

- Create a DHCP object global variable
- Start DHCP and monitor a semaphore for completion
- Provide an interactive menu to start and stop dhcp services, modify the system timer, and display dhcp status.

13.8.3 DHCP Server (DHCPD)

Demonstrates how to create a DHCP Server on your NetBurner device

13.8.4 DHCP Client - Change IP Address Via Webpage

Demonstrate how to view IP address and change between DHCP and static IPv4 IP settings using a webpage interface. After submitting IP setting changes through the webpage form, the webserver will redirect the webpage to the new address of the device if necessary. The webpage also handles multiple interfaces, in case the module supports multiple interfaces (such as dual-ethernet and/or WiFi).
To change from DHCP to Static:

- Read the Static settings from the configuration record
- Set address mode to "Static"
- save to the configuration flash
- Option to reboot or change runtime settings

To change from Static to DHCP:

- You can leave the static settings intact if you wish, it will not affect DHCP
- Set address mode to "DHCP" or "DHCP w Fallback"
- Save to the configuration flash
- Option to reboot or change runtime settings

13.9 DNS Client

DNS Client Example
DNS example using GetHostByName(), GetHostByName4() and GetHostByName6() to resolve a name to an IP address for both IPv4 and IPv6.
To run this program:

- Start MTTTY and connect to the debug serial port
- You will be prompted to enter a name, such as www.netburner.com
- The DNS IP address resolution will be displayed

For this example to function, your NetBurner device must have access to the Internet, with a working configuration for the IP address, mask and gateway.

13.10 Embedded Flash File System (EFFS)

Embedded Flash File System (EFFS) examples. There are two types of file systems:

- EFFS FAT: A FAT 32 file system used for flash memory cards
- EFFS STD: A power fail-safe file system that resides in the flash memory of the module itself

Example Catagories:

- FAT File System (EFFS-FAT)
- Standard File System (EFFS-STD)

13.10.1 FAT File System (EFFS-FAT)

EFFS FAT32 Examples:

- Appplication Update
- Basic
- FTP FTP
- HTTP
- HTML Variables
- Multiple Tasks
- Performance Tests
- RAM Drive

13.10.2 Appplication Update

This application demonstrates how to update a NetBurner application in onboard flash memory from an external SD flash card. The application will look for a file on the flash card with the name specified by APPFILENAME. Use MTTTY to monitor the serial output of your NetBurner device. If this file is found, an application update can be performed by typing a character in response to the MTTTY prompt.

Note

All EFFS FAT examples require that you add the Embedded Flash File System File Allocation Table (EFFS FAT) library to your project: Add a Library to a Project

13.10.3 Basic

This program illustrates basic file system operations for SD/MMC and Compact Flash cards:

- Mounting a drive
- Determining amount of used and free file space
- Creating files
- Writing data
- Reading data
- Unmounting a drive

When the program executes it will display program status information through the debug serial port. This application has web server support for onboard flash/ram only, web pages cannot be run from external flash cards. This capability is demonstrated in the EFFS-HTTP example to illustrate the difference between the two operations.

Note

All EFFS FAT examples require that you add the Embedded Flash File System File Allocation Table (EFFS FAT) library to your project: Add a Library to a Project

13.10.4 FTP

This program illustrates file system and FTP operations for SD Flash cards:

- Mounting a flash drive
- Determining amount of used and free file space
- FTP access. We recommend Filezilla or WinScp

Status messages will be sent out the debug serial port.
Modules with an onboard microSD flash socket should use the multi MMC header files and functions because the modules are capable of supporting both onboard and external flash cards (even if you application only uses one).

Note

All EFFS FAT examples require that you add the Embedded Flash File System File Allocation Table (EFFS FAT) library to your project: Add a Library to a Project

13.10.5 HTTP

13.10.5.0.1 Overview This example demonstrates how to use the EFFS-FAT file system with SD cards. This file system is different than the EFFS-STD file system which uses onboard flash chip memory.

Note

All EFFS FAT examples require that you add the Embedded Flash File System File Allocation Table (EFFS FAT) library to your project: Add a Library to a Project

13.10.5.0.1.1 Application Functionality

- Creation of a text file from within the application named "TestFile.txt"
- Display of the text file
- Read/Write test of the text file
- Display of file space and amount used
- Format of the file system
- Test for file access functions: fgets(), fputs(), fprintf()

13.10.5.0.1.2 Network Functionality

- NTP is used to set the system time and time zone so that file time stamps are correct.
- HTTP web page interface and file display. If you use FTP to put in your own index.html page, it will be used in place of the example default page.
- FTP Server for file access. You can use client programs such as FileZilla to upload/download files

Note

Running the example in an environment where the device has Internet access will enable the Network Time Protocol (NTP) to automatically set the time. Otherwise, you can set the time manually through the serial port menu.

The EFFS-FAT manual, "EFFS-FAT-R3p31.pdf", is located in the "\nburn\docs" folder.

13.10.5.0.2 Building The Example

13.10.5.0.2.1 NBEclipse With Example Import Feature or Command Line If you build a new project and use the NBEclipse Example Import feature, all settings will be take care of automatically. If you are building using the example makefile on the command line, the makefile contains all necessary settings.

13.10.5.0.2.2 Copy index.html and MIME.txt files Copy index.html and and MIME.txt from the example's root directory to your flashcard, or put your own there. This can be done once the example is running with FTP as well.

13.10.5.0.2.3 Changing the Flash Card Type The Flash card type can be modified in cardtype.h. The default is a SD card.

13.10.5.0.2.4 Adding EFFS-FAT Functionality to an Existing NBEclipse Project Add the FatFile.a library to the GCC Linker settings

- Right click on your project.
- Navigate to C/C++ Build > Settings > GNC C/C++ Linker > Libraries.
- In the Libraries section, add "FatFile". Note there is no .a extension.

13.10.5.0.2.5 Adding EFFS-FAT Functionality to a Makefile In the makefile, add:

```
XTRALIB = $(NNDK_ROOT)/platform/$(PLATFORM)/original/lib/libFatFile.a
DBXTRALIB = $(NNDK_ROOT)/platform/$(PLATFORM)/original/lib/libFatFile.a
```

13.10.5.0.3 Running the Example The application provides access through an interactive serial port menu, web page, and FTP.

13.10.5.0.3.1 Serial Interface

- Run MTTTY or some other serial terminal program and connect to the device
- Status messages will be displayed as well as an interactive menu. If this is the first time the application has been run, the file system will be formatted.
- You can experiment with the various menu options.

13.10.5.0.3.2 Web Server (HTTP)

- The default web page will provide various interactive options and information display.
- Entering the IP address followed by "/DIR" (eg. `http://10.1.1.57/DIR`) will display the file directory
- The web server will display the default page based on what is in the file system in the following order: – The `index.html` (will superced index.html file compiled into the application) – The first available `*.html` file
- Use FTP (see below) to upload your own index.html file and it will replace the application default.

13.10.5.0.3.3 FTP Operation Connection to the device with a FTP client such as FileZilla will enable you to upload, download and view files.

13.10.5.0.3.4 File Priority When viewing the web page, the page displayed depends on the following order and their availability on the flash card:

- index.html file
- The first available .html file
- A web page list of all directories and files that exist on the flash card

Note

If *.html files exist on the flash card, then the program will always access the first available web page file by default.

13.10.6 HTML Variables

This example program sets up HTTP access to the FAT file system on SD Flash cards.
To run this example:

1. Compile the example for your chosen flash card in cardtype.h. The NetBurner development board supports SD Flash cards.
2. Copy index.html and and MIME.txt from the example's root directory onto your flashcard, or put your own there.
3. When the program runs, status messages will be displayed on the module debug serial port.
4. You can view the files on the flash card with any web browser by typing in the IP address of the module at the "/DIR" folder Example: "http://10.1.1.57/DIR").
5. When accessing the module via browser, what web pages will be displayed depends on the following order and their availability on the flash card:

- index.html files
- The first available .html file
- A web page list of all directories and files that exist on the flash card

Note

If .html files exist on the flash card, then the program will always access the first available web page file by default. If you want to access the web page list of all existing directories and files on the flash card, then you type in the URL of the module, followed by "/dir" (Example: "http://10.1.1.57/dir").

13.10.7 Multiple Tasks

This program illustrates basic file system operations for SD Flash cards for multiple tasks:

- Mounting a drive
- Determining amount of used and free file space
- Creating files
- Writing data
- Reading data
- Unmounting a drive

  When the program executes it will display program status information through the debug serial port.

Note

All EFFS FAT examples require that you add the Embedded Flash File System File Allocation Table (EFFS FAT) library to your project: Add a Library to a Project

13.10.8 Performance Tests

This example is used to test the speed of the EFFS FAT file system for operations such as file creation and data storage speed. All testing was done on a MOD54415.
Hardware:

- MOD54415-100IR on MOD-DEV-70 development board
- Kingston microSDHC 32GB Class 4, with MidroSC Adapter

Software: NetBurner tools revision 2.8.6 Date: February 7, 2018
Test results for file creation:

- 1 file: 15ms
- 100 files:32ms
- 1,000 files:93ms

Test results for data transfer, using 1MB of data:

- Write speed: 0.61 MBytes per second
- Read speed: 2.18 MBytes per second

13.10.9 RAM Drive

This program illustrates basic file system operations for RAM drives:

- Mounting a drive
- Determining amount of used and free file space
- Creating files
- Writing data
- Reading data
- Un-mounting a drive

When the program executes it will display program status information through the debug serial port.
This example is primarily different from the EFFS-BASIC example in that it by default uses the RAM drive for the file system and is used to show what is necessary to use the RAM drive. Namely, the project must include the content contained within the 'ramdrvMcf.cpp' source file.To build any of the EFFS- examples to use the RAM drive, simply add the 'ramdrvMcf.cpp' file to the build list and modify 'cardType.h' to define USE_RAM instead of USE_MMC or USE_CFC and set the EXT_FLASH_DRV_NUM to the correct RAM drive number.

Note

All EFFS FAT examples require that you add the Embedded Flash File System File Allocation Table (EFFS FAT) library to your project: Add a Library to a Project

13.10.10 Standard File System (EFFS-STD)

Examples for on-chip flash file system:

- EFFS-STD HTTP

13.10.11 EFFS-STD HTTP

13.10.11.0.1 Overview This example demonstrates how to use the EFFS-STD on-chip flash file system. This file system is different than the EFFS-FAT file system which is a FAT32 file system that works with SD Flash cards.

13.10.11.0.1.1 Application Functionality

- Creation of a text file from within the application named "TestFile.txt"
- Display of the text file
- Read/Write test of the text file
- Display of file space and amount used
- Format of the file system
- Read/Write of a diagnostics text file. This will be the data displayed on the System Diagnostics web page if that feature is enabled.

13.10.11.0.1.2 Network Functionality

- NTP is used to set the system time and time zone so that file time stamps are correct.
- HTTP web page interface and file display. If you use FTP to put in your own index.html page, it will be used in place of the example default page.
- FTP Server for file access. You can use client programs such as FileZilla to upload/download files

Note

Running the example in an environment where the device has Internet access will enable the Network Time Protocol (NTP) to automatically set the time. Otherwise, you can set the time manually through the serial port menu.

13.10.11.0.1.3 File System Size The example provides suggested file system sizes based on the total amount of available flash memory in each product. For example, the MODM7AE70 has a total of 2MB and 128k is allocated in the example. The MOD5441x products has 32MB total flash, and 1MB is used for the file system.
The flash chip configuration header files are located in the "flashChip" folder of the EFFS-STD example:

Platform	Header File	Total Flash	Example EFFS Size
MODM7AE70, SBE70LC	SAME70↩ Q21.h	2MB	128K
MOD54415, MOD54417	MX29GL256↩ F.h	32MB	1MB
NANO54415, SB800EX	MX25L6406↩ E.h	8MB	1MB

Please refer to the header files if you wish to change the size of the file system.

Note

The EFFS-STD manual, "EFFS-STD-R1p91.pdf", is located in the "\nburn\docs" folder.

13.10.11.0.2 Building The Example

13.10.11.0.2.1 NBEclipse With Example Import Feature or Command Line If you build a new project and use the NBEclipse Example Import feature, all settings will be take care of automatically. If you are building using the example makefile on the command line, the makefile contains all necessary settings.

13.10.11.0.2.2 Adding EFFS-STD Functionality to an Existing NBEclipse Project There are two steps to add EFFS-STD to your project.

1. Add the StdFFile.a library to the GCC Linker
2. Reserve space in the Flash memory for the file system

The first step is the same for all platforms:

- Right click on your project.
- Navigate to C/C++ Build > Settings > GNC C/C++ Linker > Libraries.
- In the Libraries section, add "StdFFile". Note there is no .a extension.

The second step is described in the following sections.

13.10.11.0.2.3 MODM7AE70, SBE70LC Platforms The "Locate Application at Fixed Address in Flash" feature must be enabled. This will disable the background Flash erase feature, would will erase the flash file system sectors.

- Right-click on the project and select "Properties"
- Navigate to C/C++ Build > Settings > NB Flashpack > General
- Check the checkbox for "Locate Application at Fixed Address in Flash"

13.10.11.0.2.4 NANO54415, SB800EX Platforms The CompCode Flag settings are used to reserve space in the memory map for the file system.

- Right click on your project.
- Navigate to C/C++ Build > Settings > NBCompCode > General
- Modify the memory address range field from:
  COMPCODEFLAGS = 0x04000 0x800000 // Original
  to
  COMPCODEFLAGS = 0x04000 0x700000 // Reserve space for file system

  The first parameter is the start of application space, and the second is the the start of file system space

13.10.11.0.2.5 MOD54415, MOD54417 Platforms The CompCode Flag settings are used to reserve space in the memory map for the file system.

- Right click on your project.
- Navigate to C/C++ Build > Settings > NBCompCode > General
- Modify the memory address range field from:
  COMPCODEFLAGS = 0xC0040000 0xC2000000 // Original
  to
  COMPCODEFLAGS = 0xC0040000 0xC1F00000 // Reserve space for file system

  The first parameter is the start of application space, and the second is the the start of file system space

13.10.11.0.2.6 Adding EFFS-STD Functionality to a Makefile There are two steps:

1. Add the EFFS-STD Library
2. Add the compcode flags or enable application fixed address

13.10.11.0.2.7 Add the EFFS-STD Library
```
XTRALIB =
    $(NNDK_ROOT)/platform/$(PLATFORM)/original/lib/libStdFFile.a
DBXTRALIB = $(NNDK_ROOT)/platform/$(PLATFORM)/original/lib/libStdFFile.a
```

13.10.11.0.2.8 MODM7AE70, SBE70LC Platforms Enable "Locate Application at Fixed Address in Flash" with: EXTRAPACKARGS += -cflag C:3

13.10.11.0.2.9 NANO54415, SB800EX Platforms Modify the memory address range field from:
COMPCODEFLAGS = 0x04000 0x800000 // Original
to
COMPCODEFLAGS = 0x04000 0x700000 // Reserve space for file system

13.10.11.0.2.10 MOD54415, MOD54417 Platforms Modify the memory address range field from:
COMPCODEFLAGS = 0xC0040000 0xC2000000 // Original
to
COMPCODEFLAGS = 0xC0040000 0xC1F00000 // Reserve space for file system

13.10.11.0.3 Running the Example The application provides access through an interactive serial port menu, web page, and FTP.

13.10.11.0.3.1 Serial Interface

- Run MTTTY or some other serial terminal program and connect to the device
- Status messages will be displayed as well as an interactive menu. If this is the first time the application has been run, the file system will be formatted.
- You can experiment with the various menu options.

13.10.11.0.3.2 Web Server (HTTP)

- The default web page will provide various interactive options and information display.
- Entering the IP address followed by "/DIR" (eg. `http://10.1.1.57/DIR`) will display the file directory
- The web server will display the default page based on what is in the file system in the following order: – The `index.html` (will superced index.html file compiled into the application) – The first available `*.html` file
- Use FTP (see below) to upload your own index.html file and it will replace the application default.

13.10.11.0.3.3 FTP Operation Connection to the device with a FTP client such as FileZilla will enable you to upload, download and view files.

13.11 Ethernet

Ethernet examples:

- Manual Configuration

13.11.1 Manual Configuration

Demonstrates how to set Ethernet speed and duplex manually instead of the default of autonegotiate.

13.12 Exception Try/Catch

This example program illustrates how to use C++ exceptions. It will throw an exception and verify it can be caught. The output is displayed to stdout, which can be viewed with MTTTY on the debug serial port.
COMPILATION INSTRUCTIONS To keep the application size small for users who do not need C++ exceptions, this feature is disabled by default. To enable exceptions you need to add the "-fexceptions" and flag to the C++ build options. In addition, "-frtti" also needs to be added. If these options are not enabled, the compiler output console window will notify you.
In this example, exceptions have already been enabled. To enable in your own application:

- Right-click on your project and select Properties
- Select "GNU C++ Compiler"
- Select "Miscellaneous"
- In the Other Flags field, add "-fexceptions -frtti"

13.13 External IRQ

Supported Platforms: MOD5441x, NANO54415, MODM7AE70, SBE70LC
This example program uses a GPIO pin to trigger an external interrupt pin. A jumper must be placed from the GPIO pin to the desired interrupt input. To run the example use MTTTY or some other serial terminal to interact with the menu to trigger the interrupt and view the interrupt counter.

13.14 Extra FD Circular Buffer

The NetBurner system software has support for implementing your own custom I/O device using a File Descriptor. It is Extra File Descriptors are defines in the 'iointernal.h' header file.
This is a simple example that creates a circular buffer and treats it as an I/O device by using a file descriptor. The I/O device is a 256 byte circular buffer that you can than write() to and read() from. The 256 byte buffer size makes the circular buffer wrap around simple for a byte array.
The 6 fd status functions are interrupt safe:
void SetDataAvail(int fd); void ClrDataAvail(int fd); void SetWriteAvail(int fd); void ClrWriteAvail(int fd); void SetHaveError(int fd); void ClrHaveError(int fd);
Side note: If you have an I/O device and wish to use interrupts, after adding the appropriate interrupt code that is specific to your device, you would use USER_ENTER_CRITICAL and USER_EXIT_CRITICAL to protect the internal data structures.

13.15 FileDescriptor

These series of examples provide code utilizing file descriptors.
JSON Examples:

- Extra FD Circular Buffer
- FDBuffer
- fdprintf - printf to file descriptor
- FdToNBString

13.15.1 Extra FD Circular Buffer

The NetBurner system software has support for implementing your own custom I/O device using a File Descriptor. It is Extra File Descriptors are defines in the 'iointernal.h' header file.
This is a simple example that creates a circular buffer and treats it as an I/O device by using a file descriptor. The I/O device is a 256 byte circular buffer that you can than write() to and read() from. The 256 byte buffer size makes the circular buffer wrap around simple for a byte array.
The 6 fd status functions are interrupt safe:
void SetDataAvail(int fd); void ClrDataAvail(int fd); void SetWriteAvail(int fd); void ClrWriteAvail(int fd); void SetHaveError(int fd); void ClrHaveError(int fd);
Side note: If you have an I/O device and wish to use interrupts, after adding the appropriate interrupt code that is specific to your device, you would use USER_ENTER_CRITICAL and USER_EXIT_CRITICAL to protect the internal data structures.

13.15.2 FDBuffer

Many of NetBurner's libraries utilize file descriptors(fd) for data input/output(I/O) transmission. To name a few, some libraries that utilize fd's include TCP, WebSockets, HTTP, FTP, email, serial, and the config system.
The user may find themselves wanting to capture I/O data from an fd to a buffer instead of using the fd as originally intended by a library. This may helpful for debugging, printing to serial, or logging. The intent of this example is to demonstrate exactly that.
This example utilizes the FDBuffer class which associates a buffer with an fd which can then be used with API's that utilize fd's. The FDBuffer class is defined in fd_adapter.h. The FBuffer class supports the use of file descriptor functions defined in iosys.h, such as read(), write(), and close().

This example will demonstrate the use of an FDBuffer object to read the config system's JSON object. The config system maintains config variables and their values as a JSON object. The config system API exposes functions to write the JSON object to an fd, which can be used to write the JSON to a TCP socket or to serial. Instead, we will use the FDBuffer to write the config JSON to a local buffer. The local buffer can be adapted to be stored to persistent storage (flash, SD card, etc), or transmitted remotely (email, syslog, FTP, etc).
An FDBuffer object, by default, can dynamically allocate as many as 20 pool_buffers to store data. Each pool_buffer is capable of storing ETHER_BUFFER_SIZE (1548) bytes. This translates to a total of 30,960 bytes of storage in an FDBuffer object. This value can be increased by modifying the MAX_FDBUFFER_FIFO_BUFFERS macro defined in /nburn/nbrtos/include/fd_adapter.h.

13.15.3 fdprintf - printf to file descriptor

Application that demonstrates the ability to close, open, and print to file descriptors

13.15.4 FdToNBString

Many of NetBurner's libraries utilize file descriptors(fd) for data input/output(I/O) transmission. To name a few, some libraries that utilize fd's include TCP, WebSockets, HTTP, FTP, serial, and the config system.
The user may find themselves wanting to capture I/O data from an fd to a local variable instead of using the fd as originally intended by a library. This may helpful for logging, debugging, or printing to serial. The intent of this example is to demonstrate capturing file descriptor I/O data to an NBString.
This example utilizes the NBStringBuilder class which associates a buffer with a file descriptor which can then be used with API's that utilize fd's. The NBStringBuilder class is defined in nbstring.h. The NBStringBuilder class supports the use of file descriptor functions defined in iosys.h, such as read(), write(), and close().
This example will demonstrate the use of an NBStringBuilder object to read the system diagnostics data set. With the use of EnableSystemDiagnostics(), the system maintains a diagnostics data set as a JSON object. The diagnostics API exposes functions to write the JSON object to an fd, which can be used to write the JSON to a TCP socket or to serial. Instead, we will use the NBStringBuilder to write the diagnostics data to an NBString. The NBString can be utilized to store the diagnostics to persistent storage (flash, SD card, etc), or transmitted remotely (email, syslog, FTP, etc). For the simplicity of an example, this application will print the NBString to serial.

13.16 FTP

FTP can be run with or without a file system. The examples in this section do not use a file system. These are useful for applications such as:

- Simple download of data as a file by a FTP client, such as measurement readings.
- Updating an application image.
- Viewing web or interface data in a file format.

Since there is no file system, these examples use callback functions to handle the various FTP transactions, such as reading, writing and listing a file. If you do not wish to use the callback scheme, please refer to the file system examples described below.
Examples of FTP using a file system take advantage of the Embedded Flash File System (EFFS). There examples are located in the "\nburn\examples\EFFS" folder. There are two types of file systems:

- EFFS FAT (File Allocation Table), which is a FAT32 file system used by external flash cards.
- EFFS STD (Standard File System), which is a custom file system that runs in the onboard flash memory of the NetBurner device.

FTP in this section:

- FTP Client
- Display HTML Files (FTPD)
- Simple FTP Server (FTPD)

13.16.1 FTP Client

This program will run a FTP Client program on the NetBurner board to do the following:

- Connect to a FTP Server
- Change to a remote directory called "test1"
- Obtain a directory listing and print the listing to stdout
- Create a file on the FTP Server called WriteTest.txt
- Read back contents of WriteTest.txt and send to stdout

Setup Requirements:

1. Access to a FTP Server
2. User name and password for the FTP Server
3. A directory called "test1" must exist on the FTP Server
4. Write permissions to the test1 directory
5. You must modify the `#define` values in this file to match your FTP server.

13.16.2 Display HTML Files (FTPD)

Example to display the files in a project's HTML directory as a read-only file system.

- Serial port output displays status information
- Device web page provides example information and a link to display files, using the CPPCALL HTML tag
- Files can also be accessed with FTP clients such as Filezilla

This is an advanced application and requires knowledge of FTP and FTP applications. A significant part of the complexity involved concerns navigating and displaying file names and directories.
Starting in 2020 some browsers have begun to block FTP access. In such a case FTP client programs can be used instead.

13.16.3 Simple FTP Server (FTPD)

Simple example of a FTP server implementation without a file system. It can be very useful for a lightweight method to download system information and data, and upload configuration information. Interaction with the FTP server in an application is accomplished using callback functions to handle events, such as listing the files, sending a file, and receiving a file. Please refer to the additional FTP examples for use with a file system.

- Status messages and data are sent out the debug serial port and can be viewed with a terminal such as mttty.
- Enables the FTP client to download a file named ReadFile.txt. While this is fixed content, it can contain whatever dynamic content you wish.
- Enables the FTP Client to upload an ASCII text file named WriteFile.txt. The file is not stored in memory, but is sent out the debug serial port.

13.17 GDB Debugger

This is a very simple application to run with the debugger on a NetBurner network-enabled hardware platform. The application will boot and stay in a while() loop that displays how long the program has been running to the default serial port.

13.18 General Purpose I/O

These general GPIO examples apply to multiple NetBurner platforms. Please refer to the "Example Applications > Platform Specific" section of this document for addional GPIO examples specific to your particular NetBurner platform.
Examples in this section:

- GPIO Blink
- General Purpose I/O Service

13.18.1 GPIO Blink

This example demonstrates the basic operation of GPIO pins using the PinIO class. It blinks one of the indicator LED on the device carrier board.
This example is purposely simple and does not demonstrate all the PinIO capabilities. For further information on what the PinIO class can be used for or how to use it, see the GpioServer example or refer to the PinIO documentation. Additionally, most platforms have additional GPIO examples in the Platform Specific example directory which focus on GPIO specifics.
Please refer to the "Example Applications > Platform Specific" section of the NetBurner Documentation for additional GPIO examples specific to your particular NetBurner platform.

13.18.2 General Purpose I/O Service

Example to control GPIO pins though command interface.
This is a general purpose GPIO example and a large portion of the code is dedicated to a parser and command structure. Most platforms have additional GPIO examples in the Platform Specific example directory which focus on GPIO specifics.
This example sets up a command parser on the default serial port to configure and control GPIO signals on the NetBurner module. For devices with multiple connectors the example uses the peripheral connector, such as P2 or J2, or the main connector for devices with a single connector. Depending on your specific device, not all signals can be GPIO. Please consult the device data sheet for additional information.
To run the example build and load the project into your device. Then run the MTTTY serial terminal and connect to your device's debug/status port.
The menu options are:

- HI Set a pin high
- LO Set a pin low
- TGL Toggle a pin
- OUT Configure a pin to be an output
- IN Configure a pin to be an input
- RD Tristate and read a pin

For example, to set pin 5 high use the command: "HI 5".
Please refer to the "Example Applications > Platform Specific" section of the NetBurner Documentation for additional GPIO examples specific to your particular NetBurner platform.

13.19 IP Address Object (IPADDR)

IPADDR IP Address Object Example
An IPADDR object is a C++ object that can contain either an IPv4 or IPv6 IP address. An object is used to enable application support both IPv4 and IPv6 without having to change source code. Once you assign an address to the object, it's member functions will automatically call the appropriate member function for the task to be performed.
For example, if you have an IPADDR object named myIPaddr and assign it a value of 192.168.1.10, then calling `myIPaddr.print()` will print the address in IPv4 notation. If you assign it an IPv6 address, calling `myIPaddr.print()` will print the address in IPv6 notation.

The NetBurner system has added a printf formatting option of "%I", enabling you to easily display IP addresses with formatting. For example, `iprintf("My IP address is %I\r\n", myIPaddr);` This formatting option works for all printf variants, such as printf, iprintf, sprintf, etc.
In rare cases in which you want to create an object of a specific type, you can create objects as IPADDR4 or IPADDR6. However, we highly recommend using IPADDR for all your IP addresses. Note that if you are using a printf style format specifier, you must use `"%HI"` instead of `"%I"` and pass in a IPADDR4 object, otherwise the value displayed will be 0.0.0.0.

13.20 IPv6

IPv6 Examples:

- DHCPv6

13.20.1 DHCPv6

This example demonstrates the various callback functions of the DHCPv6 Client for requesting, adding, and processing DHCPv6 options.
The DHCP offer name is set by the DEVICE_NAME parameter in the configuration settings for each network interface.

13.21 JSON

These series of examples provide code for a JSON server running on a NetBurner device, "demo.netburner.com", and JSON client examples for GET, POST, loading applications and HTML. If you type "demo.netburner.com" in you web browser and can see a JSON server, you can run the client examples againt it.

- JSON Lexer

JSON Examples:

- NetBurner Demo Server for JSON Client Examples
- Get JSON Object From Server
- Get Non-JSON Object From Server
- Get JSON Object From Server
- Post JSON to Server and Display Result
- Post JSON to Server
- Get Application Image and Load it From a Web Server
- Simple JSON HTML Example
- Simple JSON Post Receiver Example

13.21.1 JSON Lexer

The examples here show how to user the JSON Lexer classes, ParsedJsonDataSet and JsonRef, to build and interact with JSON data sets.
JSON Lexer Examples:

- JSON Array
- JSON Array with Objects
- Parse Test

13.21.2 JSON Array

This program shows how to use the JSON Lexer classes ParsedJsonDataSet and JsonRef to create and access array information.

13.21.3 JSON Array with Objects

This program shows how to use the JSON Lexer classes ParsedJsonDataSet and JsonRef to create and access arrays containing sub-objects.

13.21.4 Parse Test

This program shows how to use the JSON Lexer classes ParsedJsonDataSet and JsonRef to create and access JSON information.

13.21.5 NetBurner Demo Server for JSON Client Examples

This is the source code for the application running on the demo.netburner.com web site. The JSON Client examples in this section interface to this server to demonstrate each of the features. The source code for this server is here for refereance and completeness rather than something you need to implement on your own. All of the techniquies used here and more clearly ilistrated with the other JSON examples.

Warning

This project requires a certificate and private key to build

13.21.6 Get JSON Object From Server

Simple program that demonstrates retrieving a JSON object from a site and using it to present the user with dynamic output

13.21.7 Get Non-JSON Object From Server

Simple program that demonstrates retrieving non-JSON object from a site and using it to present the user with dynamic output

13.21.8 Post JSON to Server and Display Result

Simple program that demonstrates POSTing a JSON object to a server and then displaying the returned JSON result.

13.21.9 Post JSON to Server

Simple program that demonstrates POSTing a JSON object to a server and then displaying the returned result as text.

13.21.10 Get Application Image and Load it From a Web Server

Simple program that demonstrates retrieving a new code image from the web.

13.21.11 Simple JSON HTML Example

Shows several methods to have a NetBurner respond to web requests with JSON objects. Shows how to:

- Respond and generate the text directly
- Respond and generate the response with a JSON lever object.
- Respond and generate the response using configuration variables.
- Respond and generate the response using a pre-built page stored in with web pages and using the dynamic web page capabilities.

13.21.12 Simple JSON Post Receiver Example

Simple JSON Post Receiver Example shows how to recieve a JSON object as a post from an external Restful API or from Javascript within a web page.

13.22 malloc

This example demonstrates the usage of malloc and free. It identifies way's to track heap space usage by calling spaceleft() and mallinfo(), which can help prevent an applications from running out of dynamic memory.
It is important to use both spaceleft() and mallinfo() to determine the size of the heap. As this application demonstrates, spaceleft() alone will not always give the total space available to malloc.
Dynamic memory and fragmentation is beyond the scope of this example, but in short, the largest amount of contiguous dynamic memory an application can malloc depends on how the application handles malloc and free. For example:

- A heap has a total of 1MB of memory available.
- The application allocates 100KB 3 times, lets call them A, B and C.
- There is now 700KB left. Functions spaceleft() and mallinfo() will report this number.
- The spaceleft() function always reports the amount of "untouched" memory. This is memory that has never been allocated, and does not include the amount available to the application that has been allocated and freed. It is useful in that it provides the application with the amount of contiguous dynamic memory that can be allocated. It is commonly referred to as a "watermark".
- The mallinfo() function returns the total amount of heap space available, including both untouched and freed memory. So at this point in this 1MB example, it would report 700KB a well.
- Now the application frees A and B. The spaceleft() function will report the watermark value of 700KB. The mallinfo() function will include the freed memory and report 900KB.

In this code example the application will allocate 3 chunks of space. The first is 1MB, then a 3MB, then a 512KB chunk. It will then free the data in a different order.
After every malloc and free, the amount of heap space will be displayed by both reporting functions showing heap space used, free, and untouched.

13.23 MQTT Examples

Examples:

- MQTT Periodic
- Simple Publish

13.23.1 MQTT Periodic

The Periodic MQTT Example is a basic demonstration of the NetBurner MQTT client implementation, and is a good starting point for building custom MQTT status reporting.

13.23.2 Simple Publish

Simple MQTT Publish example

13.24 Multicast

This example program enables you to add a NetBurner device to a multicast group by specifying the multicast ip address and port number in MTTTY when prompted. You can then send data by typing it in MTTTY, or view received data.
This example only supports IPv4

13.25 Multihome

Multihome application example (IPv4 only)
This program will demonstrate how to implement both a DHCP address and static IP address using the Multihome functionality of the NetBurner TCP/IP Stack. The NetBurner device will try to obtain a dynamic IP address from a DHCP Server for the first Network Interface, and set a static IP address for the second Network Interface. The end result is that the NetBurner device will respond to either IP address. The example will print debug information out the debug serial port, and display the IP address information on a web page that can be accessed from either IP address.
To enable multihome capability, you must uncomment the MULTIHOME definition in the include file \nburn\nbrtos\include\predef.h and rebuild the system files.

13.26 NB Approve Shutdown

The NBApproveShutdown callback function can be used by an application to put the system in a safe state before an application update, configuration update, or reboot event occurs. For example, closing active TCP sockets, ensuring Flash and/or file system write operations are complete, and putting critical peripherals in a safe state. The NBApproveShutdown() function as a weak reference to a system function that always returns true by default. If an application creates it's own function using the same signature, that function will be used instead.
A reason for the reboot request is passed to the function. The system will automatically call NBapproveShutdown() for the following reasons:

```
#define SHUTDOWN_CODEUPDATE          (1)    // A code update is requested
#define SHUTDOWN_CONFIGURE_REBOOT    (2)    // Configuration values have been modified with a requested reboot
```

An application can choose to ignore the parameters or add it's own. For custom reasons, the application should call NBApproveShutdown() with the appropriate reason:

```
// Custom reboot reason. System reasons start at 1, so pick something much larger
#define SHUTDOWN_CUSTOM_REBOOT 100

if( NBApproveShutdown(SHUTDOWN_CUSTOM_REBOOT))
{
     OSTimeDly(TICKS_PER_SECOND * 5);
     ForceReboot();
}
```

This example provides a serial port menu with options enable or disable a reboot for application or settings updates. For testing purposes you can select reboot from the serial menu, or attempt an application code update.

13.27 NBString Class

This example program demonstrates the functionality NBString class.

13.28 NetBios Name

The example demonstrates how to use NetBIOS naming, so you can access your NetBurner device by name as well as IP address on a local area network. The name can be changed on the web page.
The NetBios Name can contain any alphanumeric character except spaces and \ : / * ? ; | . It must be 15 characters or less.

13.29 Network Time Server (NTP) Client

A simple example that will get the time from a NTP server and display the UTC and PDT time through the debug serial port. Use MTTTY to monitor the output.
For this example to function, your NetBurner device must have access to the Internet, with a working configuration for the IP address, mask and gateway.

13.30 NTP & Real-Time Clock

NTP-SYSTEM-RTC DEMO APPLICATION
HARDWARE SETUP: Tested with MOD54415 on a MOD-DEV-70 hardware rev 1.93.

This example demonstrates the following features:

- Updates the system time with the current UTC time pulled from an available NTP server (pool.ntp.org). If your module is set to static IP address settings, then you will need to manually assign a DNS server address for this feature to work.
- Take the current system time and save it to the real-time clock (RTC). The RTC will then be running in sync with the system time.
- Take the current time in the RTC and set the system time information. The system time will then be running in sync with the RTC.
- When the device or module is reset or powered on again after a recent power loss, the system time will be reset, but the RTC can continue running until the super-capacitor is discharged. The software-reset option is available to demonstrate the RTC's ability to retain saved time information once it is properly set.
- Set the local time zone information with a call to tzsetchar(). For more more information on this function, refer to "Chapter 14 - NBTime Library" of the NetBurner Runtime Libraries PDF document. In this example, Pacific time zone information is used as a demonstration.
- Display the current system and RTC times, and local time zone information if applicable. It is preferred that the RTC store the UTC time, not local time.
- In order to take advantage of setting and getting time information from the real-time clock (RTC), a device that contains an RTC component or a module that is mounted on a development board with an RTC component must be used. Current RTC components supported in this example at the time of this writing are the Intersil X1226 and NXP PCF8563.

13.31 Overload Directory & System Files

The Overload Directory Example is a basic demonstration of the overload directory in a project. The overload directory is used to override any system file by including local copies of system source files or headers. This allows the user to make changes to system files exclusively for a project. The overload directory provides an alternative to modifying the original system files, which would affect all projects.
To use this feature, the file to be overloaded should be placed in the overload directory under a directory structure that matches the file's location relative to the NNDK install directory.
This example application utilizes the overload directory to override predef.h (a NetBurner configuration header file). The overloaded predef.h defines the macro NBRTOS_TIME, which is not defined by default.
To overload the predef.h header, the directory structure is replicated inside the overload directory. For example, predef.h is located in the following directory in the NNDK install:

```
/nburn/nbrtos/include/predef.h
```

Therefore, the directory structure is replicated inside the overload directory with the following structure:

```
<project root>/overload/nbrtos/include/predef.h
```

Additional instructions if the overload directory is used in an NBEclipse project:
After adding the file to be overloaded, clean and rebuild the NetBurner Archive by selecting "Clean NetBurner Archive" under the Build Targets pull-down in the project. From that point on, modifying predef.h (or any overloaded source/header) will automatically rebuild the necessary files in the NetBurner Archive when then project builds.
If a system include folder is overloaded, this folder should be added to your project include paths. Right click on the project and select project properties. Under C/C++ Build->Settings, select GNU C++ Compiler->Includes and add the overload include folder. If utilizing C code, then GNU C Compiler->Includes should also be added.

13.32 PLATFORM SPECIFIC

With the exception of the Platform Specific examples, all examples in the `\nburn\examples` directory can be run on any of the NetBurner 3.x platforms: MODM7AE70, MOD5441x, NANO54415 and SB800EX. The Platform Specific examples in this section have specific hardware requirements.
The example categories are:

- Examples designed for only one platform: MODM7AE70, MOD5441x, NANO54415 or SB800EX
- Multi-Platform: Some examples for platforms that use the same microprocessor take advantage of that fact and share the same application source code. In these instances the section will be labeled with the supported platforms, such as the section labeled: "MOD5441x, NANO54415, SB800EX".

Platform specific examples

- MOD5441x, NANO54415, SB800EX
- MODM7AE70, SBE70LC
- SOMRT1061
- External IRQ
- SPI

13.32.1 MOD5441x, NANO54415, SB800EX

Platform Platform specific examples for ColdFire MCF5441x based products:

- MOD54415
- MOD54417
- NANO54415
- SB800EX

- 1-Wire
- Analog to Digital
- CAN to Serial
- Digital to Analog (DAC)
- DSPI to Serial
- Load Application From Flash Card
- SDHC Flash Card
- EMI Emissions Reduction
- External NMI IRQ 7
- RTC - External
- I2C
- RTC - On Chip
- Periodic Interrupt Timer
- Pulse Generator and Counter
- PWM
- Core Watchdog Timer (CWT)
- WAV File Audio Player

- MOD5441x Only
- NANO54415 Only
- SB800EX Only

13.32.2 1-Wire

Supported Platforms: MOD5441x, NANO54415
This program used the 1-wire peripheral to read a registration number from the DS2401 Silicon Serial Number chip.

13.32.3 Analog to Digital

MOD5441x and NANO54415 Analog to Digital Platform Examples

- Simple ADC
- Periodic ADC

13.32.4 Simple ADC

Simple Analog to Digital converter example
Supported Platforms: MOD5441x, NANO54415
This example demonstrates how to read all 8 ADC inputs. The ADC register values will be displayed on the serial port interface, as well as the floating point conversion values.
Please refer to the data sheet for your specific platform to determine which of the 8 ADC channels are routes to interface connectors.

13.32.5 Periodic ADC

PeriodicAD - DMA Driven, periodic sampling Analog to Digital example
Supported Platforms: MOD5441x, NANO54415
This example shows how to setup a periodic sampling Analog to Digital conversion with buffering. It uses the onboard Analog to Digital Converter in loop mode to continuously sample. It then utilizes the DMA Timer to trigger a dma transfer from the selected ADC channel, to the next location in the sample buffer. The example includes interrupt triggers at a half-full and completely full buffer. Finally, the specific result of the application is to sample for some period of time, record the data to an output .wav file, and repeat.
This example requires an external SD Flash card, such as the one on the NetBurner development board.
To build and run this example:

- Create a new project.
- Import the example source files to the src directory
- Import the following EFFS support files from `\nburn\examples\_common\EFFS\FAT`: cardtype.h, FileSystemUtils.cpp, FileSystemUtils.h.
- If you are using NBEclipse, then you will also need to tell the linker to include the FatFile library:
  - In NBEclipse, right-click on your project, and select "Properties"
  - Select "C/C++ Builds -> Settings" on the left-hand side
  - Select "GNU C/C++ Linker -> Libraries" under the "Tool Settings" tab
  - In the "Libraries" list box, add "FatFile" by using the action icons provided in top-right corner of the list

13.32.6 CAN to Serial

CAN TO SERIAL EXAMPLE PROGRAM
The Controller Area Network (CAN) bus is a twisted pair multidrop cable is specified with a length ranging from 1,000m at 40Kbps to 40m at 1Mbps. The maximum payload of a message is 8 bytes, and all messages carry a cyclic redundancy code (CRC). Each message has an identifier, which can be interpreted differently depending on the application or higher-level protocols used. All nodes on the network receive each message and then decide whether that identifier value is of interest.
The NetBurner CAN to Serial example program uses the NetBurner CAN API to send and receive CAN messages though a serial port interface. It will listen for frames using the RegisterCanRxFifo() function. Frames are received using the CanRxMessage class, and are sent out the RS-232 debug port of your NetBurner CB34EX module. Note that you must have at least two CAN devices run any CAN programs.

```
   RS-232  |----------|    CAN    |----------|
<--------->| Dev 1    |<--------->| Analyzer |
           |----------|           |----------|
```

SETUP This application has been tested on the SB800EX, MOD54415, MOD54417 and NANO54415. The CAN signals were connected to a CAN analyzer. MTTTY was run on a PC to enter the data to the RS-232 serial port that is ten transmitted out the CAN interface. If you have 2 NetBurner devices you can connect them together (in place of the CAN analyzer) and send data in both directions.
PLATFORM NOTES

- This example was written to run on multiple platforms.
- The MCF5441x processor has 2 CAN modules, 0 and 1. Some platforms bring out both, but others with limited pin count only bring out only one.
- Please refer to the data sheet to determine which CAN signals correspond to which module or development board pin.
- The MOD5441x devices were tested in a MOD-DEV-100 development board which has a CAN transceiver
- The SB800EX comes in 2 versions: CAN 0 on the 5-pin terminal strip, or CAN 1 on the DB9 connector.

RUNNING THE EXAMPLE

- A minimum of TWO CAN devices are required for CAN to operate
- This example application sets the CanInit() mask to 0, which will listen for all packets and display the info on MTTTY.
- This example application will send data you type in MTTTY with an id of 101 (0x65).
- If you receive a message indicating the CAN message could not be sent, it usually means the other device did not acknowledge the CAN packet.

13.32.7 Digital to Analog (DAC)

This example demonstrates how to use the DAC0 signal in two modes:

1. Step up the output from 0 to 0xFFF by writing to the DAC data register.
2. Use the auto waveform generator feature synced to DMA timer 0 to step up the output.

You must have a voltage reference as described below

13.32.7.0.0.1 MOD5441x Platforms The DAC0 signal is on pin on pin J2.9. You must also have the analog reference voltage pin connected to something between 0 and 3.3VDC. The easiest way is to connect 3.3VDC by jumpering J2.2 (Vcc) to J2.5 (Analog ref. input)

13.32.7.0.0.2 NANO54415 Platforms The DAC0 signal in on pin on pin J2.9. You must have the analog reference voltage pin connected to something between 0 and 3.3VDC. On the NANO Development board you can jumper P3.2 (Vcc) to P2.8 (Analog ref. input)

13.32.8 DSPI to Serial

DSPI TO SERIAL EXAMPLE
Supported Platforms: MOD5441x, NANO54415
This program will illustrate how to use the NetBurner DSPI driver. The driver provides most of the standard DSPI options while providing some extra features, including:

- 4-32 bit transfers instead of the standard 4-16 bit
- Automation of the queuing system
- Interrupt driven transfers

In this example we demonstrate how to take advantage of the interrupt feature by using a Semaphore. This will allow any other lower priority tasks to run during the transfer. There is also an option to not use a semaphore and poll the transfer using the bool DSPIdone(); function provided in qspi.h.
This example can be used with no external SPI device by placing a jumper across J2[27]-J2[28] on the DEV-70/100 carrier board or Pins[33]-Pins[35] for the Nano54415 dev board. This will loop-back the DSPI data and echo the sent character back to the serial terminal.
The DSPI2Serial example can easily be expanded to DSPI2TCP by creating a TCP socket file descriptor similar to the TCP2Serial example. This TCP file descriptor would then used in place of the serial file descriptor.

13.32.9 Load Application From Flash Card

Example of how to update an application from a binary image file on a flash card.
Supported Platforms: MOD5441x
This application uses the Extra File Descriptor (ExtraFD) mechanism to read a binary application image as an input stream from a flash card to the FD. This image is then programmed into flash memory as an application. As with any application update, a reboot is required to run the new application.
As with any Extra FD object, you must create and assign callback functions for read, write, close and peek. There is another example of using the Extra FD with a circular buffer in the `\nburn\examples` directory.
This application was created and tested on a MOD5441x platform using the onboard microSD Flash card. There are also comments on how to use the external Flash card interface as well.

13.32.10 SDHC Flash Card

Supported Platforms: MOD5441x, NANO54415, SB800EX
This program demonstrates the EFFS FAT file system and FTP operations for SD/MMC/SDHC Flash cards on 5441x platforms:

- Mounting a flash drive
- Determining amount of used and free file space
- Using FTP to upload and download files

NetBurner 5441x based platforms have two types of flash card interfaces:

1. Native SPI
2. SDHC

These types are mutually exclusive and the flash card connector on the device must be wired one way or the other. As of the writing of this example: MOD54415, MOD54417:

- Onboard microSD connector is SPI.
- MOD-DEV-70 development board is wired for SPI.
- Signals are available on the module headers to wire for SDHC or SPI

NANO54415:

- Onboard microSD wired for SDHC

- NANO development board wired for SPI

SB800EX

- Onboard microSD connector is wired for SDHC

Any SPI connected card is able to run any of the EFFS FAT examples located in the EFFS directory of `\nburn\examples`.
To run the speed test you will need to place a file named TEST.BIN on the flash card. You can do this with FTP to the device directly, or copy it onto the flash drive from another device. Test results will be displayed through the debug serial port.
Modules with an onboard microSD flash socket should use the multi mmc header files and functions because the modules are capable of supporting both onboard and external flash cards (even if you application only uses one).

13.32.11 EMI Emissions Reduction

Supported Platforms: MOD5441x, NANO54415, SB800EX
This example demonstrates how to implement the EMI reduction options described in the associated document EMI Noise Reduction Suggestions. Each design will have it's own unique behaviors and some or all of these suggestions can help reduce EMI. In addition to the code described below, the app note also covers design considerations for Ethernet cabling, cables in general, enclosure design, input power filtering (very important), and PCB routing.
The example code includes:

- Enable/disable unused clocks
- Enable/disable unused peripherals
- Modify signal slew rates
- Putting the processor in wait mode while idle
- The MOD54415 also has the ability to enable spread spectrum clock mode for the processor (available on h/w revisions 2.3 and later).

13.32.12 External NMI IRQ 7

MCF441X NON-MASKABLE IRQ 7 EXAMPLE
Supported Platforms: MOD5441x, NANO54415
This example program uses a GPIO pin to trigger a level 7 interrupt. A jumper must be placed from the GPIO pin to the interrupt input pin. To run the example, use MTTTY or some other serial terminal to interact with the menu to trigger the interrupt and view the interrupt counter.
The MCF54415 processor is used on both the MOD54415 and NANO54415 modules. The table below list the GPIO jumper configuration:
MOD54415:

- GPIO signal pin: J2.33
- IRQ 7: pin 48

NANO54415, and dev board revision 1.3:

- GPIO signal: Nano pin 12 = Dev board pin P2.12
- IRQ 7: Nano pin 9 = Dev board pin P1.9

Examples of assembly language and C/C++ ISR processing functions are provided.

13.32.13 RTC - External

External I2C RTC example
Supported Platforms: MOD5441x, NANO54415
HARDWARE SETUP: Tested with MOD54415 on a MOD-DEV-70 hardware rev 1.93.
This example demonstrates the following features:

- Updates the system time with the current UTC time pulled from an available NTP server (pool.ntp.org). If your module is set to static IP address settings, then you will need to manually assign a DNS server address for this feature to work.
- Take the current system time and save it to the real-time clock (RTC). The RTC will then be running in sync with the system time.
- Take the current time in the RTC and set the system time information. The system time will then be running in sync with the RTC.
- When the device or module is reset or powered on again after a recent power loss, the system time will be reset, but the RTC can continue running until the super-capacitor is discharged. The software-reset option is available to demonstrate the RTC's ability to retain saved time information once it is properly set.
- Set the local time zone information with a call to tzsetchar(). For more more information on this function, refer to "Chapter 14 - NBTime Library" of the NetBurner Runtime Libraries PDF document. In this example, Pacific time zone information is used as a demonstration.
- Display the current system and RTC times, and local time zone information if applicable. It is preferred that the RTC store the UTC time, not local time.
- In order to take advantage of setting and getting time information from the real-time clock (RTC), a device that contains an RTC component or a module that is mounted on a development board with an RTC component must be used. Current RTC components supported in this example at the time of this writing are the Intersil X1226 and NXP PCF8563.

13.32.14 I2C

MCF5441x I2C Platform Examples

- I2C Address Scan
- I2C PicKit Demo Board

13.32.15 I2C Address Scan

Scan the I2C bus for devices and display their addresses.
Supported Platforms: MOD5441x, NANO54415
ScanI2CBus sends out a request on the I2C bus and waits for an ack, scanning each address in the address space. If a device responds, the device address is reported.
This utility is very helpful to:

- Verify the hardware is connected correctly.
- Understanding the addressing in I2C device data sheets. Some data sheets include the read/write bit as part of the address, and some do not.

Notes when using a MOD5441x and the MOD-DEV-70 development board:

- The MOD5441x has up to 5 I2C ports that can be used: 0, 1, 2, 4 and 5.
- I2C0, J2-39 and J2-42: This is the default I2C peripheral. It has 4.7k pull up resistors on SDA and SCL, and is connected to a real time clock chip.
- I2C1, J2-41 and J2-44: Sgnals are not connected to anything, but you will need to add pull up resistors.
- I2C2, J2-17 and J2-18, are used for USB, but can be converted to I2C2 by moving 2 resistors on the MOD5441x. The ordering model number is MOD54415-100IR-I2C2. Refer to PCN 001 on the MOD5441x product page.
- I2C4, J2-3 and J2-4: Signals are used for UART0 serial port
- I2C5, J2-21 and J2-22: Signals are used for UART1 serial port

13.32.16 I2C PicKit Demo Board

Microchip Pickit Serial I2C Demo Board Driver
Supported Platforms: MOD5441x, NANO54415
Demonstrates the use of 5 Microchip I2C Peripherals:

- 24LC0B EEPROM, 2Kbit (256 x 8)
- MCP9800 Temperature sensor
- MCP23008 GPIO port expander, connected to LEDs
- TC1321 10-bit DAC
- MCP3221 12-bit ADC

13.32.17 RTC - On Chip

Supported Platforms: MOD5441x, NANO54415
A simple Real Time Clock (RTC) example using the RTC on the MCF5441x microprocessor. Demonstrates how to get and set time through NTP, RTC and manually. An interactive menu is accessed through MTTTY.
For this example to function, your NetBurner device must have access to the Internet, with a working configuration for the IP address, mask, gateway and DNS server.

13.32.18 Periodic Interrupt Timer

Periodic Interrupt Timer (PIT) Example
Supported Platforms: MOD5441x, NANO54415 SB800EX
There are a total of 4 PIT timers on the MCF54415 processor, numbered 0 - 3:

- 0 is reserved for the RTOS system clock
- 3 is reserved for the network debugger
- 1 and 2 are available for customer use

The PIT functions below are part of the 54415 system libraries, and defined in pitr_sem.cpp and pitr_sem.h, located in the include and system folders of your MOD54415 or NANO54415 platform directories.
In this example a semaphore will be set in the PIT interrupt routine, and the semaphore count will be displayed through the serial port.

13.32.19 Pulse Generator and Counter

Supported Platforms: MOD5441x, NANO54415
Demonstrates how to use the DMA Timers to generate a waveform output on one timer, and use the timer capture feature to count the waveform pulses on a second timer. Features include:

- Programmable frequency output
- Measurement of frequency input
- Timer count display
- Values also displayed on the web page interface

The application will work on the MOD54415, MOD54417 and NANO5441. The selected timer channels must be jumpered together on the development board so that the waveform output is connected to the capture input.
MOD5441x: Timer J2 Pin
0 36 1 34 2 33 3 32
NANO54415: Timer Pin
0 19 1 21 2 23 3 25

13.32.20 PWM

Example program for ColdFire MCF5441x PWM capability on the MOD5441x core module.
Supported Platforms: MOD5441x, NANO54415
Description: Toggles a PWM signal on pin 35 on the NetBurner NANO or MOD54415 module using two different settings. While each PWM module on the 5441x core is capable of producing multiple output signals for use in controlling motors as well as other applications, this example sets the output for one pin only.
There are 2 functions illustrating different methods to create the pulses:

1. Center-Aligned pulses
2. Edge-Aligned pulses

13.32.21 Core Watchdog Timer (CWT)

Supported Platforms: MOD5441x, NANO54415, SB800EX
This example demonstrates how to enable the watchdog, set the timeout value, and service the watchdog. An interactive menu and status messages are displayed through the debug serial port. If the watchdog is not serviced before the timeout expires, events such as a system reset or interrupt service request (IRQ) can occur. The following watchdog timeout modes are supported by the 5441x processor:

- System reset
- IRQ only
- IRQ on first timeout, then system reset on the second timeout
- Windowed mode, this is an advanced mode of operation that processes a timeout in 25% increments. Please refer to the MCF5441X Users Manual for more information. This mode is not covered in the example.

Servicing the watchdog can be accomplished in a number of ways:

1. A higher priority task that uses a timer, such as the Periodic Interrupt Timer (PIT). This is the most common method.
2. In an Interrupt Service Routine (ISR). A drawback here is that interrupts can still occur, even if a specific task is not functioning correctly.
3. Servicing throughout an application with specific function calls. This is very difficult to cover all of the edge cases.

A good system implementation would be to use the task and timer method, along with flags, counters or some other indication in each critical task that is evaluated in the watchdog service task to ensure the system is functioning properly.
The example includes code for both the task with timer method and interrupt method; you will only need one of these in your application.
Example Features:

- Demonstrates 3 service methods: task with timer, interrupt and manual
- Interactive menu on serial port provides options for enabling, disabling and servicing modes

This example does not enable the watchdog in the Flash memory system configuration so that a power cycle will reset the system with the watchdog disabled. This is done to make it easier to experiment with the various watchdog modes of operation. If you wish to have the watchdog persistently enabled through power cycles without explicitly enabling it in your application, the watchdog must be enabled in the configuration system.

13.32.22 WAV File Audio Player

Supported Platforms: MOD5441x, NANO54415
This example demonstrates how to play audio files (uncompressed WAV) using the onboard Digital to Analog Converters (DAC) on the MCF5441X processor. It is fully capable of playing 8 bit and down converted 16 and 32 bit stereo audio at 44100 samples per second. It has minimal support for 24 bit samples, treating it as 8 bit data with sample padding.

Before attempting to build the application, you will first need to extract the "wav_data_srcs.zip" archive, containing the sample audio files converted to data arrays. The originals used to create these source files are found in the archive 'SD_card.zip'. This contents directory are also intended to be placed directly onto an (micro)SD card and files be played from it.
IMPORTANT: Since these files are external references in main.cpp, they must be aligned on an even boundary. One way to do this is the aligned attribute. For example: uint8_t sinewav_440[] **attribute**((aligned(16))) = { ..... }
You can see the full declaration in the nbwav_44k .cpp file

13.32.23 MOD5441x Only

MOD5441x Platform Examples

- MOD5441x Multiple EFFS Flash Card Test
- MOD5441x Factory application
- MOD5441x Rapid GPIO
- MOD5441x USB Mass Storage Device

13.32.24 MOD5441x Multiple EFFS Flash Card Test

Supported Platforms: MOD5441x
Application to demonstrate how to access both the microSD flash card on the module as and the external flash card on the NetBurner MOD-DEV-70development board.

13.32.25 MOD5441x Factory application

Supported Platforms: MOD5441x
Application that is pre-loaded on MOD5441x modules at the factory. Features include:

- Web server
- FTP server
- EFFS flash file system access to the development board flash card socket

This example requires that you edit the project properties to add the FatFile library.

13.32.26 MOD5441x Rapid GPIO

Supported Platforms: MOD5441x
Example program for ColdFire MCF5441x Rapid GPIO capability on the MOD5441x core module.
The RGPIO module provides 16-bits of high-speed GPIO functionality, mapped to the processor's bus. The key features of this module include:

- 16 bits of high-speed GPIO functionality connected to the processor's local 32-bit bus
- Memory-mapped device connected to the ColdFire core's local bus
- Support for all access sizes: byte, word, and longword
- All reads and writes complete in a single data phase cycle for zero wait-state response
- Data bits can be accessed directly or via alternate addresses to provide set, clear, and toggle functions
- Alternate addresses allow set, clear, toggle functions using simple store operations without the need for read-modify-write references
- Unique data direction and pin enable control registers
- Pin toggle rates typically 1.5–3.5x faster than comparable pin mapped onto peripheral bus

This example:

- Toggles RGPIO signal 0 (module pin 37) 50 times as fast as possible at 1 second intervals.
- The time from rising edge to rising edge of a pulse measures 16ns (62.5MHz).
- The processor slew rate must be set to maximum to acheive a speed this fast and you will
- need a fast scope and probe with a very short (1 inch) ground connection to measure it.

There are 3 functions illustrating the various methods to create the pulses in order of fastest to slowest:

- Assembly code
- Hybrid of assembly and C
- C/C++ code

13.32.27 MOD5441x USB Mass Storage Device

Supported Platforms: MOD5441x
Required Hardware: -MOD54415 or MOD54417 -NBUSB-ADPT-100R: USB development adapter -MOD-DEV-70↩
CR: development board
Very simple example program that demonstrates USB Mass Storage Device functionality.
This example demonstrates:

- Using USB Mass Storage Device Host Mode to connect to external USB flash drives
- Reading files and directories from a USB flash drive
- Writing files to USB flash drives
- Deleting files on USB flash drives

13.32.28 NANO54415 Only

NANO54415 Platform Examples

- NANO54415 Factory App

13.32.29 NANO54415 Factory App

Supported Platforms: NANO54415
Application that is pre-loaded at the factory. Features include:

- Web server
- FTP server
- EFFS flash file system access to the development board flash card socket

This example requires that you edit the project properties linker settings to add the `FatFile` library.

13.32.30 SB800EX Only

SB800EX Platform Examples

- Serial Transceiver Test
- SB800EX Diagnostic Monitor

13.32.31 Serial Transceiver Test

Supported Platforms: SB800EX
Application to demonstrate how to configure the SB800EX software programmable transceivers.
The SB800EX has the ability to switch between RS-232, RS-422 and RS-485 in software. This means customers do not need to remove the cover and manually configure jumpers. This is a very simple program demonstrating how you can modify these settings in your application.

13.32.32 SB800EX Diagnostic Monitor

Supported Platforms: SB800EX
Adds a web page diagnostic monitor and log file

13.32.33 MODM7AE70, SBE70LC

Platform specific examples for Microchip ARM SAME70 based products: MODM7AE70 and SBE70LC.

- GPIO Command Server
- I2C
- SPI
- Timers
- Watchdog Timer
- MODM7AE70
- SBE70LC

13.32.34 GPIO Command Server

SUPPORTED_PLATFORMS = MODM7AE70 SBE70LC
This example uses the Command Processor Library to establish a GPIO server that provides external control and readback functionality for general purpose input/outputs.
The example establishes the GPIO server using a telnet server by default, but it can also be configured to operate through the serial port or a TCP connection.
A detailed description of the applications features and command set can be found on the application's webpage. That can be accessed by running the application and accessing the device with a web browser.

13.32.35 I2C

MODM7AE70, SBE70LC I2C Examples

- I2C Address Scan
- I2C PicKit Demo Board

13.32.35.1 I2C Address Scan

Supported Platforms: MODM7AE70 SBE70LC
Scans the I2C bus for devices and display their addresses.
ScanI2CBus sends out a request on the I2C bus and waits for an ack, scanning each address in the address space. If a device responds, the device address is reported.
This utility is very helpful to:

- Verify the hardware is connected correctly.
- Understanding the addressing in I2C device data sheets. Some data sheets include the read/write bit as part of the address, and some do not.

13.32.35.2 I2C PicKit Demo Board

Supported Platforms: MODM7AE70 SBE70LC
Microchip Pickit Serial I2C Demo Board Driver
Demonstrates the use of 5 Microchip I2C Peripherals:

- 24LC0B EEPROM, 2Kbit (256 x 8)
- MCP9800 Temperature sensor

- MCP23008 GPIO port expander, connected to LEDs
- TC1321 10-bit DAC
- MCP3221 12-bit ADC

13.32.36 SPI

These SPI examples are common to the MODM7AE70 and SBE70LC However, there are additional SPI examples in the MODM7AE70 and SBE70 folders, such as the MODM7AE70 QuadSPI and USART SPI peripherals.

- SPI PicKit Demo Board

13.32.36.1 SPI PicKit Demo Board

Microchip Pickit Serial SPI Demo Board Driver
Supported Platforms: MODM7AE70, SBE70LC
Demonstrates the use of 7 Microchip SPI Peripherals:

- 25LC020A 2K SPI Bus Serial EEPROM
- TC77-5.0 Thermal Sensor with SPI Interface
- MCP3201 2.7V 12-Bit A/D Converter with SPI Serial Interface
- MCP4822 12-Bit DAC with Internal VREF and SPI Interface
- MCP41010 Single/Dual Digital Potentiometer with SPI Interface
- MCP6S92 Single-Ended, Rail-to-Rail I/O, Low-Gain PGA
- MCP23S08 8-Bit I/O Expander with Serial Interface

13.32.37 Timers

SAME70 Timer Platform Examples

- Timer Capture
- Chaining Timers
- Timer Waveform Output

13.32.37.1 Timer Capture

Supported Platforms: MODM7AE70 SBE70LC
Timer Counter Capture Example
This example will generate a pulse waveform on Timer 11, and capture the waveform on Timer 7.
For the MODM7AE70: The timer 11 output is on P2.28, and the timer 7 capture is on P1.4. You will need to add a jumper between these two pins. After the capture is complete, the frequency and duty cycle of the captured waveform will be displayed.
For the SBE70LC The timer 11 output is on J1.4, and the timer 2 capture is on J4.6. You will need to add a jumper between these two pins. After the capture is complete, the frequency and duty cycle of the captured waveform will be displayed.
This is a good example to demonstrate how to:

- Learn the structure of the SAME70 Timer Counter peripheral
- Produce a pulse output waveform of various frequencies and duty cycles
- Capture and count waveform pulses
- Implement a peripheral interrupt service routine

SAME70 Timer Architecture

- There are a total of 12 16-bit timers, numbered 0 - 11.
- They are divided up into 4 blocks. Each block is called a Timer Counter: TC0, TC1, TC2, TC3.
- Each timer within a Timer Counter block is called a Timer Counter Channel. While the timers number from 0 to 11, each Timer Counter Channel within a Timer Counter block is numbered from 0 to 2. For example, Timer Counter Channel 11 is in Timer Counter group TC3, with Timer Counter Channel number 2.
- While all Timer Counter Channels can be used, not all signal pins are brought out to the headers on the MODM7AE70/SBE70LC.
- The diagrams in the Timer Counter (TC) section of the SAME70 Reference Manual describe the structure of the Timer Counters.

13.32.37.2 Chaining Timers

Supported Platforms: MODM7AE70 SBE70LC
Timer Chaining Example
This example will chain timers 9, 10 and 11 to create a 48-bit timer. The output pulse waveform will appear on timer 11 as indicated in the pin tables below.
Timer channels 9, 10 and 11 are chained together to create a 48-bit timer. The default settings for the chained timers LineA are P2.28 for the MODM7AE70, and J1.4 for the SBE70LC. A GPIO output is used to output a pulse signal on each timer interrupt. P2.30 for the MODM7AE70, and J1.3 for the SBE70LC.

Warning

P2.28 on the NetBurner MOD-DEV-70 development board also connects to the SD Flash socket. The socket must be empty for this example to run properly, or you should select alternate pins for the timer functions.

See also: Timer Waveform Capture example.
This is a good example to demonstrate how to:

- Learn the structure of the SAME70 Timer Counter peripheral
- Produce a pulse output waveform of various frequencies and duty cycles

SAME70 Timer Architecture

- There are a total of 12 16-bit timers, numbered 0 - 11.
- They are divided up into 4 blocks. Each block is called a Timer Counter: TC0, TC1, TC2, TC3.
- Each timer within a Timer Counter block is called a Timer Counter Channel. While the timers number from 0 to 11, each Timer Counter Channel within a Timer Counter block is numbered from 0 to 2. For example, Timer Counter Channel 11 is in Timer Counter group TC3, with Timer Counter Channel number 2.
- While all Timer Counter Channels can be used, not all signal pins are brought out to the headers on the MODM7AE70. The table below describes the header pinout.
- The diagrams in the Timer Counter (TC) section of the SAME70 Reference Manual describe the structure of the Timer Counters.

Timer signals on MODM7AE70 header pins:

```
Timer 0
   P2.19   LineB TIOB0
   P2.42   Clock TCLK0

 Timer 1
   P2.16   Clock TCLK1

Timer 2
   P2.17   LineA TIOA2
   P2.18   LineB TIOB2
   P2.20   Clock TCLK2
```

```
Timer 5
   P1.7   LineB TIOB5

Timer 7
   P1.4   LineA TIOA7

Timer 8
   P1.6   Clock TCLK8     Note: cannot be used if EBI is enabled
   P1.8   LineA TIOA8     Note: cannot be used if EBI is enabled
   P2.6   LineB TIOB8

Timer 11
   P2.15  Clock TCLK11
   P2.28  LineA TIOA11
   P2.25  LineB TIOB11
```

Timer signals on SBE70LC header pins:

```
Timer 11
   J1.4   LineA TIOA11
   J1.7   LineB TIOB11
```

Note

Timer 11 signals are alternate functions to SPI0 MOSI and Clock

13.32.37.3 Timer Waveform Output

Supported Platforms: MODM7AE70 SBE70LC
Timer Counter Waveform Output Example
This example will generate a pulse waveform on Timer 11. The timer 11 output is on P2[28] on the MODM7AE70, and J1[4] on the SBE70LC
See also: Timer Waveform Capture example.
This is a good example to demonstrate how to:

- Learn the structure of the SAME70 Timer Counter peripheral
- Produce a pulse output waveform of various frequencies and duty cycles

SAME70 Timer Architecture

- There are a total of 12 16-bit timers, numbered 0 - 11.
- They are divided up into 4 blocks. Each block is called a Timer Counter: TC0, TC1, TC2, TC3.
- Each timer within a Timer Counter block is called a Timer Counter Channel. While the timers number from 0 to 11, each Timer Counter Channel within a Timer Counter block is numbered from 0 to 2. For example, Timer Counter Channel 11 is in Timer Counter group TC3, with Timer Counter Channel number 2.
- While all Timer Counter Channels can be used, not all signal pins are brought out to the headers on the MODM7AE70. The table below describes the header pinout.
- The diagrams in the Timer Counter (TC) section of the SAME70 Reference Manual describe the structure of the Timer Counters.

13.32.38 Watchdog Timer

Supported Platforms: MODM7AE70 SBE70LC
The Watchdog example shows how to use the watchdog on the SAME70.

13.32.39 MODM7AE70

MODM7AE70 Platform Examples

- Analog to Digital (ADC)

- CAN Send and Receive
- EBI Page Control
- EBI Page Control - Simple
- GPIO - Simple
- I2C
- Factory Application
- Interrupts
- Open All UARTs
- PWM
- NTP & Real-Time Clock
- Serial Callback Function
- SPI
- SSC I2S Interface
- Timers
- DACC Tone Generator
- Watchdog Timer

13.32.39.1 Analog to Digital (ADC)

Supported Platforms: MODM7AE70
The Simple ADC example implements basic analog input functions of the MODM7AE70.
The MODM7AE70 offers two Analog Front-End Controllers (AFEC) that are based on an Analog Front-End (AFE) cell. The MODM7AE70 provides seven 12-bit Analog-to-Digital (ADC) lines available through the P2 header. The SBE70LC provides 8 12-bit ADC lines available between the JP1 and JP3 headers. Both platform's have one additional ADC line that is connected to the CPU's internal temperature sensor – it cannot be reconfigured for other applications.
An ADC voltage reference is required for accurate ADC readings. The MODM7AE70 ADC voltage reference is accessible through pin P2[5]. If using the MODM7AE70 with a MOD-DEV-70 development board, a jumper can be applied to the JP5 header on the MOD-DEV-70 to apply a 3.3V reference. The SBE70LC ADC voltage reference is accessible through the JP3[1] pin.
The following table maps the available ADC lines on the MODM7AE70 to the corresponding pins on the P2 header: +-------+----------+----------+ | Pin | AFE Module | ADC Channel | +-------+----------+----------+ | P2[3] | 0 | 10 | +-------+----------+----------+ | P2[6] | 1 | 3 | +-------+----------+----------+ | P2[7] | 0 | 0 | +-------+----------+----------+ | P2[8] | 0 | 6 | +-------+----------+----------+ | P2[21] | 0 | 1 | +-------+----------+----------+ | P2[29] | 0 | 5 | +-------+----------+----------+ | P2[38] | 0 | 2 | +-------+----------+----------+ | ∗Temp | 0 | 11 | +-------+----------+----------+ Temperature sensor internal to the CPU, not available through the pin header
Note that AFE module 0, channel 1 (P2[3]) collides with serial port 0 receive. If using this ADC channel (like this example does), the application will not be able to receive data on serial port 0. It can still print to serial port 0, though.
The following table maps the available ADC lines on the SBE70LC to the corresponding pins on the JP1 and JP3 headers: +--------+----------+----------+ | Pin | AFE Module | ADC Channel | +--------+----------+----------+ | JP1[5] | 0 | 2 | +--------+----------+----------+ | JP1[10] | 1 | 0 | +--------+----------+----------+ | JP1[11] | 0 | 10 | +--------+----------+----------+ | JP1[13] | 0 | 1 | +--------+----------+----------+ | JP1[14] | 0 | 5 | +--------+----------+----------+ | JP3[3] | 1 | 3 | +--------+----------+----------+ | JP3[5] | 0 | 0 | +--------+----------+----------+ | JP3[7] | 0 | 6 | +--------+----------+----------+ | ∗Temp | 0 | 11 | +--------+----------+----------+ Temperature sensor internal to the CPU, not available through the pin header

13.32.39.2 CAN Send and Receive

Supported Platforms: MODM7AE70
A very simple CAN example to send and receive packets:

- Creates a CAN receive task
- Serial port menu with options to send, receive, manipulate RTR filters
- CAN communication requires 2 CAN devices. As a test you can send/receive by jumpering the CAN pins together

13.32.39.3 EBI Page Control

Supported Platforms: MODM7AE70
The EBI Page Control example implements a transparent Page Fault based page manager for the SAME70's External Bus Interface. It is primarily intended for use with systems requiring a full 24-bit address bus or for multiplexing a specific chip select against multiple identical devices.

13.32.39.4 EBI Page Control - Simple

Supported Platforms: MODM7AE70
This example demonstrates basic use of the SAME70's External Bus Interface (EBI). To implement a bus interface requires an understanding of the EBI signals, which are provided in the Microchip SAME70 Reference Manual and Data Sheet location in the "\nburn\docs" folder of your installation. Section 33 "External Bus Interface" provides an overview of EBI operation. The most common interface is desribed in section 35 "Static Memory Controller (SMC). The example demonstrates configuration of the EBI via the NetBurner EBI driver, performing a single read or write operation on the bus.

13.32.39.5 GPIO - Simple

Supported Platforms: MODM7AE70
Very simple GPIO example that blinks a LED on P2[15] of the MOD-DEV-70 development board. There is a more advanced multi-platform example in the `\nburn\examples` folder.

13.32.39.6 I2C

MODM7AE70, SBE70LC I2C Examples

- I2C Address Scan
- I2C PicKit Demo Board

13.32.39.7 I2C Address Scan

Supported Platforms: MODM7AE70 SBE70LC
Scans the I2C bus for devices and display their addresses.
ScanI2CBus sends out a request on the I2C bus and waits for an ack, scanning each address in the address space. If a device responds, the device address is reported.
This utility is very helpful to:

- Verify the hardware is connected correctly.
- Understanding the addressing in I2C device data sheets. Some data sheets include the read/write bit as part of the address, and some do not.

13.32.39.8 I2C PicKit Demo Board

Supported Platforms: MODM7AE70 SBE70LC
Microchip Pickit Serial I2C Demo Board Driver
Demonstrates the use of 5 Microchip I2C Peripherals:

- 24LC0B EEPROM, 2Kbit (256 x 8)

- MCP9800 Temperature sensor
- MCP23008 GPIO port expander, connected to LEDs
- TC1321 10-bit DAC
- MCP3221 12-bit ADC

13.32.39.9 Factory Application

Supported Platforms: MODM7AE70
Application that is pre-loaded on MODM7AE70 modules at the factory. Features include:

- Web server
- FTP server
- EFFS flash file system access to the development board flash card socket

This example requires that you edit the project properties to add the FatFile library.

13.32.39.10 Interrupts

MODM7AE70 Interrupt Platform Examples

- External Pin IRQ
- Pin Interrupt Request

13.32.40 External Pin IRQ

Supported Platforms: MODM7AE70
This example is based on the PinIrq example, which shows how to use the interrupt capabilities of the PIO controller and the PinIrq functions for the MODM7AE70.
This example is much simpler. It uses a single pin as a IRQ input, and a second pin to generate IRQ requests once every second. These 2 pins will need to be jumpered on the development board in order for the example to run.
When configuring the IRQ priority for a pin, the configuration affects the entire port that the GPIO pin belongs to. For example, this application configures pin P2[8] for an IRQ input. As can be seen in the table below, P2[8] belongs to PIO_PORT_A. Therefore, the IRQ priority configuration will also affect pins P2[9], P2[12] and so on. The other ports are unaffected by the IRQ priority configuration of PIO_PORT_A in this example.
The table below maps the pins on the P1 and P2 headers of the MODM7AE70 to the port that they belong to.
+===+ | PIO_PORT_A | PIO_↩
PORT_B | PIO_PORT_C | PIO_PORT_D | PIO_PORT_E | +===
| P2[8] | P2[3] | P2[6] | P2[7] | P1[20] | +----------—+----------—+----------—+----------—+----------—+ | P2[9] | P2[4] | P1[4]
| P2[10] | P1[22] | +----------—+----------—+----------—+----------—+----------—+ | P2[12] | P2[11] | P1[6] | P2[15] | P1[23] |
+----------—+----------—+----------—+----------—+----------—+ | P2[13] | P2[12] | P1[11] | P2[23] | P1[24] | +----------—+--------
—+----------—+----------—+----------—+ | P2[16] | P2[22] | P1[12] | P2[24] | P1[25] | +----------—+----------—+----------—+----
------—+----------—+ | P2[17] | P2[29] | P1[13] | P2[25] | P1[27] | +----------—+----------—+----------—+----------—+----------—+
| P2[18] | P2[38] | P1[14] | P2[26] | | +----------—+----------—+----------—+----------—+----------—+ | P2[19] | | P1[15] |
P2[27] | | +----------—+----------—+----------—+----------—+----------—+ | P2[20] | | P1[16] | P2[28] | | +----------—+----------—
+----------—+----------—+----------—+ | P2[21] | | P1[17] | P2[30] | | +----------—+----------—+----------—+----------—+----------—+
| P2[31] | | P1[18] | P2[37] | | +----------—+----------—+----------—+----------—+----------—+ | P2[32] | | P1[19] | P2[41] | |
+----------—+----------—+----------—+----------—+----------—+ | P2[33] | | P1[21] | P2[44] | | +----------—+----------—+----------—
+----------—+----------—+ | P2[34] | | P1[32] | P1[7] | | +----------—+----------—+----------—+----------—+----------—+ | P2[35] |
| P1[33] | P1[9] | | +----------—+----------—+----------—+----------—+ | P2[36] | | P1[34] | | | +----------—+--------
—+----------—+----------—+----------—+ | P2[39] | | P1[35] | | | +----------—+----------—+----------—+----------—+----------—+ |
P2[40] | | P1[36] | | | +----------—+----------—+----------—+----------—+----------—+ | P2[42] | | P1[37] | | | +----------—+------
---—+----------—+----------—+----------—+ | P2[43] | | P1[38] | | | +----------—+----------—+----------—+----------—+----------—+ |
P2[45] | | P1[39] | | | +----------—+----------—+----------—+----------—+----------—+ | P2[47] | | P1[40] | | | +----------—+------
---—+----------—+----------—+----------—+ | P2[48] | | P1[41] | | | +----------—+----------—+----------—+----------—+----------—+ |
P1[5] | | P1[42] | | | +----------—+----------—+----------—+----------—+----------—+ | P1[10] | | P1[43] | | | +----------—+--------
—+----------—+----------—+----------—+ | P1[26] | | P1[44] | | | +----------—+----------—+----------—+----------—+----------—+ |

P1[29] | | P1[45] | | | +----------—+----------—+----------—+----------—+----------—+ | P1[31] | | | | | +----------—+----------—+----------—+----------—+----------—+ | P1[46] | | | | | +----------—+----------—+----------—+----------—+----------—+ | P1[47] | | | | | +----------—+----------—+----------—+----------—+----------—+

13.32.41 Pin Interrupt Request

Supported Platforms: MODM7AE70
The PinIrq example shows how to use the interrupt capabilities of the PIO controller and the PinIrq functions for the MODM7AE70.

13.32.41.1 Open All UARTs

Supported Platforms: MODM7AE70
Open all serial ports on MODM7AE70 SOM Platform
There can be a maximum of 7 serial port configured on the device. Note that each signal pin on the microprocessor can be configured for up to 5 different functions, so to achieve the maximum number of serial ports some functions/ peripherals will be unavailable.
The SAME70 processor has two peripherals that can be used as a UART function: Three USARTs, representing Serial Port numbers 0 and 1 (USART numbers 0 - 1) Four UARTs, representing Serial Port numbers 3, 4, 5 and 6 (UART numbers 0 - 4) This mapping is described in the MODM7AE70 data sheet.
Serial Port P1 Pin P2 Pin Periph Mode USART/UART
0 RX 3 C USART 0 RX 0 TX 4 C USART 0 TX 0 CTS 29 C USART 0 CTS 0 RTS 38 C USART 0 RTS Note: Serial port 0 is stdio by default
1 RX 21 A USART 1 RX 1 TX 22 D USART 1 TX 1 RTS 32 A USART 1 RTS 1 CTS 33 A USART 1 CTS 1 DSR 16 A USART 1 DSR 1 DCD 17 A USART 1 DCD 1 DTR 18 A USART 1 DTR 1 RI 20 A USART 1 RI
2 RX 34 A UART 0 RX NOTE: tied to LED on mod-dev-70 2 TX 35 A UART 0 TX
3 RX 12 C UART 1 RX, PA5∗ 3 TX 31 C UART 1 TX 3 TX 42 C UART 1 TX (Alternate) 3 TX 44 D UART 1 TX (Alternate)
4 RX 41 C UART 2 RX 4 TX 44 C UART 2 TX
5 RX 23 A UART 3 RX 5 TX 7 A UART 3 TX 5 TX 24 B UART 3 TX (Alternate)
6 RX 10 C UART 4 RX 6 TX 7 C UART 4 TX CAUTION: P1.7 Cannot be used if the external address/data bus interface (EBI) is enabled.

13.32.41.2 PWM

Supported Platforms: MODM7AE70
Demonstrates the use of the PWM peripheral.

13.32.41.3 NTP & Real-Time Clock

Supported Platforms: MODM7AE70
NTP-SYSTEM-RTC DEMO APPLICATION
HARDWARE SETUP: Tested with MODM7AE70 on a MOD-DEV-70 hardware rev 1.93.
This example demonstrates the following features:

- Updates the system time with the current UTC time pulled from an available NTP server (pool.ntp.org). If your module is set to static IP address settings, then you will need to manually assign a DNS server address for this feature to work.
- Take the current system time and save it to the real-time clock (RTC). The RTC will then be running in sync with the system time.
- Take the current time in the RTC and set the system time information. The system time will then be running in sync with the RTC.
- When the device or module is reset or powered on again after a recent power loss, the system time will be reset, but the RTC can continue running until the super-capacitor is discharged. The software-reset option is available to demonstrate the RTC's ability to retain saved time information once it is properly set.
- Set the local time zone information with a call to tzsetchar(). For more more information on this function, refer to "Chapter 14 - NBTime Library" of the NetBurner Runtime Libraries PDF document. In this example, Pacific time zone information is used as a demonstration.

- Display the current system and RTC times, and local time zone information if applicable. It is preferred that the RTC store the UTC time, not local time.
- In order to take advantage of setting and getting time information from the real-time clock (RTC), a device that contains an RTC component or a module that is mounted on a development board with an RTC component must be used. Current RTC components supported in this example at the time of this writing are the Intersil X1226 and NXP PCF8563.

13.32.41.4 Serial Callback Function

Supported Platforms: MODM7AE70
The Serial Empty Callback example demonstrates a robust process for triggering a specified action upon complete transmission of all data written to a specified serial port. The expected use case for this would be disabling a transmitter connected to a shared physical layer such as with RS485 or a wireless link (when using one of the device's UART serial ports).
As an aside, it should be noted that when enabling the Serial485HalfDupMode for one of the MODM7AE70s *USART* serial ports, the underlying hardware module will automatically toggle the RTS line for enabling and disabling the transmitter.

13.32.41.5 SPI

MODM7AE70 SPI Platform Examples

- Serial-to-SPI QuadSPI
- Serial-to-SPI USART

13.32.42 Serial-to-SPI QuadSPI

Supported Platforms: MODM7AE70
The Serial-to-SPI example demonstrates the basic configuration of the SPI driver and sending/receiving data over an SPI interface. This example demonstrates how to use the driver for the QuadSPI peripheral in single-bit, master mode SPI.

13.32.43 Serial-to-SPI USART

Supported Platforms: MODM7AE70
The Serial-to-SPI example demonstrates the basic configuration of the SPI driver and sending/receiving data over an SPI interface. This example demonstrates how to use the driver for the USART peripheral in single-bit, master mode SPI.

13.32.43.1 SSC I2S Interface

Supported Platforms: MODM7AE70
SSC_I2S implements an I2S interface using the Synchronous Serial Controller and uses it to communicate with a WM8904 Audio Codec.

13.32.43.2 Timers

MODM7AE70 Timer Platform Examples

- Timer Capture
- Timer Waveform Output
- Chaining Timers

13.32.44 Timer Capture

Supported Platforms: MODM7AE70 SBE70LC
Timer Counter Capture Example
This example will generate a pulse waveform on Timer 11, and capture the waveform on Timer 7.
For the MODM7AE70: The timer 11 output is on P2.28, and the timer 7 capture is on P1.4. You will need to add a jumper between these two pins. After the capture is complete, the frequency and duty cycle of the captured waveform will be displayed.
For the SBE70LC The timer 11 output is on J1.4, and the timer 2 capture is on J4.6. You will need to add a jumper between these two pins. After the capture is complete, the frequency and duty cycle of the captured waveform will be displayed.
This is a good example to demonstrate how to:

- Learn the structure of the SAME70 Timer Counter peripheral
- Produce a pulse output waveform of various frequencies and duty cycles
- Capture and count waveform pulses
- Implement a peripheral interrupt service routine

SAME70 Timer Architecture

- There are a total of 12 16-bit timers, numbered 0 - 11.
- They are divided up into 4 blocks. Each block is called a Timer Counter: TC0, TC1, TC2, TC3.
- Each timer within a Timer Counter block is called a Timer Counter Channel. While the timers number from 0 to 11, each Timer Counter Channel within a Timer Counter block is numbered from 0 to 2. For example, Timer Counter Channel 11 is in Timer Counter group TC3, with Timer Counter Channel number 2.
- While all Timer Counter Channels can be used, not all signal pins are brought out to the headers on the MODM7AE70/SBE70LC.
- The diagrams in the Timer Counter (TC) section of the SAME70 Reference Manual describe the structure of the Timer Counters.

13.32.45 Timer Waveform Output

Supported Platforms: MODM7AE70 SBE70LC
Timer Counter Waveform Output Example
This example will generate a pulse waveform on Timer 11. The timer 11 output is on P2[28] on the MODM7AE70, and J1[4] on the SBE70LC
See also: Timer Waveform Capture example.
This is a good example to demonstrate how to:

- Learn the structure of the SAME70 Timer Counter peripheral
- Produce a pulse output waveform of various frequencies and duty cycles

SAME70 Timer Architecture

- There are a total of 12 16-bit timers, numbered 0 - 11.
- They are divided up into 4 blocks. Each block is called a Timer Counter: TC0, TC1, TC2, TC3.
- Each timer within a Timer Counter block is called a Timer Counter Channel. While the timers number from 0 to 11, each Timer Counter Channel within a Timer Counter block is numbered from 0 to 2. For example, Timer Counter Channel 11 is in Timer Counter group TC3, with Timer Counter Channel number 2.
- While all Timer Counter Channels can be used, not all signal pins are brought out to the headers on the MODM7AE70. The table below describes the header pinout.
- The diagrams in the Timer Counter (TC) section of the SAME70 Reference Manual describe the structure of the Timer Counters.

13.32.46 Chaining Timers

Supported Platforms: MODM7AE70 SBE70LC
Timer Chaining Example
This example will chain timers 9, 10 and 11 to create a 48-bit timer. The output pulse waveform will appear on timer 11 as indicated in the pin tables below.
Timer channels 9, 10 and 11 are chained together to create a 48-bit timer. The default settings for the chained timers LineA are P2.28 for the MODM7AE70, and J1.4 for the SBE70LC. A GPIO output is used to output a pulse signal on each timer interrupt. P2.30 for the MODM7AE70, and J1.3 for the SBE70LC.

Warning

P2.28 on the NetBurner MOD-DEV-70 development board also connects to the SD Flash socket. The socket must be empty for this example to run properly, or you should select alternate pins for the timer functions.

See also: Timer Waveform Capture example.
This is a good example to demonstrate how to:

- Learn the structure of the SAME70 Timer Counter peripheral
- Produce a pulse output waveform of various frequencies and duty cycles

SAME70 Timer Architecture

- There are a total of 12 16-bit timers, numbered 0 - 11.
- They are divided up into 4 blocks. Each block is called a Timer Counter: TC0, TC1, TC2, TC3.
- Each timer within a Timer Counter block is called a Timer Counter Channel. While the timers number from 0 to 11, each Timer Counter Channel within a Timer Counter block is numbered from 0 to 2. For example, Timer Counter Channel 11 is in Timer Counter group TC3, with Timer Counter Channel number 2.
- While all Timer Counter Channels can be used, not all signal pins are brought out to the headers on the MODM7AE70. The table below describes the header pinout.
- The diagrams in the Timer Counter (TC) section of the SAME70 Reference Manual describe the structure of the Timer Counters.

Timer signals on MODM7AE70 header pins:

```
Timer 0
   P2.19   LineB TIOB0
   P2.42   Clock TCLK0

 Timer 1
   P2.16   Clock TCLK1

Timer 2
   P2.17   LineA TIOA2
   P2.18   LineB TIOB2
   P2.20   Clock TCLK2

 Timer 5
    P1.7   LineB TIOB5

 Timer 7
    P1.4   LineA TIOA7

 Timer 8
    P1.6   Clock TCLK8     Note: cannot be used if EBI is enabled
    P1.8   LineA TIOA8     Note: cannot be used if EBI is enabled
    P2.6   LineB TIOB8

 Timer 11
    P2.15  Clock TCLK11
    P2.28  LineA TIOA11
    P2.25  LineB TIOB11
```

Timer signals on SBE70LC header pins:

```
Timer 11
   J1.4   LineA TIOA11
   J1.7   LineB TIOB11
```

Note

Timer 11 signals are alternate functions to SPI0 MOSI and Clock

13.32.46.1 DACC Tone Generator

Supported Platforms: MODM7AE70
Digital to Analog converter (referred to as "DACC" in the SAME70 manuals) example implementing basic DACC interface and timer triggered conversions.

13.32.46.2 Watchdog Timer

Supported Platforms: MODM7AE70 SBE70LC
The Watchdog example shows how to use the watchdog on the SAME70.

13.32.47 SBE70LC

SBE70LC Platform Examples

- GPIO - Simple
- PWM

13.32.47.1 GPIO - Simple

Supported Platforms: SBE70LC
Very simple GPIO example that toggles the GPIO on J1[3] of the SBE70LC development board. There is a more advanced multi-platform example in the `\nburn\examples` folder.

13.32.47.2 PWM

Supported Platforms: SBE70LC
Demonstrates the use of the PWM peripheral. The PWM module is used to generating a 5kHz square wave with a 50% duty cycle on pin J1[7] and a 10 Hz square wave with a 60% duty cycle on pin J1[15].

13.32.48 SOMRT1061

SOMRT1061 Platform Examples

- Analog to Digital (ADC)
- Open All UARTs
- I2C
- SEMC, Smart External Memory Controller

13.32.49 Analog to Digital (ADC)

Supported Platforms: SOMRT1061
Simple ADC example implementing basic analog input functions of the SOMRT1061

13.32.50 Open All UARTs

Supported Platforms: SOMRT1061
Open all serial ports on SOMRT1061 Platform
There can be a maximum of 7 serial port configured on the device. Note that each signal pin on the microprocessor can be configured for up to 5 different functions, so to achieve the maximum number of serial ports some functions/ peripherals will be unavailable.
Serial Port Pin Number Peripheral Mode LPUART Comments for use with DEV-S61-100CR
0 TX 8 2 LPUART 3 TX 0 RX 7 2 LPUART 3 RX 0 CTS 6 2 LPUART 3 CTS 0 RTS 4 2 LPUART 3 RTS Note: Serial port 0 is stdio by default
1 TX 42 2 LPUART 2 TX 1 RX 43 2 LPUART 2 RX Connected to GND through 0.1uF cap. High bus speed not recommended. 1 CTS 40 2 LPUART 2 CTS 1 RTS 41 2 LPUART 2 RTS Connected to GND through 0.1uF cap. High bus speed not recommended.
2 TX 88 2 LPUART 4 TX 2 RX 87 2 LPUART 4 RX 2 CTS 3 2 LPUART 4 CTS 2 RTS 2 2 LPUART 4 RTS
3 TX 86 2 LPUART 5 TX Shares trace with RTC CLK OUT. Cannot be used if RTC clock output is enabled (not enabled by default). 3 RX 85 2 LPUART 5 RX Shares trace with NBWIFI enable pin. Cannot be used simultaneously with NBWIFI. 3 CTS 81 2 LPUART 5 CTS Shares trace with RGB blue LED. High bus speed not recommended. 3 RTS 82 2 LPUART 5 RTS Shares trace with RGB green LED. High bus speed not recommended.
4 TX 84 2 LPUART 6 TX Only supports RS485. 4 RX 83 2 LPUART 6 RX Only supports RS485. 4 CTS 78 2 LPUART 6 CTS Only supports RS485. Shares trace with LED0. High bus speed not recommended. 4 RTS 80 2 LPUART 6 RTS Only supports RS485.
5 TX 77 2 LPUART 7 TX Shares trace with RGB red LED. HIGH bus speed not recommended. 5 RX 76 2 LPUART 7 RX Shares trace with Enet 2. Ethernet driver source modifications can be made for testing while not mounted in DEV-S61-100CR WARNING: LPUART7 RX cannot be used while mounted in the DEV-S61-100CR
6 TX 29 2 LPUART 8 TX Shares trace with SDHC (PIN_29_USDHC1_DATA2). Trace pulled high with 47k resistor. High bus speed not recommended. Cannot be used with SD card is mounted. 6 RX 28 2 LPUART 8 RX Shares trace with SDHC (PIN_28_USDHC1_DATA3). Trace pulled high with 47k resistor. High bus speed not recommended. Cannot be used with SD card is mounted. 6 CTS 25 2 LPUART 8 CTS Shares trace with SDHC (PIN_25_↩ USDHC1_DATA0). Trace pulled high with 47k resistor. High bus speed not recommended. Cannot be used with SD card is mounted. 6 RTS 24 2 LPUART 8 RTS Shares trace with SDHC (PIN_25_USDHC1_DATA1). Trace pulled high with 47k resistor. High bus speed not recommended. Cannot be used with SD card is mounted. WARNING: LPUART 8 cannot be used if an SD card is mounted in the SD card socket.

13.32.51 I2C

SOMRT1061 I2C Examples

- I2C Address Scan
- exampleI2CPicKitSerialDemoBd-SOMRT1061

13.32.52 I2C Address Scan

Supported Platforms: SOMRT1061
Scans the I2C bus for devices and display their addresses.
ScanI2CBus sends out a request on the I2C bus and waits for an ack, scanning each address in the address space. If a device responds, the device address is reported.
This utility is very helpful to:

- Verify the hardware is connected correctly.
- Understanding the addressing in I2C device data sheets. Some data sheets include the read/write bit as part of the address, and some do not.

13.32.53 SEMC, Smart External Memory Controller

Supported Platforms: SOMRT1061
Example demonstrating the basic usage of the SOMRT1061 External Memory interface.

13.32.54 External IRQ

Supported Platforms: MOD5441x, NANO54415, MODM7AE70, SBE70LC
This example program uses a GPIO pin to trigger an external interrupt pin. A jumper must be placed from the GPIO pin to the desired interrupt input. To run the example use MTTTY or some other serial terminal to interact with the menu to trigger the interrupt and view the interrupt counter.

13.32.55 SPI

These SPI examples are common to all 3.x platforms.

Note

These SPI examples work on multiple processor/product architectures. There are additional SPI examples in the processor/product specific folders that may be of interest. For example, the MODM7AE70 has a quad SPI peripheral, but the MOD5441x does not.

- DSPI Multiplex
- Serial to SPI

13.32.56 DSPI Multiplex

Supported Platforms: MOD5441X NANO54415 MODM7AE70 SBE70LC SB800EX
This example demonstrates how to use the DSPIModule, object based DSPI driver, to use multiple bus configurations on the same DSPI hardware module without having to re-initialize the driver every time when switching configurations.

13.32.57 Serial to SPI

Supported Platforms: MOD5441X NANO54415 MODM7AE70 SBE70LC
The Serial to SPI example demonstrates the basic configuration of the SPI driver and sending and receiving data over an SPI interface.

13.33 PPP

This PPP example shows how to create two PPP server instances.
There are two types of high level connections that can be made with PPP is either direct or modem. A direct connection is usually performed with only a serial cable connection and will jump right into the PPP negotiations with the host or client. A modem connection is one where there is a modem between the PC and NetBurner. The modem PPP functions will send AT commands before any PPP negotiation to initialize the modem.
A good way to test PPP is to create a PPP interface on a computer and make a direct PPP connection to the server.

13.34 Profiler

This is a very simple profiler program. The way the NBRTOS_TIME functions operate is that there is an additional routine that runs during task switching to record the tick and tickfraction since the task swapped in (runtime since last change).
This feature has overhead and you normally only use it for development. By default, this example enabled profiling through the overload folder functionality. To enable profiling in your own application, you can copy the overload folder in to your application, or following these steps:

1. Edit `\nburn\nbrtos\include\prefef.h` and uncomment: `#define NBRTOS_TIME(1)`
2. Rebuild the system libraries

   The following functions are now available:

   uint32_t GetCurrentTaskTime(uint32_tconst TotalTicks); Returns the number of time ticks the current task has run. TotalTicks = the total number of ticks recorded.

```
void ShowTaskTimes( void );
```

Print a list of tasks, times and percentages to stdout (normally uart0)

```
void ClearTaskTimes( void );
```

Resets all task times to 0

13.35 RTOS

NetBurner RTOS (NBRTOS) Examples:

- OSCrit - Critical Section
- OSFifo - FIFO
- OSFlags
- OSMailbox
- Multiple Task User Input
- OSQueue
- OSSemaphore
- OSTaskCreate

13.35.1 OSCrit - Critical Section

This example will show how to create critical sections with the standard C type function calls OSCritEnter() and OSCritLeave(), and compare them to the C++ object type which eliminates the problem of matching each OSCritEnter() to a corresponding OSCritLeave().
While this example is trivial with only a simple global variable, critical sections need to be used for objects, such as linked lists, that could be changed by multiple tasks. In such cases, each task would use a critical section to ensure the modifications could be completed before a task switch occurred.

13.35.2 OSFifo - FIFO

This program creates two tasks and a FIFO. Messages are sentfrom one task to another using the FIFO. A timeout value for the pend function is used to illustrate the timeout feature of the FIFO.

13.35.3 OSFlags

This illustrates how OSFlags can be used to pend on multiple events. `UserMain()` creates 3 tasks, each of which will set a OSFlag after a time delay of some number of seconds. UserMain will then pend and block until ANY of the 3 flags are set. You can also modify the example to pend until ALL of the 3 flags are set.

13.35.4 OSMailbox

This program will create 2 tasks and a mailbox. The UserMain task will block for a mailbox message posted from MyTask. The output for this example is displayed through the serial debug port, which can be viewed with the MTTTY program.

13.35.5 Multiple Task User Input

This program illustrates how to create a task and use a mailbox to accept user input from a serial port. This example also illustrates how to terminate a task

1. UserMain task will wait pend on a mailbox for user input
2. UserInput task will block waiting for a user to type in data from the serial port

13.35.6 OSQueue

Queue Example Program
This program creates two tasks and a Queue. Messages are sentfrom one task to another using the Queue. A timeout value for the pend function is used to illustrate the timeout feature of the FIFO.

13.35.7 OSSemaphore

This program will create 2 tasks and a semaphore. The UserMain task will block for a semaphore posted from My↩ Task. The output for this example is displayed through the serial debug port, which can be viewed with the MTTTY program.

13.35.8 OSTaskCreate

This example demonstrates how to use two of the task creation functions:

- OSSimpleTaskCreatewName()
- OSTaskCreatewName()

13.36 Save to User Parameter Flash

The User Parameter Flash functionality was introduced in the NetBurner 2.x tools. The preferred method for non-volatile storage and settings in the 3.x tool set is the Configuration Server, which provides for multi-platform web access, application access and JSON capabilities. All of these methods are configurable. Please see the configuration users guide and the configuration examples for more information.
In contrast, the User Parameter Flash is a sector of flash memory set aside to be used as raw bulk storage. In situations such as bulk binary storage, or porting a 2.x application to 3.x with minimal changes, this method of storage can be useful.
This example to demonstrates how to read and write data to the on-board flash memory User Parameter Area. The amount of flash space is equal to one flash sector, which will vary from 8k bytes to 64k bytes depending on your platform.
Warning: Data corruption can occur under the following circumstances:

1. If power is interrupted during the flash programming process. We strongly recommend that UserParameters are not written immediately upon power-up, since it is not uncommon for a user to cycle a power switch on/off quickly every once in a while.
2. Your application writes beyond the UserParameter size allocated for your platform (8k or 64k). In this situation the application may be corrupted.

Important notes on packed versus integer alignment when using structures and classes: This example uses a structure named NV_SettingsStruct, which consists of types that are 8, 16 and 32 bits long. If you do not add an attribute tag at the end of the structure, the default will be integer-aligned (32-bits in this case). This means that padding will be added to ensure each member will be on an integer boundary.
While this can increase execution speed, it also means that the stored data will be a bit larger, and doing something like an overlay or indexing into the structure with a pointer will not work correctly. To tell the compiler not to use any padding, add `__attribute__((packed));` to the end of the structure definition as demonstrated in this example. Be aware that when using a packed structure elsewhere in your application, access is packed as well.

13.37 Serial

Serial Port Examples:

- Dual TCP To Serial
- Serial To Ethernet
- Serial HTTP Get Request
- Serial Receive Callback

- Serial to Serial
- TCP to Serial

13.37.1 Dual TCP To Serial

This sample application connects two serial ports to two different TCP ports, and continuously transfers data between them. In other words, you have a dual serial-to-Ethernet device. An easy way to test this operation is to open two telnet sessions and two MTTTY sessions as shown below. To start a telnet session on a windows pc, open a command prompt and type `telnet 10.1.1.2 2000`, where you would replace 10.1.1.2 with the IP address of your NetBurner device.
Telnet on port 2000 >----------< MTTTY on serial port 0
Telnet on port 2001 >----------< MTTTY on serial port 1
Now anything you type in the first telnet session will be sent to serial port 0, and vice versa. Anything you type in the second telnet session will be sent to serial port 1 and vice versa.

13.37.2 Serial To Ethernet

This program is designed to connect a RS-232 device to a network. It opens a serial port and an Ethernet port, then connect them.This allows any data sent to the Ethernet port to be forwarded to the serial port, and vice versa.
A connection timeout is used to check for link activity, and the connection will be closed if the timout expires. The timeout value is set with TIMEOUT_LIMIT. A value of 0 will disable the timeout feature.
Only one connection is allowed at a time. Additional client connections will be queued, and serviced as the active connection completes.
Override Timeout: When a new TCP connection is attempted while there is already an active connection, one of 3 actions can be taken:

1. Don't ever disconnect the active connection. Set `OVERRIDE_TIMEOUT to 0xFFFFFFFF`.
2. Only disconnect the existing connection if it has been idle for x seconds. Set `OVERRIDE_TIMEOUT` to the number of seconds.
3. Always disconnect the existing connection. Set `OVERRIDE_TIMEOUT` to 0.

13.37.3 Serial HTTP Get Request

Example of sending a HTTP GET request through serial port when running a serial to Ethernet application.
Hardware Configuration:
Serial Device <–serial--> NetBurner Serial to Ethernet Device <–ethernet--> Internet
In this example we will be using a SB800EX as the "Serial Device", but this would typically be a microprocessor or FPGA with a UART that needs to communicate over a network. The source code in this file is an example of what you would need to implement in your own microprocessor to send a GET request through a NetBurner device to some other device on the network/Internet runnig a web server.
The code in the main.cpp source file was tested with the SB800EX, but should work with any of the NetBurner standard devices. A serial cable was connected from port 1 of the SB800EX to port 1 of the SBL2e development board. Since the SBL2e development board DB9 connector is cross-wired for RX and TX, all you need is a straight through serial cable.
SB800EX <–serial--> SBL2e <–ethernet--> Internet
The web page contacted is myip.dnsdynamic.com, which return the IP address of the HTTP GET request.
Software Configuration for SBL2e, serial port 1:
Running the SBL2e factory serial to Ethernet application. This was done through the web page, but you can also do it through serial AT commands.

1. Incoming TCP Settings: Uncheck Listen for Incoming Network Connections
2. Outgoing TCP Settings: Make outgoing connections: If serial data received
3. Outgoing TCP Settings: Connect on network port: 80
4. Outgoing TCP Settings: Connect to this address: myip.dnsdynamic.com

13.37.4 Serial Receive Callback

Demonstrates how to register a callback function to intercept system serial port receive processing.

13.37.5 Serial to Serial

This simple program opens both serial ports and sends data from one to another using the select() function, which can pend on multiple file descriptors at one time.
The serial ports are initialized by the monitor in polled mode. To enable buffered interrupt driven mode, the serial ports are closed and then reopened with OpenSerial().

13.37.6 TCP to Serial

This is a very simple TCP to Serial example. It listens on a TCP port for a single incoming connection. Once a TCP connection is accepted, all data recevied will be sent out the serial port, and all data recevied by the serial port will be sent out the TCP connection.
A simple way to test this program is to use the Telnet program on a PC and connect to the NetBurner device. Telnet is available to run from a command prompt, or with a program such as PUTTY.
If you would like to extend this example and create a robust Serial To Ethernet application, the next step would be to add error detection and timeouts to handle a situation such as the TCP client crashing or going away without a proper TCP close connection sequence. That would result in a half open socket condition and the NetBurner device would have no way of knowing what happened.
To help combat situations such as these this application takes a simple approach of closing the connection is no activity has taken place for a period of time.

13.38 Serial Webserver

The Serial Webserver is a simple example demonstrating receiving and replying to HTTP requests over a serial port, using the NetBurner web server implementation. The example will display a "Hello World" type message on the device's index page: index.html.
An valid HTTP GET request that is compatible with this application is the following:

Note

The webserver expects a double carriage return line feed

```
'GET /INDEX.HTML HTTP/1.1\r\n\r\n'
```

The expected response should look similar to the following:

```
HTTP/1.0 200 OK
Pragma: no-cache
MIME-version: 1.0
Content-Type: text/html
Content-length: 231

<html>
<head><link rel="stylesheet" type="text/css" href="style.css"></head>
<body>
<a href="http://www.netburner.com"><img src="logo.jpg" width="200" height="104" border="0"></a>
<h1>Thank you for NetBurning!</h1>
</body>
</html>
```

13.39 SHA1 Digest

Calculate the SHA1 128-bit digest on an application. To display the digest when building the application in NBEclipse, go to the project properties and add -D to the compcode flags.

13.40 Show Network Interfaces

Example program to display IPv4 and IPv6 address information on both the serial port and on a web page. The web page will also display the addresses as links that can be clicked on to view the web page using the respective IP addresses, such as IPv4, IPv6 and AutoIP.

13.41 SOCKS5 Client

This program demonstrates how to create a client connection using the SOCKS5 protocol. This protocol is detailed in RFC 1928. The NetBurner libraries support no authorization and plain text username/password options, and provide a callback for the user to implement their own version of GSSAPI, AuthWithGssApi(). This function can be overridden to implement GSSAPI according to specific requirements.
To test this, you can use a Docker image that will run a SOCK5 server, found here: https://hub.docker.com/r/serjs/go-socks5-proxy/

13.42 SPI

These SPI examples are common to all 3.x platforms.

Note

These SPI examples work on multiple processor/product architectures. There are additional SPI examples in the processor/product specific folders that may be of interest. For example, the MODM7AE70 has a quad SPI peripheral, but the MOD5441x does not.

- DSPI Multiplex
- Serial to SPI

13.42.1 DSPI Multiplex

Supported Platforms: MOD5441X NANO54415 MODM7AE70 SBE70LC SB800EX
This example demonstrates how to use the DSPIModule, object based DSPI driver, to use multiple bus configurations on the same DSPI hardware module without having to re-initialize the driver every time when switching configurations.

13.42.2 Serial to SPI

Supported Platforms: MOD5441X NANO54415 MODM7AE70 SBE70LC
The Serial to SPI example demonstrates the basic configuration of the SPI driver and sending and receiving data over an SPI interface.

13.43 SSH

Secure Shell (SSH) Examples:

- SSH Server
- SSH Server with User Key
- SSH Client
- SSH Server with User Authorization
- SecureSerToEthFactoryApp

13.43.1 SSH Server

SSH Minimal Server Example
A SSH Server task is created that will listen for incoming SSH connections on TCP port 22. This simple example uses the SSH key stored in the NetBurner library.
To use the example:

- Compile and load the application into your NetBurner device.
- Run MTTTY and connect to USB or Serial port to view status messages and send data to the SSH Client.
- Run a SSH Client like Putty, and connect to the NetBurner device.

13.43.2 SSH Server with User Key

This example demonstrates how to use the NetBurner SSH library to create a SSH server that can utilize the SSH library key, user application compiled-in key and upload a user supplied key.
Please refer to the pdf documentation in the example directory for detailed information.

13.43.3 SSH Client

This example shows how to run a minimal SSH client application. This application will try to create an SSH connection to the server with the IP address defined by SSH_SERVER_NAME. Upon connection, it will read data from the server until the connection is closed.
To use the example:

- Find an SSH server that you'd like to connect to, and set the IP address as SSH_SERVER_NAME.
- Compile and load the application into your NetBurner device.
- Run MTTTY and connect to USB or Serial port to view status messages and send data to the SSH Client.

13.43.4 SSH Server with User Authorization

SSH Server with User Authorization Example
A SSH Server task is created that will listen for incoming SSH connections on TCP port 22. This simple example uses a key that is generated onboard the device the first time it is powered on. This key will be valid for one year. To allow the certificate to automatically renew when it's expired, enable `ENABLE_AUTOCERT_REGEN`, found in `nbrtos\include\predef.h`.
This example demonstrates how to user the UserAuthManager, which will let you maintain user password/key information that can be used when authenticating users trying to log on to your device. In this example, we store the data in the UserParam space, but the UserAuthManager will let you determine where that information is saved and loaded from with the callbacks that are passed into the object during construction.
To use the example:

- Compile and load the application into your NetBurner device.
- Run MTTTY and connect to USB or Serial port to view status messages and send data to the SSH Client.
- Run a SSH Client like Putty, and connect to the NetBurner device.

13.43.5 SecureSerToEthFactoryApp

This program is designed to connect a serial device to a network. When using RS-232 or RS-485, the application basically opens a serial port and an Ethernet port, then connects them. This allows any data sent to the Ethernet port to be forwarded to the serial port, and vice versa. When using I2C, the program creates a dedicated TCP server for Ethernet-to-I2C communication.

1. Only 1 connection is allowed at a time on each serial port. Additional client connections will be queued and serviced as the active connection completes.
2. Override time-out: When a new TCP connection is attempted while there is already an active connection, one of 3 actions can be taken:
   (a) Do not disconnect the active connection.
   (b) Only disconnect the existing connection if it has been idle for a number number of seconds, as specified by the "Allow new connection... X seconds" field.
   (c) Always disconnect the existing connection. Configure "Allow new connection..." to '0'.

13.44 SSL/TLS

SSL/TLS examples. Note that all references to SSL represent TLS.

- FTPS Server

- HTML File Post
- HTML Form Post
- HTTPS Dual Cert
- HTTPS Upload Cert
- HTTPS Web Server Demo
- HTTPS GET Request
- SSL/TLS Client
- SSL/TLS Client Certificate
- SSL/TLS Client Verify Peer - Basic
- SSL/TLS Client Verify Peer - EFFS-STD
- SSL/TLS Onboard Self-signed Certificate Generation
- SSL/TLS Receive Mail (POP3)
- SSL/TLS Send Mail
- SSL/TLS Send Mail w/ Attachment
- SSL/TLS Send Mail w/ EFFS Attchment
- SSL/TLS Server
- HTTPS Configuration Mirror

13.44.1 FTPS Server

This example will create a simple FTPS Server running on the NetBurner device. The server will start and wait for a FTPS client connection on port 990. The goal of this example is strictly the encryption aspect; it can be extended to use the file systemZby combining it with the Embedded Flash File System (EFFS) examples, EFFS-FAT or EFFS-STD.
The server provides two functions:

1. Allows the FTPS Client to download a single file named ReadFile.txt
2. Allows the FTPS Client to upload a text file named WriteFile.txt, which will be displayed on the serial port. It is not stored since this simple examle does not use the file system.

The server uses a built in self-signed certificate and key. Status messages, including sockets in use, are displayed on the serial port.
To run the example:

- Run MTTTY and connect to the debug serial port
- Run a program such as WinSCP that supports FTPS (not SFTP)
- Connect to the FTPS Server on the NetBurner device
- Select the ReadFile.txt file to download or view it
- Select the WriteFile.txt file in the FTPS Client and send it to the FTPS Server on the NetBurner device. It will be displayed on the serial port

13.44.2 HTML File Post

This example presents a web page with a field for selecting a file to upload to the NetBurner device. When the "Send File" button is selected, the upload is performed. The uploaded file is then displayed on the web page. A quick test is to send this example's ReadMe.txt file.
This example is similar to the `examples\web\HtmlFilePost`, with the addition of SSL/TLS capability.
Browsers are starting to require a valid certificate on the device, as well as the installation of the Certificate Authority (CA) certificate installed in the browser for self-signed certs. Receiving a warning message in the browser and selecting the option to continue anyway may work for variables posted in a form, but it will not work when posting a file.

13.44.3 HTML Form Post

- Similar to the HtmlFormPost in the `examples\web` folder, but works with SSL/TLS
- Uses the HTML POST command (the other is HTML GET) to send HTML form data from a web browser to the NetBurner device.
- Illustrates the use of the NetBurner POST callback objects, named `postForm1` and `postForm2` in this example, to intercept and process HTML Post data.
- Demonstrates how to handle multiple web pages with HTML forms and content delivery using the library functions SendFullResponse() and RedirectResponse().

13.44.4 HTTPS Dual Cert

HTTPS Server Example for multiple certificates
Supported Platforms: MODM7AE70, SBE70LC, MOD5441X, NANO54415, SBE70LC
This example creates a web page that can be viewed as normal or encrypted with SSL/TLS. An example certificate is included in DEVICE.CRT and DEVICE.KEY. Load these files on the flash card prior to the device booting up. Since this is an example cert, the web browser will display warning messages to this effect. For this example select the "continue to page" options to allow the HTTPS for a single session.
In addition this example illustrates how to have both a permanent compiled-in certificate and key, as well as one that can be loaded from an external Flash card using the EFFS FAT file system. If a certificate and key are present on the Flash card named DEVICE.CRT and DEVICE.KEY, they will be used instead of the local, compiled-in certificate and key.
If you are using NBEclipse, then you will also need to tell the linker to include the FatFile.a library. To do this, complete the following steps:

- In NBEclipse, right-click on your project, and select "Properties"
- Select "C/C++ Builds -> Settings" on the left-hand side
- Select "GNU C/C++ Linker -> Libraries" under the "Tool Settings" tab
- In the "Libraries" list box, add "FatFile" by using the action icons provided in top-right corner of the list box

Example Features:

- Compiled-in certificate and key
- Optional Flash card certificate and key
- NTP client to set the time
- FTP to enable the upload of a certificate and key to a Flash card
- Flash card test code to verify the card can be read/written to
- Web page and links to load a HTTPS or HTTP version

13.44.5 HTTPS Upload Cert

This program demonstrates how to upload a user certificate and key to support SSL/TLS web page access. The uploaded certificate and key are stored in the on-chip flash memory using the standard file system (EFFS-STD). The serial port provides status messages on the application operation.
The application has a built-in certificate and key, ServerCert.cpp and ServerKey.cpp, to provide security before before a user certificate and key are installed. You can generate a certificate and key by your own method, or using the makeca and makeserver scripts found in <nndk_install>\CreateCerts\<ECDSA or RSA>. The Certificate Authority is createed by makeca, which is then used to sign the server certificate and key with makeserver.
The display on the web interface will indicate whether you are using the compiled certificate and key (listed as "Default"), or the uploaded version (listed as "User Installed").
To run the example:

- Create a server certificate and key
- Run MTTTY to view status messages on the serial port
- Open the web page in a browser, which will be using the compiled in certificate and key if this is the first time
- Use the web interface to upload your own certificate and key

Note

Browsers are starting to require a valid certificate on the device, as well as the installation of the Certificate Authority (CA) certificate installed in the browser for self-signed certs. Receiving a warning message in the browser and selecting the option to continue anyway may work for variables posted in a form, but it will not work when posting a file.

13.44.6 HTTPS Web Server Demo

SSL/TLS example program that demonstrates getting an ACME cert automaticalls. will only work for a device on the big I internet with a DNS entry. Make sure to change the DNSNAME constant in main.cpp to match where you are.
SSL/TLS Server example program with web page redirection for unauthorized access. The example demonstrates how you can have both secure and non-secure access to files and directories.
The application starts the web server with SSL/TLS capability. The directory structure is such that index.html and the files in the images subdirectory can be viewed with a http or https connection. However, files in the httpsdir directory (ie repeat.html) can only be viewed with a secure https connection.

```
html
|-- index.html
|-- images
      |-- (various image files)
|-- httpsdir
      |-- repeat.html
```

The `HTTP_ACCESS CheckHttpAccess(int sock, int access_level, HTTP_Request &Req)` function is used to authenticate the web page access. It will allow access or redirect depending on the file and access rights.

13.44.7 HTTPS GET Request

This example demonstrates how a SSL/TLS Client can retrieve a web page from a HTTPS server.

- Run the MTTTY serial terminal
- When the application starts there will be a prompt to enter a HTTPS web site name (eg. www.google.com)
- The retrieved web site data will be displayed as text in MTTTY

13.44.8 SSL/TLS Client

SSL/TLS Client Example
This example demonstrates how SSL/TLS Client connections can be made. It will attempt to connect to the specified SSL Server and keep track of the number successful and failed attempts. Status messages will be sent to both the SSL/TLS Server, and to the serial debug port on the NetBurner device.
You must modify the `SSL_SERVER_NAME` to match your SSL Server device.
Testing: One method of testing a SSL/TLS client is to use `openssl s_client`. A typical command to start the server at the time of this writing is: `openssl s_server -accept 4433 -cert Server.crt -key Server.key`
Where you supply the .crt and .key files created by the NetBurner tools, OpenSSL, or through a Certificate Authority. For more information please refer to the openssl s_server documentation online.
Certificate checking is disabled by default. To enable certificate checking you must:

1. Include a CA certificate list in your project
2. Uncomment `#define NB_SSL_CLIENT_CERTIFICATE_CHECKING_ENABLED` in sslclient.cpp
3. Rebuild the system libraries

13.44.9 SSL/TLS Client Certificate

PROGRAM DESCRIPTION: SSL Client Certificate Example
This example demonstrates:

1. How SSL Client connections can be made.
2. How to load and use Client Certificates.
3. This example will only work if you are connecting to a SSL server that is configured to request and receive client certificates. In this example the client certificate is compiled as part of the application.

The application will attempt to connect to the specified SSL Server and keep track of the number successful and failed attempts. Status messages will be sent to both the SSL Server, and to the serial debug serial port on the NetBurner device.
IMPORTANT: A Client Certificate is a very different thing than when a client checks a server certificate against a list of Certificate Authorities (CA):
Certificate Checking against a CA: In this mode the Client will check the CA portion of the server certificate against a list of CA's maintained by the client. If the CA's match, the connection is allowed. This mode requires a modification to predef.h to enable client certificate checking. This is NOTthe purpose of this example application.
Sending a Client Certificate to the Server: In this mode, in addition to the server sending a certificate to the client as in the above case, the client is required to send a certificate to the server. That is the purpose of this example application.
CREATING A CERTIFICATE AND CA: Please refer to the NetBurner Security Library documentation for a detailed description. For the convenient of this example, we have created one for you.

13.44.10 SSL/TLS Client Verify Peer - Basic

This program will demonstrate how to use CA Lists for use in support of peer verification. One certificate is defined in caList.h. This is the root certificate for GitHub. They are passed into the call to SSL_connect(), and the verify peer parameter in that function is set to true.
To test this example, use the command 'C' at the debug prompt, and then type the name of the site that you wish to test a connection to.

13.44.11 SSL/TLS Client Verify Peer - EFFS-STD

This program will demonstrate how to upload CA Lists to use in support of verify peer. The uploaded information is stored in the onchip flash memory using the standard file system. The serial port provides a debug menu.
We include github.pem with this example to use for your testing. Load this file onto the device through the web interface and then try to connect to GitHub.com through the serial menu. You should see a message stating that there's a good SSL connection if it's successful.

13.44.12 SSL/TLS Onboard Self-signed Certificate Generation

SSL/TLS Examples for Onboard Certificate Generation

- On-board Cert Generation - Simple
- On-board Cert Generation - Advanced
- On-board Cert Generation - Compiled Certificate Authority

13.44.12.1 On-board Cert Generation - Simple

This example shows how to enable auto-generation of self-signed certificates. The certificate will enable secure communication to the system configuration web server and the HTTPS web server of the application. On-board generation of certificates are an alternative to creating your own self-signed certificates and uploading them to your NetBurner device. This simple example should cover most use cases. Please refer to the other examples for additional functionality.
Auto-generated certificates will operate as follows:

- If enabled and no certificate exists, a certificate will be created on first call to SSL_accept() or SslInitServer().
- The Common Name (CN) will be the IP address of the device. Alternate names are also supported.
- The application must ensure it has the correct system time, otherwise the certificate dates will be invalid.
- Calling the function `EnableOnboardCertificateCreation()` enables certificate generation.
- If `ENABLE_AUTOCERT_REGEN` is defined in predef.h, the certificate will update when it expires, creating a new 1 year certificate. Generation will occur on next SSL/TLS access.
- If `ENABLE_AUTO_CERT_REGEN` is defined in predef.h, AUTO_CERT_GEN_CHECK will dictate how frequently a certificate is checked for expiration. The default value is one minute.
- The certificate will update if the IP address changes to handle environments such as DHCP address assignments.

If the system time is incorrect, the certificate will be invalid.
Additional Examples:
Examples of a HTTPS server with a compiled-in certificate:

- `<NNDK install>\examples\ssl\sslserver`
- `<NNDK install>\examples\ssl\SslWebDemo`

Examples of uploading a certificate/key pair manually:

- `<NNDK install>\examples\ssl\HttpsDualCert`
- `<NNDK install>\examples\ssl\HttpsUploadCert`

13.44.12.2 On-board Cert Generation - Advanced

Please refer to the ReadMe.txt for the Simple version of this example for an operational description of the On-board certificate generation feature.
The simple auto-generate self-signed certificate example is what is normally used in applications. This example provide advanced options, such as interactive control to:

- Generate the certificate
- Delete the certificate
- View certificate expiration date
- Check for a valid certificate
- Manually enter the system time

13.44.12.3 On-board Cert Generation - Compiled Certificate Authority

Please refer to the ReadMe.txt for the Simple version of this example and for an operational description of the On-board certificate generation feature.
The simple auto-generate self-signed certificate example is what is normally used in applications. This example provides advanced users with a demonstration of how to write their own onboard certificate generation function. This includes signing the generated certificate with a compiled in Certificate Authority cert. To understand how to generate Certificate Authority certificate as well as how to compile it into your program, please see our documentation on creating self-signed certificates, found in the Programmers Guide.
For any network device, if you generate your own Certificate Authority (CA), you must accept responsibility for the security of the CA certificate. If a malicious party obtains your CA from the device or any other means, they would be able to use it to create their own "trusted" certificate and access your devices. This may be less of an issue if you are operating on a closed/private network, rather than devices on the Internet.
Taking full advantage of this example will also require some familiarity with the wolfSSL API.

13.44.13 SSL/TLS Receive Mail (POP3)

This example demonstrates how to implement retrieving email from a server that requires SSL/TLS.
When trying to connect to a Google Account, you will likely need to enable less secure application access. If you use two-factor authentication with your account, you will need to setup an application password for the account being connected to. This can be done at the following URL: https://myaccount.google.↩com/apppasswords
Note that when setting this up, it will tell you that you do not need to save the password. For this example, this is not the case. The application password is not stored anywhere, and will need to be reentered for every request.

13.44.14 SSL/TLS Send Mail

This program illustrates how to send an email message to a server that uses SSL/TLS.
All the magic happens in the `webfuncs.cpp` file, this code just initializes the app
Note that this application uses a secure SSL/TLS connection when accessing the web server. All connections made through it will need to take place via HTTPS.
There are two ways the SSL/TLS SMTP handshake can happen, with or without "STARTTLS". With servers that don't require STARTTLS, all that needs to be done differently than a regular SMTP handshake is to connect using "SSL_connect" to port 465 or 587 (instead of a regular connect to port 25), and from there proceed like a normal SMTP handshake. This method is used when we know for sure the server accepts SSL/TLS mail and we definitely want to use SLL (since some server like Yahoo allow you to connect in both ways).
If we are not sure if a server supports SSL/TLS, or, we only want to use SSL/TLS if it is required, we use the STARTTLS method. With this method, the send_mail function first opens a connection to the regular SMTP port 25 and does an "EHLO". if the server has the key word "STARTTLS" in its response, it means that it supports TLS, from there, the client says "STARTTLS" and then calls the SSL_negotiate function which establishes a secure connection with the server. After the secure connection is established, we say "EHLO" again and proceed as normal.
Gmail requires SSL/TLS, so if we try to connect to it on port 25 (the non SLL port) it will say STARTTLS without giving the option AUTH, which means SSL/TLS is required and we must start the SSL/TLS negotiation. Yahoo allows both, but but does not offer STARTTLS. This means that it doesn't let you know that SSL/TLS is available if connected to on the regular SMTP port, it expects you to connect to port 465 or 587 if you want to use SSL/TLS, and port 25 for non SSL/TLS.
When trying to connect to a Google Account, you will likely need to enable less secure application access. If you use two-factor authentication with your account, you will need to setup an application password for the account being connected to. This can be done at the following URL: https://myaccount.google.↩com/apppasswords
If you are using NBEclipse, then you will also need to tell the linker to include the `/nburn/platform/<platform>/original StdFFile.a library`. To do this, complete the following steps:

- In NBEclipse, right-click on your project, and select "Properties"
- Select "C/C++ Builds -> Settings" on the left-hand side
- Select "GNU C/C++ Linker -> Libraries" under the "Tool Settings" tab
- In the "Libraries" list box, add "StdFFile" by using the action icons provided in top-right corner of the list box

13.44.15 SSL/TLS Send Mail w/ Attachment

Send an email with a plain text attachment
Note that this application uses a secure SSL/TLS connection when accessing the web server. All connections made through it will need to take place via HTTPS.
This example illustrates how to send a MIME email attachment in plain text, without using the EFFS FAT32 file system (there is a different example for EFFS). Once the example is running, all interaction is done through the web page.
When trying to connect to a Google Account, you will likely need to enable less secure application access. If you use two-factor authentication with your account, you will need to setup an application password for the account being connected to. This can be done at the following URL: https://myaccount.google.↩com/apppasswords
The default STMP port value will be set to 465 for a connection using SSL/TLS. This example is not structured to handle a STARTTLS response. Please see the SslSendMail example for more information.

13.44.16 SSL/TLS Send Mail w/ EFFS Attchment

Supported Platforms: MOD5551X, NANO54415, MODM7AE70, SBE70LC
This example demonstrates how to attach files from the EFFS FAT32 file system to an email. Once the program starts, all interaction is done from the web page.
Note that this application uses a secure SSL connection when accessing the web server. All connections made through it will need to take place via HTTPS.
When trying to connect to a Google Account, you will likely need to enable less secure application access. If you use two-factor authentication with your account, you will need to setup an application password for the account being connected to. This can be done at the following URL: https://myaccount.google.↩com/apppasswords
The default STMP port value will be set to 465 for a connection using SSL/TLS. This example is not structured to handle a STARTTLS response. Please see the SslSendMail example for more information.

13.44.17 SSL/TLS Server

This program will create a SSL/TLS server task that uses Elliptic Curve Cryptographic (ECC) keys. To test the application you can use another NetBurner device configured as a client, or openssl. The command line options for openssl as of 13-Nov-2018 are: `openssl s_client -cipher ECDHE-ECDSA-AES256-GCM-SHA384 -connect <ip address>:<port>`
For example: `openssl s_client -cipher ECDHE-ECDSA-AES256-GCM-SHA384 -connect 10.1.1.191:8883`
If you need an RSA key for your specific application needs, you can generate them using the scripts found in `\nburn\CreateCerts\RSA`. Just replace ServerKey.cpp and ServerCert.cpp in this example with those that you generate from the scripts and rebuild the application. You can test this with the following OpenSSL commands.
`openssl s_client -connect <ip address>:<port>`
For example: `openssl s_client -connect 10.1.1.191:8883`
This example uses a simple `read()` function to receive data from a TCP client.

13.44.18 HTTPS Configuration Mirror

SSL/TLS Server example program with web page redirection for unauthorized access. The example demonstrates how you can have both secure and non-secure access to files and directories.
The application starts the web server with SSL/TLS capability. The directory structure is such that index.html and the files in the images subdirectory can be viewed with a http or https connection. However, files in the httpsdir directory (ie repeat.html) can only be viewed with a secure https connection.

```
html
|-- index.html
|-- images
      |-- (various image files)
|-- httpsdir
      |-- repeat.html
```

The `HTTP_ACCESS CheckHttpAccess(int sock, int access_level, HTTP_Request`

`&Req)` function is used to authenticate the web page access. It will allow access or redirect depending on the file and access rights.

13.45 Stack Protection

Demonstrate various stack overflow/underflow protection methods.
To run this example, the following lines must be uncommented in `\nburn\nbrtos\include\predef.h` to enable stack checking features:

```
#define NBRTOS_STACKCHECK     (1)
#define NBRTOS_STACKOVERFLOW  (1)
#define NBRTOS_STACKUNDERFLOW (1)
```

13.46 Syslog

The Syslog utility enables you to send logging information to a destination host computer using UDP (port number 514). This example uses syslog to send a simple counting variable to a host computer.
To run the example:

1. Verify you have proper network communication with you NetBurner device. It must have an IP address and mask.
2. Modify the source code line: `SysLogAddress = AsciiToIp("10.1.1.191");` so that it specifies the address of your computer. If this line is commented out and no SysLogAddress is specified, then the syslog data will be sent as a UDP broadcast.
3. Run the NetBurner "UDP Terminal Tool" application. Make sure the local listening port field is set to 514, the default syslog port.
4. Download and run the application on your NetBurner device.

13.47 System Diagnostics

By including the function `EnableSystemDiagnostics()` in your application, you can view system diagnostic information by going to the device's Configuration Server web page and clicking on the "Diagnostics" button.
This example demonstrates how to add your own additional variables to the diagnostic display, such as application integers, floats, strings, definitions, etc. To do so:

- Add `#include <diagnostics.h>`
- Declare the corresponding `DiagVarMon()` function. For example, to add an integer variable myInt you would declare: `static DiagVarMon MyIntMon("My Int", myInt);` The first parameter is the name or description you wish to use to identify the variable.

13.48 TCP

TCP Examples:

- TCP No Block Connect Test
- TCP Keepalive
- TCP Client
- Multiple TCP Interface Test
- TCP Multi-Socket Server
- TCP Resource Information
- TCP Server

- TCP Server Using Via
- TCP Speed Test
- TCP Stress Test

13.48.1 TCP No Block Connect Test

The standard TCP connect call will block until the connection is either made or fails. The non-blocking connect call will initiate a TCP connection and return immediately.

13.48.2 TCP Keepalive

This example program will create a TCP server that also implements keep alive functionality with the TCP client. Keep-alive is implemented with the `TcpGetLastRxTime()` and `TcpSendKeepAlive()` functions. `TcpGetLastRxTime()` returns the number of time ticks since the last time a packet was received. `TcpSendKeepAlive()` sends a keep-alive packet to the client, which is used when no data is has been transmitted within the timeout period.
The general concept is to call `TcpGetLastRxTime()`, call `TcpSendKeepAlive()`, wait a bit, then call `TcpSendKeepAlive()` a second time and verify the number of time ticks is different. If not, no packets were received and the client is not responding. Make sure to allow time for the client to respond to the keep alive packet. Do not call TcpGetLastRxTime more often than once every second to avoid performance issues.
This example creates a TCP server task that listens on port 23 by default. The server blocks with "ReadWithTimeOut" until data is received or the timer times out. If a timeout occurs the time of the last received packet is recorded and a keep-alive packet is sent. Then, the server goes back to waiting for a read again. If ReadWithTimeOut times out a second time, the value of the last received TCP packet is checked again. If the client is still active the value of lastRxTime will be different than the previous value (due to the keep-alive packet that was sent). If the client did not respond to the keep-alive packet the number will remain the same and the client is assumed to be non-responsive, and the connection is closed.
To test the application you can use a TCP client such as telnet or putty. For example, from a windows command prompt, type `telnet <ip address of NetBurner>`. Status messages are sent to the debug/console serial port.

13.48.3 TCP Client

This program demonstrates how to create a TCP Client. All interaction is through the web page interface, which enables you to type a message as well as the IP address and port number of the destination TCP server. You should be able to use any TCP server. If you do not have one, we have a windows example TCP server in the `\nburn\pctools\TCP\TcpServerWin` directory (as well as source code).

13.48.4 Multiple TCP Interface Test

TCP Multiple Interface Test

13.48.5 TCP Multi-Socket Server

This example creates a TCP server that listens on the specified TCP port number and can handle multiple TCP connections simultaneously (10 in this example). The select() function is a great way method to pend and process multiple connections.
An easy way to test the example is to use multiple Telnet sessions to create simultaneous connections to the NetBurner device. Status messages are sent out stdio to the debug serial port, and to the client TCP connections.

13.48.6 TCP Resource Information

This example demonstrates the use of functions to obtain the number of free system buffers and the number of free network sockets. A socket will be consumed when you listen or connect. For example, the web server will consume one socket listening to port 80 for incoming connections.

If the system is out of buffers or sockets, it will not be able to accept any incoming connections or make any outgoing connections.
`int32_t GetFreeCount()` returns the number of free system buffers `int32_t GetFreeScoket↩ Count()` returns the number of free sockets/file descriptors, not included extra fds `int32_t GetFreeExtraFDCount()` returns the number of extra file descriptors

13.48.7 TCP Server

This program will create a TCP server task, which listens on port 23 by default. To test the application you can use Telnet. For example, from a windows command prompt, type: `telnet <ip address>`
This example uses a simple read() function to receive data from a TCP client. Refer to the TCP Multi Socket Server example to handle multiple client connections.

13.48.8 TCP Server Using Via

This example is based on the TCP Multi-Socket Server example, with the addition of the listenvia() function call to specify a specific network interface for multi-interface devices.
This program will create a TCP server task, with multiple listening sockets, one listening on all interfaces, and an additional socket each listening on a specific network interface. By default, these sockets listen on port 23 and incrementing from there. To test the application you can use Telnet. For example, from the command line, type: `telnet <ip address> <port>`
This example uses the select() method to handle the multiple sockets within the context of a single task.

13.48.9 TCP Speed Test

This program will test the receive speed of the NetBurner device. To run the program:

- Run MTTTY and connect to the debug serial port
- Build and download this example application
- Run the TcpSpeedTest.exe PC application, found in the "pc" subdir in this example in a windows command prompt window
- The test results will be displayed in MTTTY and in the command prompt window

Executing TcpSpeedTest.exe from a PC:

- Need a NetBurner device running it's speed test code on the same LAN
- Open a command prompt, type "./TcpSpeedTest <ip address>" where "ip address" is the address of the NetBurner device.
- The "-r" command line option will repeat forever or until you use cntl-c to exit.

13.48.10 TCP Stress Test

Example to test various response times. Work in conjunction with a windows program to exercise the test cases:

- Listen for an incomeing connection and exchange data
- Listen for an incomeing request, then make an outgoing request and exchange data
- Worse case time for sending and receiving one byte at a time both directions
- Response time of the select() function to block on multiple fds
- Bulk data transfer time

Time for the various tests will be in multiples of a time tick, which is by default 20 ticks per second (TICKS_PER_↩ SECOND).
Be sure not to include the pc folder in your NetBurner project. It is a windows program.

13.49 Telnet Command

This example shows how to use the Command Processor Library, which will accept and process user commands.↩ Connections can be made to the command processor through the serial port, or through a telnet connection.TCP connections may also be used.
To use the Command Processor Library, a set of callback functions are defined to accept user connections, handle authentication, handle disconnections, provide a prompt, and of course to process the commands.

13.50 TFTP - Trivial File Transfer Protocol

This example program will use a NetBurner device to communicate with the NetBurner TFTP Server running on a PC. To run the example:

1. The TFTP Server on the PC must be started and configured to handle both reads and writes.
2. You must specify the TFTP server in the NetBurner device using IPSetp or through the monitor

13.51 Time Functions

Example of how to set and read system time.
Example of system time functions:

- Set system time from a NTP server or manual entry (see also: RTC example)
- Use of time functions such as time(), mktime(), localtime(), etc
- How to display date and time
- Time zones and use of tzsetchar() to set a time zone
- Use of time_t and tm structure

13.52 Timers

System timer examples:

- Timer
- Interval Timer
- Stopwatch Timer

13.52.1 Timer

The HiResDelay class uses a hardware timer to provide a microsecond delay function.

13.52.2 Interval Timer

The Interval Timer user a processor hardware timer to schedule periodic interrupts. This can be useful for applications that need to do something as a reliable interval.
The example demonstrates how to:

- Create an IntervalTimer object
- Trigger an interrupt and access an interrupt service routine
- Post a semaphore
- Post a flag

13.52.3 Stopwatch Timer

A StopWatch object is used to time events. Once the object has been created, the Start and Stop member functions are used to control timing events.

13.53 UDP

The UDP API includes both socket based and object based (UDP Packet) implementations.
UDP Examples:

- UDP to Serial
- UDP Notify Callback Function
- UDP C++ Packet Class
- UDP Echo
- UDP Send/Receive
- UDP Sockets
- UDP Sockets Via

13.53.1 UDP to Serial

This program will send/receive data between a UDP program, such as the NetBurner UDP Terminal, and a serial port (such as the MTTTY program). The UDP configuration parameters are handled through the web page configuration interface.

13.53.2 UDP Notify Callback Function

This example illustrates how to use a callback function to be notified of any incoming UDP packets. The key components are:

- Standard UDP packet class
- A callback function to be executed when data is available
- Using the RegisterUDPFifoWithNotify() instead of RegisterUDPFifo()

13.53.3 UDP C++ Packet Class

This application will send/receive UDP packets with another host on a network, such as a PC. Use the MTTTY serial port program to access the menu and prompts to specify the destination IP address and port number.
NetBurner supplies an API for handling UDP as a C++ Class using UDPPacket, or you can use a UDP sockets API (see UDP socket example).
For an external UDP host you can use the NetBurner Java example, or the NetBurner UDP terminal program.

13.53.4 UDP Echo

This example program will receive a UDP packet from another device or host computer, and then send a response. To run the example, connect a serial port from your PC to the debug serial port on your NetBurner device and run a terminal program such as MTTTY. On the PC, run the NetBurner UDP Terminal (be sure to set the IP address and port numbers to match). Send data from the NetBurner UDP Terminal, and observe that the packet is received in MTTTY. Then observe the response from the device in the NetBurner UDP Terminal.

13.53.5 UDP Send/Receive

This example program will receive a UDP packet from another device or host computer, and then send a response. UDP is handles using the UDP C++ Class.
Note there is also an API for handling UDP using a socket interface.
To run the example, connect a serial port from your PC to the debug serial port on your NetBurner device and run a terminal program such as MTTTY.
On the PC, run the NetBurner UDP Terminal (be sure to set the IP address and port numbers to match) or some other UDP program of your choice. You will then be able to type characters in the UDP Terminal and see them in MTTTY, and vice versa.
You will be prompted for the port number to send/receive data and the destination IP address of the other device or host. Note that the application uses the same port number to send and receive data, but you can use any other port number you wish.
The application will create a thread to receive packets and display them on the debug port, while the main task will take any data you type in to the MTTTY terminal and send it as a UDP packet to the destination IP address.

13.53.6 UDP Sockets

This application will send/receive UDP packets with another host on a network using the UDP Sockets API. Use the MTTTY serial port program to access the menu and prompts to specify the destination IP address and port number.
Note ther is also an API an API for handling UDP as a C++ Class using UDPPacket.
For an external UDP host you can use the NetBurner Java example, or the NetBurner UDP terminal program.

13.53.7 UDP Sockets Via

This application will send/receive UDP packets with another host on a network using the UDP Sockets API. This opens multiple sockets for all different interfaces. Use the MTTTY serial port program to access the menu and prompts to specify the destination IP address. It also demonstrates using Select for UDP receive.
There is also an API an API for handling UDP as a C++ Class using UDPPacket
For an external UDP host you can use the NetBurner Java example, or the NetBurner UDP terminal program.

13.54 VLAN

Creates 3 VLAN interfaces:

- Ethernet 0, DHCP, VLAN ID 1
- Ethernet 0, DHCP, VLAN ID 2
- Ethernet 0, Static IP, VLAN ID 10

The serial port interface provides an interactive menu to display interface and VLAN information, as well as other system parameters.
To build this example MULTIHOME must be enabled in predef.h. Rather than change the file in `\nburn\nbrtos\include`, use the override feature and add predef.h to your project.

13.55 Web Server

Web Server Examples:

- Ajax Graph
- Bootstrap Hello World
- HTML File Post
- Flash Form
- GIF Canvas
- HTML Application File Post

- HTML Cookie
- HTML Form Post
- HTML Form Post 2.X Compatible
- HTML Password
- HTML Server GET Request
- HTMLVariables
- Serial Webserver
- Signed Application
- Simple HTML
- TicTacToe

13.55.1 Ajax Graph

This example demonstrates the code to create a webpage with a dynamic graph, generated by data from the NetBurner device. Specifically, it has been designed to imitate the data formats found in log files, where ordering and frequency are varied between data sources.
The vast majority of the example is simply the support code needed to generate the random data. This code is located in the other support files. The locations that actually are relevant to the AJAX portion of the example are:
main.cpp: `getSensorData()` `DumpLogData()`

ticks.html: `<!--FUNCTIONCALL getHTMLTicks -->`

data.html: `<!--CPPCALL getSensorData() -->`

index.html: Basically, all the Javascript.

13.55.2 Bootstrap Hello World

The Bootstrap HelloWorld Example demonstrates how easy it is to utilize Twitter Bootstrap (`http://twitter.github.io/bootstrap/`) on a NetBurner device. This example loads a simple "Hello World" web page on to the device, along with the Twitter Bootstrap framework (version 4.0).

13.55.3 HTML File Post

This example presents a web page with a field for selecting a file to upload to the NetBurner device. When the "Send File" button is selected, the upload is performed. The uploaded file is then displayed on the web page.

13.55.4 Flash Form

The FlashForm example demonstrates three different capabilities:

1. Dynamic HTML
2. HTML Form processing
3. User Flash parameter storage. Note that as of the 3.x release, we recommend you use the configuration server/storage.

All the processing happens when a web browser posts a form, and all the comments are in the formcode.cpp source file.

13.55.5 GIF Canvas

This example program describes how to programmatically create images and text. This file, main.cpp, starts the application, network services and web server. The functions that create the images and interact with the web browser are located in drawimage.cpp.
To run this application compile and download the program file, then use your web browser to connect to the NetBurner device.

13.55.6 HTML Application File Post

This example presents a web page where the user can post a file to update the code in the device.
Optionally you can uncomment:

```
#define I_WANT_TO_AUTHENTICATE_UPDATE
```

to enforce a username password.

13.55.7 HTML Cookie

This example shows how to implement cookies with the NetBurner web server. The example program works as follows:

- Three web pages are created: an index page, a page to set the cookie value to "MyCookie", and a page to view the cookie value.
- When you run the program open a web browser and go to the index page.
- Select the "set cookie" link to set the cookie. Note that if you select "show cookie" before setting the value, no value will be displayed.
- Select the "show cookie" link to view the cookie

A cookie will apply to a "site". In this example a "site" will probably be the IP address of your NetBurner device. The cookie will be sent as part of the HTML header for any page requested from the "site".

13.55.8 HTML Form Post

- Uses the HTML POST command (the other is HTML GET) to send HTML form data from a web browser to the NetBurner device.
- Illustrates the use of the NetBurner POST callback objects, named `postForm1` and `postForm2` in this example, to intercept and process HTML Post data.
- Demonstrates how to handle multiple web pages with HTML forms and content delivery using the library functions SendFullResponse() and RedirectResponse().

13.55.9 HTML Form Post 2.X Compatible

The standard HTTP POST methodology in 3.x should be used whenever creating a new project. The 3.X methods enable cleaner code, less URL parsing, and better performance.
For customers migrating large and complex applications from legacy NNDK 2.X tools and platforms to NNDK 3.X, it is possible to create a HTTP POST callback function that in turn calls a MyDoPost() style 2.X function. This example assumes familiarity with the NNDK 3.X HTTP POST methods.

- A NNDK 3.X HTTP POST callback function is declared with a wildcard mask ("*") so that it is called for all HTTP POST operations.
- Strings are created for the URL containing the form action name and the posted form data
- The 2.X style MyDoPost() function is called with the URL and data strings
- The ExtractPostData() function is copied from the 2.X tools an included as a function in the application

Once the example has been loaded, interaction is through the device's web page. There are two web pages with forms for data entry, and a summary page at the end to display the assigned values from the post. Status messages are displayed on the serial port that can be viewed with a serial terminal, such as the MTTTY utility.
While every attempt has been made to ensure compatibility, every application must be tested to verify proper operation. Also, this example will not work with multi-part forms.

13.55.10 HTML Password

Demonstrates how to use user names, passwords and access groups. The application consists of an index page with links to three additional pages with access group numbers 0, 1 and 2.

- Group 0 = Open access
- Group 1 = Admin access
- Group 2 = User access

The function `HTTP_ACCESS CheckHttpAccess(int sock, int access_level, HTTP_Request & Req)` is used to authenticate the user names and passwords.
Note: A web browser will cache the username and password once entered. To force a password request again you will need to clear the browser cache, restart the browser, or open a new private browsing session.

13.55.11 HTML Server GET Request

Demonstrates how to use web server GET function callbacks to intercept browser GET requests. This mechanism can be used to enable the application to add custom processing and dynamic web content when a specific web page our URL is requested, or to add custom processing for all GET requests.

13.55.12 HTMLVariables

This example illustrates how to use dynamic HTML content with variables and function call parameters embedded in HTML code. The HTML tags VARIABLE and FUNCTIONCALL enable you to display application variables and call application functions directly from the HTML code.
In this example, the device's IP, Mask, Gateway, and DNS Server are displayed dynamically on the web page. The page also displays an uptime counter which updates when the page is refreshed.
Note for NBEclipse Users: This example requires that the auto-generated file htmldata.cpp have a path to include the htmlvar.h header file. To add the path in NBEclipse:
Right-click on your project and select properties Select "C/C++ Build" options Select "GNU C++ Compiler" -> Directories Use the "+" in the include path box to add the project's "HtmlVariables\html" folder to the list of include paths.

13.55.13 Serial Webserver

The Serial Webserver is a simple example demonstrating receiving and replying to HTTP requests over a serial port, using the NetBurner web server implementation. The example will display a "Hello World" type message on the device's index page: index.html.
An valid HTTP GET request that is compatible with this application is the following:

Note

The webserver expects a double carriage return line feed

```
'GET /INDEX.HTML HTTP/1.1\r\n\r\n'
```

The expected response should look similar to the following:

```
HTTP/1.0 200 OK
Pragma: no-cache
MIME-version: 1.0
Content-Type: text/html
Content-length: 231

<html>
<head><link rel="stylesheet" type="text/css" href="style.css"></head>
<body>
<a href="http://www.netburner.com"><img src="logo.jpg" width="200" height="104" border="0"></a>
<h1>Thank you for NetBurning!</h1>
</body>
</html>
```

13.55.14 Signed Application

Demonstrates the signed application update capability.
To use this you need to create a public/private key pair. It is VERY important that the private key is not stored on the device in order to be secure. The signing key needs to be different than all other certs and keys used on the device as a server, such as those for HTTPS.
From this public/private key pair, the public key needs to be stored in the application, and the private key needs to be stored in some other secure location. If the private key is lost, you will be unable to ever again update the device.
To create the public key and compile it into the application, from the command line:

- openssl genrsa -out signkey.pem 1024
- openssl rsa -in signkey.pem -pubout -out src/public.key
- set NB_SIGN_KEY = WHERE_EVER_THE_KEY_LIVES\signkey.pem

This example includes a batch file named makekey.bat that executes the above steps.
Once the public key has been created, the following commands can be used from the command line to create the signed application and load it into the device:

- make sign
- make loadsign

When using NBEclipse:

- Make the public/private key pair as described above.
- In the directory where you have the public key, run `compfile public.key codekey_array codekey_len codekey.cpp`.
- Now include the codekey.cpp in your NBEclipse project.

After NBEclipse has built your project:

1. Create an External Tool Configuration with Run->External Tools->External Tools Configurations.
2. In the new external tool configuration, specify the following: – Name: Sign – Location: "${env_var:NNDK_↩ ROOT}\pcbin\nbsign.exe" – Working Directory: Browse Workspace and select your project – Arguments: -k <full path to private key.pem> -in Release${project_name}.bin -o Release${project_name}.signed.bin
3. Project can now be signed by running the Sign External Tool

13.55.15 Simple HTML

The Simple HTML Example is a basic demonstration of the NetBurner web server implementation, and is a good starting point for custom applications. The example will display a "Hello World" type message on the device's index page: index.html.

13.55.16 TicTacToe

The TicTacToe application provides a number of simple examples:

- Play tic tac toe, of course
- Manipulate the LEDs on the NetBurner development board. This is an example of URL Encoding.
- Echo web browser requests, so you can see what is sent to the web server.
- Example of a meta tag to create a "reloading web page", which refreshes itself at 1 second intervals.

13.56 Web Client

Web Socket Examples:

- Earthquake
- Find My IP
- Find My IP Task
- Message Passer
- MessagePasserTask JSON

13.56.1 Earthquake

Simple program that demonstrates retrieving a JSON object from a site and using it to present the user with dynamic output; in this case obtaining the number of earthquakes that have occurred today and their details.

13.56.2 Find My IP

This program starts a separate, higher-priority task that navigates to a website and retrieves information; in this case we navigate to a site and retrieve the IP address that we connected with.

13.56.3 Find My IP Task

This program starts a separate, higher-priority task that navigates to a website and retrieves information; in this case we navigate to a site and retrieve the IP address that we connected with.

13.56.4 Message Passer

Message Passer example that checks the current time.

13.56.5 MessagePasserTask JSON

Message Passer Task to send a JSON object.

13.57 Web Sockets

Web Socket Examples:

- Web Socket Connect
- Web Socket Console
- Web Socket DIP Switch
- Web Socket Echo

13.57.1 Web Socket Connect

This is an example that demonstrates websocket connections.

13.57.2 Web Socket Console

Web Sockets Console example

13.57.3 Web Socket DIP Switch

Supported Platforms: MODM7AE70, MOD5441x, NANO54415
This program illustrates how to use the NetBurner WebSocket class to display the state of the DIP switches on the MOD-DEV-70CR in real-time on a webpage. That means as soon as you flip the DIP switches on the development board, the state will change on the webpage without having to refresh the page.
The application continuously polls the DIP switches and sends the state of the DIP switches via a websocket to the client. It's not the most efficient implementation when considering CPU utilization but this app demonstrates the capabilities of a WebSocket to allow the server(NetBurner device) to send the state of a variable to the client(webpage) with low latency and minimal packet size. This app also illustrates how to use the NetBurner JSON library to build and send JSON objects from the NetBurner device to the client. In this case, JSON objects are used to pass the state of the DIP switches to the webpage.

13.57.4 Web Socket Echo

Web Sockets Echo example

13.58 Wifi

Wifi Examples:

- Wifi Config AP
- Configure AP with JSON
- Wifi Firmware Update
- Wifi Access Point
- Wifi Client
- Wifi Scan

13.58.1 Wifi Config AP

Configure the wifi as an access point.

13.58.2 Configure AP with JSON

Configure the WiFi interface as an access point to provide the user a means to connect to local networks.
If no network SSID is provided to connect to upon boot, the application will start a WiFi Access Point. The Net↩ Burner's WiFi Access Point can be used to scan for local networks. The NetBurner device hosts a webpage that allows the user to establish a connection to local networks after performing a scan.
The default WiFi access point SSID will be of the form "NB_<last 6 bytes of MAC>". For example, a NetBurner access point may look like "NB_012345". If you would like modify the prefix of the access point to include custom product branding, the prefix can be re-defined by un-commenting and modifying the following line located at the top of main.cpp:
extern const NB::SSID_t default_SSID_Prefix = { 4, "FOO_" };
For the user's convenience, the device's SSID and WiFi Interface IP address are printed out the serial port. These can be used to access the device's webpage, which allows the user to connect to a local network.
This list of networks is generated using JavaScript. JavaScript is used to parse a JSON object sent by the NetBurner, which contains the access point scan results.

13.58.3 Wifi Firmware Update

Update the firmware in the wifi module.

13.58.4 Wifi Access Point

This example shows how to run the wifi module in Access Point mode. It also launches a DHCP Server running on the wifi interface, making it easier for a client to configure when connecting to the access point.
To setup your access point properly, you will need to configure the IP address and Mask of the Wifi interface. This can be done using IPSetup. You will also need to set the start address for the DHCP server to begin assigning IP addresses to connected devices. The variable 'startAddr' is used to so in this application.
For example, a proper AP configuration would be a WiFi interface with an IP address of 192.168.0.1, Mask of 255.255.255.0, and a DHCP server start address of 192.168.0.2

13.58.5 Wifi Client

This example demonstrates how to connect to a wifi network and use the wifi driver object to access the status information of the wifi connection.
If you are using a 5270 or 5234 processor based design it also enables spread spectrum mode to reduce emissions that can interfere with the wifi radio. These platforms should be using a wifi module with an external antenna.

13.58.6 Wifi Scan

Scan for access points

Chapter 14

Todo List

Member JsonRef::MakeInvalid ()

Document

Member MACADR::operator+ (uint32_t rhs)

Document

Chapter 15

Deprecated List

Member OSCritEnter (OS_CRIT ∗pCrit, uint16_t timeout)

This function is now deprecated. Please see OS_CRIT for current usage.

Member OSCritEnterNoWait (OS_CRIT ∗pCrit)

This function is now deprecated. Please see OS_CRIT for current usage.

Member OSCritInit (OS_CRIT ∗pCrit)

This function is now deprecated. Please see OS_CRIT for current usage.

Member OSCritLeave (OS_CRIT ∗pCrit)

This function is now deprecated. Please see OS_CRIT for current usage.

Member OSFifoInit (OS_FIFO ∗pFifo)

This function is now deprecated. Please see OS_FIFO for current usage.

Member OSFifoPend (OS_FIFO ∗pFifo, uint16_t timeout)

This function is now deprecated. Please see OS_FIFO for current usage.

Member OSFifoPost (OS_FIFO ∗pFifo, OS_FIFO_EL ∗pToPost)

This function is now deprecated. Please see OS_FIFO for current usage.

Member OSFifoPostFirst (OS_FIFO ∗pFifo, OS_FIFO_EL ∗pToPost)

This function is now deprecated. Please see OS_FIFO for current usage.

Member OSFlagClear (OS_FLAGS ∗flags, uint32_t bits_to_clr)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSFlagPendAll (OS_FLAGS ∗flags, uint32_t bit_mask, uint16_t timeout)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSFlagPendAllNoWait (OS_FLAGS ∗flags, uint32_t bit_mask)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSFlagPendAny (OS_FLAGS ∗flags, uint32_t bit_mask, uint16_t timeout)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSFlagPendAnyNoWait (OS_FLAGS ∗flags, uint32_t bit_mask)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSFlagSet (OS_FLAGS ∗flags, uint32_t bits_to_set)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSFlagState (OS_FLAGS ∗flags)

This function is now deprecated. Please see OS_FLAGS for current usage.

Member OSMboxInit (OS_MBOX ∗pmbox, void ∗msg)

This function is now deprecated. Please see OS_MBOX for current usage.

Member OSMboxPend (OS_MBOX ∗pmbox, uint16_t timeout, uint8_t ∗err)

This function is now deprecated. Please see OS_MBOX for current usage.

Member OSMboxPendNoWait (OS_MBOX *pmbox, uint8_t *err)

This function is now deprecated. Please see OS_MBOX for current usage.

Member OSMboxPost (OS_MBOX *pmbox, void *msg)

This function is now deprecated. Please see OS_MBOX for current usage.

Member OSQInit (OS_Q *pq, void **start, uint8_t size)

This function is now deprecated. Please see OS_Q for current usage.

Member OSQPend (OS_Q *pq, uint16_t timeout, uint8_t *err)

This function is now deprecated. Please see OS_Q for current usage.

Member OSQPendNoWait (OS_Q *pq, uint8_t *err)

This function is now deprecated. Please see OS_Q for current usage.

Member OSQPost (OS_Q *pq, void *msg)

This function is now deprecated. Please see OS_Q for current usage.

Member OSQPostFirst (OS_Q *pq, void *msg)

This function is now deprecated. Please see OS_Q for current usage.

Member OSQPostUnique (OS_Q *pq, void *msg)

This function is now deprecated. Please see OS_Q for current usage.

Member OSQPostUniqueFirst (OS_Q *pq, void *msg)

This function is now deprecated. Please see OS_Q for current usage.

Member OSSemInit (OS_SEM *psem, long value)

This function is now deprecated. Please see OS_SEM for current usage.

Member OSSemPend (OS_SEM *psem, uint16_t timeout)

This function is now deprecated. Please see OS_SEM for current usage.

Member OSSemPendNoWait (OS_SEM *psem)

This function is now deprecated. Please see OS_SEM for current usage.

Member OSSemPost (OS_SEM *psem)

This function is now deprecated. Please see OS_SEM for current usage.

Chapter 16

Module Index

16.1 NetBurner Library API

Here is a list of all modules:

Chapter 17

Namespace Index

17.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 18

Hierarchical Index

18.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 19

Class Index

19.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 20

File Index

20.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 21

Module Documentation

21.1 ARP - Address Resolution Protocol

Functions

- void ShowArp ()

  Display ARP cache, output will be the stdio serial port.
- void fShowArp (FILE *fp)

  Display ARP cache, output will be sent to the specified file pointer.
- BOOL GetArpMacFromIp (IPADDR4 ip, MACADR &ma)

  Check to see if the specified IP address is in the ARP cache.
- void sendGratuitousArp (int interfaceNumber, IPADDR4 ip)

  Send Gratuitous ARP Request.

21.1.1 Detailed Description

The Address Resolution Protocol (ARP) is a protocol for mapping an Internet Protocol address (IP Address) to a physical machine address called the Media Access Control (MAC), that is recognized on the local network. An local ARP cache correlating machine IP address to MAC address is maintained on each local machine.
When a host such as a PC computer attempts to communicate with another device, ARP is used to resolve the IP address to the corresponding MAC address.

21.1.2 Function Documentation

21.1.2.1 fShowArp()

```
void fShowArp (
            FILE * fp )
```

Display ARP cache, output will be sent to the specified file pointer.

Parameters

*fp	pointer to type FILE.

See also

ShowArp()

21.1.2.2 GetArpMacFromIp()

```
BOOL GetArpMacFromIp (
            IPADDR4 ip,
            MACADR & ma )
```

Check to see if the specified IP address is in the ARP cache.
This function does not send and arp request, it only checks the arp cache. If you want it to send an arp you could do something like send a ping to the IP address before calling this function.

Parameters

ip	IP address to check for
&ma	MACADR structuire to hold the result (C++ pass by reference)

Return values

TRUE	if found, otherwise FALSE.

21.1.2.3 sendGratuitousArp()

```
void sendGratuitousArp (
            int interfaceNumber,
            IPADDR4 ip )
```

Send Gratuitous ARP Request.
Used after adding an interface or changing an IP address to update the ARP caches of other hosts on the network. Ensure the interface has a valid IP address before sending.

Parameters

interfaceNumber	Specified which network interface to send on.
ip	The interface's IP address

21.1.2.4 ShowArp()

```
void ShowArp ( )
```

Display ARP cache, output will be the stdio serial port.

See also

fShowArp()

21.2 Authorization Responses

Enumerations

- enum AuthResponse : int16_t {
  eAuthSuccess = 1 , eAuthErrorUserExists = -1 , eAuthErrorUserDoesNotExist = -2 , eAuthErrorNoEmptyUserAuthRecords = -3 ,

eAuthErrorUnableToCreateHash = -4 , eAuthErrorAuthCheckFailed = -5 , eAuthErrorAuthTypeMismatch = -6 , eAuthErrorFailedRecordUpdate = -7 ,
eAuthErrorUnableToAddUser = -8 , eAuthErrorSaveFailed = -9 }

Response return codes when checking for the authorization status of a user.

21.2.1 Detailed Description

21.2.2 Enumeration Type Documentation

21.2.2.1 AuthResponse

`enum AuthResponse : int16_t`

Response return codes when checking for the authorization status of a user.

Enumerator

eAuthSuccess	The authorization request or function was successful.
eAuthErrorUserExists	The user already exists.
eAuthErrorUserDoesNotExist	The user does not exist.
eAuthErrorNoEmptyUserAuthRecords	The authorization records are full.
eAuthErrorUnableToCreateHash	There was an error hashing the authorization value.
eAuthErrorAuthCheckFailed	The authorization check failed.
eAuthErrorAuthTypeMismatch	The authorization type didn't match the request.
eAuthErrorFailedRecordUpdate	The record update failed.
eAuthErrorUnableToAddUser	The user was not added successfully.
eAuthErrorSaveFailed	Unable to save user authorization records.

21.3 Authorization Types

Enumerations

- enum AuthType : int8_t { eAuthTypeDefault = 0 , eAuthTypePassword = 1 , eAuthTypeKey = 2 }

  The types of authorization requests that are managed. These just indicate what the has value is, and don't provide any specific processing logic.

21.3.1 Detailed Description

21.3.2 Enumeration Type Documentation

21.3.2.1 AuthType

`enum AuthType : int8_t`

The types of authorization requests that are managed. These just indicate what the has value is, and don't provide any specific processing logic.

Enumerator

eAuthTypeDefault	Default value, should be considered an invalid type.
eAuthTypePassword	Password.
eAuthTypeKey	Key.

21.4 BSS Options

Option list BSS type values.

Macros

- #define **BSSTYPE_VALUE_INFR** (0x00)

  Infrastructure.
- #define **BSSTYPE_VALUE_ADHOC** (0x01)

  ADHOC.
- #define **BSSTYPE_VALUE_ANY** (0x02)

  Any.
- #define **BSSTYPE_VALUE_UNKNOWN** (0xFF)

  Unknown.

21.4.1 Detailed Description

Option list BSS type values.

21.5 Buffers - System Buffer Pool

Functions

- uint16_t GetFreeCount ()

  Returns the number of free buffers in the system. Buffers are used for both serial and network interfaces.
- void ShowBuffer (PoolPtr p)

  Prints a pool buffer to stdout.

21.5.1 Detailed Description

Functions associated with the buffers allocated for use by network interfaces and serial ports. If a system runs out of buffers it will no longer be able to send or receive network data.

21.5.2 Function Documentation

21.5.2.1 GetFreeCount()

```
uint16_t GetFreeCount ( )
```

Returns the number of free buffers in the system. Buffers are used for both serial and network interfaces.
This function returns the number of free pool buffers, which are used for network and serial communication. The maximum number of buffers is defined in \nburn\nbrtos\include\constants.h:

```
#define BUFFER_POOL_SIZE (128)
```

The size of each buffer is defined as:

```
#define ETHER_BUFFER_SIZE 1548
```

If the number of free buffers reaches zero, then no further network communication will be possible until some buffers are freed. This is a good function to use as a debug tool for detecting buffer leaks in your application.
Serial ports also use buffers, which are allocated in constants.h:

```
#define SERIAL_TX_BUFFERS (2)
#define SERIAL_RX_BUFFERS (2)
```

Each serial port buffer is equal in size to an Ethernet buffer.

Returns

The number of free buffers in the system.

21.5.2.2 ShowBuffer()

```
void ShowBuffer (
            PoolPtr p )
```

Prints a pool buffer to stdout.

21.6 CAN

Modules

- ColdFire MCF5441x Platforms
- MODM7AE70

21.6.1 Detailed Description

Controller Area Network (CAN) API for NetBurner Platforms

21.7 Cipher Options

Macros

- #define **CIPH_VALUE_NONE** (0x00)

  Option list cipher value: No cipher.
- #define **CIPH_VALUE_TKIP** (0x01)

  Option list cipher value: TKIP.
- #define **CIPH_VALUE_AES** (0x02)

  Option list cipher value: AES.
- #define **CIPH_VALUE_MIXED** (0x03)

  Option list cipher value: Mixed TKIP/AES.
- #define **CIPH_VALUE_UNKNOWN** (0xFF)

  Option list cipher value: Unkown cipher.

21.7.1 Detailed Description

21.8 ColdFire MCF5441x Platforms

Classes

- class canMCF5441x::CanRxMessage

  Class to hold received CAN messages.

21.8.1 Detailed Description

Control Area Network (CAN) for ColdFire MCF5441x platforms including MOD54415, MOD54417, NANO54415 and SB800EX.

21.9 Command Processor

Modules

- Command Processor Disconnect Causes
- Command Processor Listen Channels
- Command Processor Response Codes

Functions

- int CmdStartCommandProcessor (int priority)

  Start the command processor.
- int CmdAddCommandFd (int fd, bool require_auth, bool time_out_conn, bool local_echo=true)

  Add an established FD connection to the list of fd's managed by the command processor.
- int CmdRemoveCommandFd (int fd)

  Remove an established FD (either a TCP session, a serial connection, or an SSH session).
- int CmdListenOnTcpPort (uint16_t port, int do_telnet_processing, int max_connections)

  Start listening for a connection over TCP.
- int CmdListenQuietOnTcpPort (uint16_t port, int do_telnet_processing, int max_connections)

  Start listening for a connection over TCP, but without the siggnon or password.
- int CmdListenOnSshPort (uint16_t port, int max_connections)

  Start listening for a connection over SSH.
- int CmdListenQuietOnSshPort (uint16_t port, int max_connections)

  Start listening for a connection over SSH, but without the siggnon or password.
- int CmdStopListeningOnTcpPort (uint16_t port)

  Stop Listening for connections on the specified port. Also closes all open connections that were based on that port.
- int CmdStopListeningOnSshPort (uint16_t port)

  Stop Listening for connections on the specified port. Also closes all open connections that were based on that port.
- void SendToAll (const char *buffer, int len, bool include_serial_ports)

  Send a message to all connected sockets, excluding "Listening sockets".

Variables

- int(* CmdAuthenticateFunc)(const char *name, const char *passwd)

  External Authentication function CALLBACK for TCP connections, used to verify username and password. If this function pointer is not NULL then each new Telnet session will be asked to authenticate.
- int(* CmdAuthenticateSshFunc)(const char *name, const char *authVal, AuthType authType)

  External Authentication function CALLBACK for SSH connections, used to verify username and password. If this function pointer is not NULL then each new SSH session will be asked to authenticate. If this function pointer is NULL, it tries to use CmdAuthenticateFunc instead.
- int(* CmdCmd_func)(const char *command, FILE *fRespondto, void *pData)

  The command processing callback function for handling string commands.
- int(* CmdChar_func)(char command, FILE *fRespondto, void *pData)

  The command processing callback function for handling single character commands. If this is implemented does not do echo or line editing this is the responsibility of the application programmer.
- void *(* CmdConnect_func)(FILE *fRespondto)

  Connect callback function. If this function is not NULL, then the system will call this function every time a new session is started.
- void(* CmdPrompt_func)(FILE *fRespondto, void *pData)

  Prompt callback function. If this function is not NULL, then the system will call this function every time a new prompt line needs to be displayed.
- void(* CmdDisConnect_func)(FILE *fRespondto, int cause, void *pData)

  Dis-Connect callback function, if this function is not NULL then the system will call this function every time a session is terminated.
- int CmdIdleTimeout
- const char * **Cmdlogin_prompt**

  If this is defined, then it will be sent to the socket on connection before Authentication is tried.

21.9.1 Detailed Description

This is is NetBurner's Command Processor Library. It operates as a task and reads input one line at a time from multiple sources. When it has gathered a whole line, it sends it to a command processor that is defined by the user.

21.9.2 Function Documentation

21.9.2.1 CmdAddCommandFd()

```
int CmdAddCommandFd (
            int fd,
            bool require_auth,
            bool time_out_conn,
            bool local_echo = true )
```

Add an established FD connection to the list of fd's managed by the command processor.

Parameters

fd	The file descriptor.
require_auth	Do we Authenticate the connection on this FD?
time_out_conn	Do we time out the connection on this FD?
local_echo	Do we provide a local echo for this FD (default is true)?

Return values

CMD_OK	Command processor task started properly.
CMD_FAIL	Command processor task failed to start.
CMD_TO_MANY_FDS	Too many file descriptors are currently in use.

21.9.2.2 CmdListenOnSshPort()

```
int CmdListenOnSshPort (
            uint16_t port,
            int max_connections )
```

Start listening for a connection over SSH.

Parameters

port	The port number to listen on.
max_connections	What are the max number of connections we should allow on this port?

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_TO_MANY_FDS	Too many file descriptors are currently in use.
CMD_FAILED_SSH_KEY_GEN	Unable to generate the key needed for SSH communication.

21.9.2.3 CmdListenOnTcpPort()

```
int CmdListenOnTcpPort (
            uint16_t port,
            int do_telnet_processing,
            int max_connections )
```

Start listening for a connection over TCP.

Parameters

port	The port number to listen on.
do_telnet_processing	Should we treat the port as telnet and process telnet negotiations?
max_connections	What are the max number of connections we should allow on this port?

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_TO_MANY_FDS	Too many file descriptors are currently in use.

21.9.2.4 CmdListenQuietOnSshPort()

```
int CmdListenQuietOnSshPort (
            uint16_t port,
            int max_connections )
```

Start listening for a connection over SSH, but without the siggnon or password.

Parameters

port	The port number to listen on.
max_connections	What are the max number of connections we should allow on this port?

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_TO_MANY_FDS	Too many file descriptors are currently in use.
CMD_FAILED_SSH_KEY_GEN	Unable to generate the key needed for SSH communication.

21.9.2.5 CmdListenQuietOnTcpPort()

```
int CmdListenQuietOnTcpPort (
            uint16_t port,
            int do_telnet_processing,
            int max_connections )
```

Start listening for a connection over TCP, but without the siggnon or password.

Parameters

port	The port number to listen on.
do_telnet_processing	Should we treat the port as telnet and process telnet negotiations?
max_connections	What are the max number of connections we should allow on this port?

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_TO_MANY_FDS	Too many file descriptors are currently in use.

21.9.2.6 CmdRemoveCommandFd()

```
int CmdRemoveCommandFd (
            int fd )
```

Remove an established FD (either a TCP session, a serial connection, or an SSH session).

Parameters

fd	The file descriptor to remove.

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_FAIL	Failed to remove the file descriptor.

21.9.2.7 CmdStartCommandProcessor()

```
int CmdStartCommandProcessor (
            int priority )
```

Start the command processor.

Parameters

priority	The NBRTOS task priority to run at.

Return values

CMD_OK	Command processor task started properly.
CMD_FAIL	Command processor task failed to start.

21.9.2.8 CmdStopListeningOnSshPort()

```
int CmdStopListeningOnSshPort (
            uint16_t port )
```

Stop Listening for connections on the specified port. Also closes all open connections that were based on that port.

Parameters

port	The port number to listen on.

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_FAIL	Port requested wasn't listening for a connection.

21.9.2.9 CmdStopListeningOnTcpPort()

```
int CmdStopListeningOnTcpPort (
            uint16_t port )
```

Stop Listening for connections on the specified port. Also closes all open connections that were based on that port.

Parameters

port	The port number to listen on.

Return values

CMD_OK	Successfully removed the file descriptor.
CMD_FAIL	Port requested wasn't listening for a connection.

21.9.2.10 SendToAll()

```
void SendToAll (
            const char * buffer,
            int len,
            bool include_serial_ports )
```

Send a message to all connected sockets, excluding "Listening sockets".

Parameters

buffer	A buffer holding the message to send.
len	The length of the buffer to send.
include_serial_ports	Also send to serial ports?

21.9.3 Variable Documentation

21.9.3.1 CmdAuthenticateFunc

```
int(* CmdAuthenticateFunc) (const char *name, const char *passwd) (
            const char * name,
            const char * passwd )  [extern]
```

External Authentication function CALLBACK for TCP connections, used to verify username and password. If this function pointer is not NULL then each new Telnet session will be asked to authenticate.

Parameters

name	The user name trying to authenticate.
passwd	The password provided for the user.

Return values

CMD_OK	If the authentication was successful.
CMD_CLOSE	If the authentication was not successful.

21.9.3.2 CmdAuthenticateSshFunc

```
int(* CmdAuthenticateSshFunc) (const char *name, const char *authVal, AuthType authType) (
            const char * name,
            const char * authVal,
            AuthType authType )  [extern]
```

External Authentication function CALLBACK for SSH connections, used to verify username and password. If this function pointer is not NULL then each new SSH session will be asked to authenticate. If this function pointer is NULL, it tries to use CmdAuthenticateFunc instead.

Parameters

name	The user name trying to authenticate.
authVal	The authorization value provided for the user.
authType	The type of authorization value passed in.

Return values

CMD_OK	If the authentication was successful.
CMD_CLOSE	If the authentication was not successful.

21.9.3.3 CmdChar_func

```
int(* CmdChar_func) (char command, FILE *fRespondto, void *pData) (
            char command,
            FILE * fRespondto,
            void * pData )  [extern]
```

The command processing callback function for handling single character commands. If this is implemented does not do echo or line editing this is the responsibility of the application programmer.

Parameters

command	The single command character.
fRespondto	The FILE * all responses should be set to (use fprintf).
pData	Any data that is associated from the session. This is returned by CmdConnect_func.

Return values

CMD_OK	If the command was okay.
CMD_CLOSE	If the command causes the session to terminate (Close).

See also

CmdCmd_func

CmdConnect_func

21.9.3.4 CmdCmd_func

```
int(* CmdCmd_func) (const char *command, FILE *fRespondto, void *pData) (
            const char * command,
```

```
            FILE * fRespondto,
            void * pData )  [extern]
```

The command processing callback function for handling string commands.

Parameters

command	A pointer to the null terminated ASCII text of the command.
fRespondto	The FILE * all responses should be set to (use fprintf).
pData	Any data that is associated from the session and is returned by CmdConnect_func.

Return values

CMD_OK	If the command was okay.
CMD_CLOSE	If the command causes the session to terminate (Close).

See also

CmdChar_func

CmdConnect_func

21.9.3.5 CmdConnect_func

```
void *(* CmdConnect_func) (FILE *fRespondto) (
            FILE * fRespondto )  [extern]
```

Connect callback function. If this function is not NULL, then the system will call this function every time a new session is started.

Parameters

fRespondto	The FILE * all responses should be set to (use fprintf).

Return values

Returns	an arbitrary void* data item to be associated with the session. This data item is the same pData item used in other callback functions such as CmdPrompt_func.

See also

CmdCmd_func

CmdChar_func

21.9.3.6 CmdDisConnect_func

```
void(* CmdDisConnect_func) (FILE *fRespondto, int cause, void *pData) (
            FILE * fRespondto,
            int cause,
            void * pData )  [extern]
```

Dis-Connect callback function, if this function is not NULL then the system will call this function every time a session is terminated.

Parameters

fRespondto	The FILE * all responses should be set to (use fprintf).
cause	Why it disconnected. Can be any value from Command Processor Disconnect Causes.
pData	Any data that is associated from the session. This is returned by CmdConnect_func.

See also

CmdConnect_func

21.9.3.7 CmdIdleTimeout

```
int CmdIdleTimeout  [extern]
```

User defined value that specifies the number of seconds a connection is idle before it is terminated due to inactivity.

21.9.3.8 CmdPrompt_func

```
void(* CmdPrompt_func) (FILE *fRespondto, void *pData) (
            FILE * fRespondto,
            void * pData )  [extern]
```

Prompt callback function. If this function is not NULL, then the system will call this function every time a new prompt line needs to be displayed.

Parameters

fRespondto	The FILE * all responses should be set to (use fprintf).
pData	Any data that is associated from the session. This is returned by CmdConnect_func.

See also

CmdConnect_func

21.10 Command Processor Disconnect Causes

Macros

- #define **CMD_DIS_CAUSE_TIMEOUT** (1)

  Connection timed out.
- #define **CMD_DIS_CAUSE_CLOSED** (2)

  Connection closed.
- #define **CMD_DIS_SOCKET_CLOSED** (3)

  Socket closed. Don't send a response for this case.
- #define **CMD_DIS_AUTH_FAILED** (4)

  Authorization failed. Don't send a response for this case.

21.10.1 Detailed Description

Possible causes given for disconnects.

21.11 Command Processor Listen Channels

Enumerations

- enum ListenOn : int8_t { eListenOnTcp = 1 , eListenOnSsh = 2 }

The various remote channels that the command processor can listen to.

21.11.1 Detailed Description

21.11.2 Enumeration Type Documentation

21.11.2.1 ListenOn

```
enum ListenOn :  int8_t
```

The various remote channels that the command processor can listen to.

Enumerator

eListenOnTcp	Listen on TCP.
eListenOnSsh	Listen on SSH.

21.12 Command Processor Response Codes

Macros

- #define **CMD_OK** (0)

  Function was excecuted properly.
- #define **CMD_CLOSE** (1)

  Connection closed.
- #define **CMD_FAIL** (2)

  General function failed.
- #define **CMD_AUTH_FAIL** (3)

  Authorization failed.
- #define **CMD_TO_MANY_FDS** (4)

  Too many file descriptors currently in use.
- #define **CMD_FAILED_SSH_KEY_GEN** (5)

  Unable to correctly generate key needed for SSH connections.

21.12.1 Detailed Description

Valid return codes that can be returned by the NetBurner Command Processor functions.

21.13 Configuration Errors

Macros

- #define **CONFIG_ERR_SUCCESS** 0

  Success.
- #define **CONFIG_ERR_MSG_LENGTH** -1

  Message length.
- #define **CONFIG_ERR_SSID_LEN_SHORT** -2

  SSID length too short.
- #define **CONFIG_ERR_SSID_LEN_LONG** -3

  SSID length too long.
- #define **CONFIG_ERR_PASSWD_LEN_LONG** -4

  Password length too long.

- #define **CONFIG_ERR_INVALID_TABLE** -5

  Invalid table.
- #define **CONFIG_ERR_INVALID_OPTION** -6

  Invalid option.
- #define **CONFIG_ERR_BSSID_OVERRUN** -7

  BSSID overrun.
- #define **CONFIG_ERR_BSSID_UNDERRUN** -8

  BSSID underrun.
- #define **CONFIG_ERR_MULTI_CHANNEL** -9

  Multi-channel.
- #define **CONFIG_ERR_CONNECTED** -10

  Already connected.
- #define **CONFIG_ERR_UNKNOWN** 1

  Unknown.

21.13.1 Detailed Description

21.14 Configuration Server

Functions

- void EnableConfigMirror ()

  Enable the configuration mirror.
- void SaveConfigToStorage ()

  Write all pending data to flash memory.
- size_t ConfigSize ()

  Returns the number of bytes currently in use by configuration flash.
- size_t ConfigMaxSize ()

  Returns the number of bytes available in configuration flash.

21.14.1 Detailed Description

The configuration server manages the configuration variables, which can be used internally, or optionally presented on a configuration server web page. Please refer to the configuration object section for more details.
There is a default factory web interface that is normally running on the device as a user interface to set configuration options. Enabling the Configuration Mirror provides a method to replace the factory interface with your own custom interface. The is an easy way to use your company brand on the device.
Any time a configuration variable is modified, it is marked as pending. Modified values will not be written to flash memory until a call to SaveConfigToStorage() is executed. Note that if a user clicks on the Update button on the configuration web page, values will be written to flash.

21.14.2 Function Documentation

21.14.2.1 ConfigMaxSize()

`size_t ConfigMaxSize ( )`
Returns the number of bytes available in configuration flash.

Returns

The maximum number of bytes available. This function is implemented in hal.cpp

21.14.2.2 ConfigSize()

```
size_t ConfigSize ( )
```

Returns the number of bytes currently in use by configuration flash.

Returns

Number of bytes in use

21.14.2.3 EnableConfigMirror()

```
void EnableConfigMirror ( )
```

Enable the configuration mirror.
When enabled, the application configuration web interface will replace the default system web interface. In this way you can easily customize and brand the configuration web interface for your company.

Note

This is an empty function. To enable the config mirror, it needs to be called within the application code, which in turn forces the function to be linked during compilation. The linking is what enables the config mirror.

21.14.2.4 SaveConfigToStorage()

```
void SaveConfigToStorage ( )
```

Write all pending data to flash memory.
Checks the pending flag of all configuration system objects and writes updates to flash memory

21.15 Configuration Variable Flags

Variables

- const uint32_t **fConfigValueLeaf** = 0x01

  Value is a leaf.
- const uint32_t **fConfigReadOnly** = 0x02

  Variable is read-only.
- const uint32_t **fConfigModified** = 0x04

  Variable has been modified, but not yet saved.
- const uint32_t **fConfigHidden** = 0x08

  Not visible to configuration web server display.
- const uint32_t **fConfigNoSave** = 0x10

  Do not save to flash memory when save functions are called.
- const uint32_t **fConfigNoObscure** = 0x20

  Do not obscure the value.
- const uint32_t **fConfigNeedReboot** = 0x40

  System reboot required for changes to take effect.
- const uint32_t **fConfigNoReload** = 0x80

  Disable reloading value from flash during a call to ReloadFromFlash.
- const uint32_t **fConfigDetached** = 0x100

  Disable reloading value from flash during a call to ReloadFromFlash.

21.15.1 Detailed Description

Specifies the characteristics of a configuration variable, such as read-only, hidden, etc.

21.16 Configuration Variables

Configuration System Variables.

Modules

- Configuration Variable Flags

Classes

- class config_obj

  Base class used to create configuration objects.
- class config_value

  Base class used to create a configuration value.
- class config_uint

  Unsigned 32-bit Integer Configuration Variable.
- class config_int

  Signed 32-bit Integer Configuration Variable.
- class config_double

  Double Float Configuration Variable.
- class config_bool

  Boolean Configuration Variable.
- class config_string

  String Configuration Variable.
- class config_pass

  Password string Configuration Variable.
- class config_IPADDR4

  Configuration Variable for IPADDR4 (IPv4) object types.
- class config_IPADDR

  Configuration Variable for IPADDR (IPv6) object type.
- class config_MACADR

  Configuration Variable for MACADR object type.
- class config_chooser

  Chooser Configuration Variable - Select From a List of Items.

Functions

- void SaveConfigToStorage ()

  Save configuration to flash storage.

21.16.1 Detailed Description

Configuration System Variables.
Configuration variable objects are part of the NetBurner 3.x configuration system. The use of configuration objects facilitates a much easier method to create and store non-volatile information for both system configuration and for user application configuration and storage. They use a JSON interface so the information can be easily downloaded to save a configuration, or uploaded to configure the entire device at one time, such as in production bring-up. This includes system configuration such as IP addresses, and also any settings specific to your application.
The JSON value types include: integer, unsigned integer, float, boolean and string. There are also objects for IPv4 IP, IPv6 IP, MAC addresses, passwords, and selection lists options. In JSON everything is represented as a string, so there are object member functions to convert between values. These functions make use of the NBString class. For customers using the 2.x tools, the User Parameter flash sector method is still available, but we do recommend converting to the new configuration system. In 3.x you have the ability to access the data through a web configuration page and also through JSON, but you can certainly choose not to use those features and treat the configuration system as internal storage, and any configuration can be done through your application just as with the 2.x tool set.

Note

All modifications to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

21.16.2 Function Documentation

21.16.2.1 SaveConfigToStorage()

```
void SaveConfigToStorage ( )
```

Save configuration to flash storage.

21.17 Connect Request Errors

Macros

- #define **CONNECT_ERR_SUCCESS** 0

  Successfull connect request.
- #define **CONNECT_ERR_NOT_CONFIG** -1

  Paramaters not configured.
- #define **CONNECT_ERR_INVALID_CONFIG_NUM** -2

  Invalid config number.
- #define **CONNECT_ERR_CONNECTED** -3

  Already connected.
- #define **CONNECT_ERR_SSID_NOT_FOUND** -4

  SSID not found.
- #define **CONNECT_ERR_BSSID_NOT_FOUND** -5

  BSSID not found.
- #define **CONNECT_ERR_SEC_MISMATCH** -6

  Security mismatch.
- #define **CONNECT_ERR_CIPH_MISMATCH** -7

  Cipher mismatch.
- #define **CONNECT_ERR_INVALID_KEY** -8

  Invalid password.
- #define **CONNECT_ERR_UNKNOWN** 1

  Unknown error.

21.17.1 Detailed Description

21.18 DHCP - IPv4 DHCP Client

Modules

- DHCP State

Classes

- class DhcpObject

  DHCP client class.

Functions

- int32_t GetIntefaceDHCPState (int interface=0)

  Returns current state of the DHCP lease, with optional interface parameter.
- int32_t WaitForDHCPInterface (int interface=0, uint16_t TicksToWait=10 ∗TICKS_PER_SECOND)

  Wait until a DHCP lease is obtained, or the timeout occurs.

21.18.1 Detailed Description

IPv4 DHCP Client functions. Refer to the IPv6 documents for DHCPv6.
The DHCP client services are automatically controlled by the configuration option when the init() function is called in an application. Options are:

- DHCP
- DHCP with Static Fallback
- Static

You may also create your own DHCP objects.

21.18.2 Function Documentation

21.18.2.1 GetIntefaceDHCPState()

```
int32_t GetIntefaceDHCPState (
            int interface = 0 )
```

Returns current state of the DHCP lease, with optional interface parameter.
The GetDHCPState() function is a member function of the DHCP client class.

Returns

DHCP State

See also

GetDHCPState()

21.18.2.2 WaitForDHCPInterface()

```
int32_t WaitForDHCPInterface (
            int interface = 0,
            uint16_t TicksToWait = 10 *TICKS_PER_SECOND )
```

Wait until a DHCP lease is obtained, or the timeout occurs.

Parameters

interface	The network interface to wait on.
TicksToWait	The number of system Time Ticks to wait. The TICKS_PER_SECOND constant can be used to more easily specify a number of seconds. If not specified, the default timeout is 10 seconds.

Returns

DHCP State

See also

GetDHCPState()

21.19 DHCP Server

Functions

- bool AddStandardDHCPServer (int intf=0, IPADDR4 startAddr=IPADDR4::NullIP())

  Starts a standard allocator DHCP server.

21.19.1 Detailed Description

Adding a DHCP server will enable your NetBurner device to serve IPv4 addresses. This can be useful in circumstances such as direct connections between 2 devices.

21.19.2 Function Documentation

21.19.2.1 AddStandardDHCPServer()

```
bool AddStandardDHCPServer (
            int intf = 0,
            IPADDR4 startAddr = IPADDR4::NullIP() )
```

Starts a standard allocator DHCP server.
Also checks for existing DHCP servers on the specified network interface.

Parameters

intf	The interface to use. If no parameter is specified the default value is 0 which will use the first system interface.
startAddr	The starting IP address. If no parameter is specified the default range of 192.168.1.100 to 192.168.1.249 will be used.

Return values

true	Success
false	A server already exists or error starting the server

21.20 DHCP State

Macros

- #define **SDHCP_NOTSTARTED** 0

  The System has not been initialized.
- #define **SDHCP_DISCOVER** 1

  The system is discovering the DHCP servers.
- #define **SDHCP_OFFER** 2

  The system has responded to an OFFER.
- #define **SDHCP_ACK** 3

  The System has Acknowledged the OFFER.
- #define **SDHCP_INIT** 4

  The System is reinitalzing.

- #define **SDHCP_CMPL** 5

  The System has obtained a valid DHCP lease.
- #define **SDHCP_RENEW** 6

  The System is in the process of renewing.
- #define **SDHCP_REBIND** 7

  The System has failed the Renew and is trying to Rebind.
- #define **SDHCP_RELEASE** 8

  The System is trying to release the Lease.
- #define **SBOOTP_TRANSMITTING** 9

  Trying BOOTP.
- #define **SBOOTP_DONE** 10

  BOOTP complete.
- #define **SDHCP_FAILED** 11

  DHCP attempt failed - could not obtain a DHCP lease.

21.20.1 Detailed Description

21.21 DHCPv6

Functions

- void StartDHCPv6_Solicit (int ifnum=-1)

  Manually starts the DHCPv6 Client in Full Solicitation mode.
- void StartDHCPv6_InfoReq (int ifnum=-1)

  Manually starts the DHCPv6 Client in Full Solicitation mode.
- void StartDHCPv6 (int ifnum=-1)

  Manually starts the DHCPv6 Client in Information Request mode.
- bool AddStaticIPv6Address (const IPADDR6 &ip, int ifnum=-1)

  Add a static IPv6 address to an interface.
- bool RemoveStaticIPv6Address (const IPADDR6 &ip, int ifnum=-1)

  Add a static IPv6 address to an interface.

21.21.1 Detailed Description

21.21.2 Function Documentation

21.21.2.1 AddStaticIPv6Address()

```
bool AddStaticIPv6Address (
            const IPADDR6 & ip,
            int ifnum = -1 )
```

Add a static IPv6 address to an interface.

Parameters

ip	The IP address to add.
ifnum	The desired IPv6Interface to operate on. A negative value will use the first interface.

Return values

true	IP address successfully added
false	Error

See also

RemoveStaticIPv6Address()

21.21.2.2 RemoveStaticIPv6Address()

```
bool RemoveStaticIPv6Address (
            const IPADDR6 & ip,
            int ifnum = -1 )
```

Add a static IPv6 address to an interface.

Parameters

ip	The IP address to remove.
ifnum	The desired IPv6Interface to operate on. A negative value will use the first interface.

Return values

true	IP address successfully removed
false	Error

See also

AddStaticIPv6Address()

21.21.2.3 StartDHCPv6()

```
void StartDHCPv6 (
            int ifnum = -1 )  [inline]
```

Manually starts the DHCPv6 Client in Information Request mode.

Parameters

ifnum	The desired IPv6Interface to operate on. A negative value indicates all interfaces

See also

StartDHCPv6_Solicit()

21.21.2.4 StartDHCPv6_InfoReq()

```
void StartDHCPv6_InfoReq (
            int ifnum = -1 )
```

Manually starts the DHCPv6 Client in Full Solicitation mode.

Parameters

ifnum	The desired IPv6Interface to operate on. A negative value indicates all interfaces

21.21.2.5 StartDHCPv6_Solicit()

```
void StartDHCPv6_Solicit (
            int ifnum = -1 )
```

Manually starts the DHCPv6 Client in Full Solicitation mode.

Parameters

ifnum	The desired IPv6Interface to operate on. A negative value indicates all interfaces

See also

StartDHCPv6()

21.22 DNS - Domain Name System

Modules

- DNS Record Types
- DNS Return Codes

Functions

- bool IsNameIPAddress (const char *name)

  Determine if the name is a valid IP Address and does not need to be looked up.
- int fd_dns_part1 (const char *name, const IPADDR &dns_server, uint16_t TYPE=DNS_A, uint16_t TYPE2=0, int ifn=-1)

  Open a UDP socket and initiate a DNS lookup.
- bool fd_dns_processresult (int fd, const char *name, IPADDR &addr_out, uint16_t TYPE=DNS_A, uint16_t TYPE2=DNS_AAAA, uint32_t *ttl=0)

  Process any responses on the UDP socket opened for DNS.
- int fd_outstanding_Responses (int fd)

  Check to see if there are any outstanding DNS requests.
- bool AnyDNSInterFaceActive ()

  Determine if we have an active DNS route; DNS server is set for an active interface.
- int GetHostByName (const char *name, IPADDR *pIpaddr, const IPADDR &dns_server, const TickTimeout tout, uint16_t TYPE1=DNS_A, uint16_t TYPE2=extra_dns_t, uint32_t *ttl=NULL)

  Get the IP address associated with the specified domain name.
- int GetHostByNameViaIfNum (const char *name, IPADDR *pIpaddr, const IPADDR &dns_server, int ifn, const TickTimeout &tout, uint16_t TYPE1=DNS_A, uint16_t TYPE2=extra_dns_t, uint32_t *ttl=NULL)

  Get the IP address associated with the specified domain name on a specific interface.

21.22.1 Detailed Description

The current DNS system is agnostic about the use of IPADDR, IPADDR6 or IPADDR4. The capability of C++ is such that you pass in any of those types and the compiler will select the correct function for that type. This is a change from tools versions 3.4.0 and earlier in which explicit IPv4 and IPv6 functions were used.

Note

While we are pointing out the automatic functionality in this DNS API, the same is true for most functions, structures and classes throughout the NetBurner API libraries.

The DNS system can be accessed as a blocking or synchronous operation with: GetHostByName(). If you want to do asynchronous non-blocking lookups, please refer to the functions:

- fd_dns_part1()
- fd_dns_processresult()

21.22.2 Function Documentation

21.22.2.1 AnyDNSInterFaceActive()

```
bool AnyDNSInterFaceActive ( )
```

Determine if we have an active DNS route; DNS server is set for an active interface.

Returns

true if active DNS is available somewhere

See also

GetHostByName()

InterfaceLinkActive(int interface);

21.22.2.2 fd_dns_part1()

```
int fd_dns_part1 (
            const char * name,
            const IPADDR & dns_server,
            uint16_t TYPE = DNS_A,
            uint16_t TYPE2 = 0,
            int ifn = -1 )
```

Open a UDP socket and initiate a DNS lookup.
This socket can be put in a select and used for asynchronous DNS lookups. Opens a UDP receive FD and sends DNS queries.

Parameters

name	Pointer to domain name to resolve
dns_server	Reference to the DNS server to use. Pass INADDR_ANY to use the DNS server associated with the interface.
TYPE	Optional DNS record type. If no type is specified the default is IPv4 DNS_A
TYPE2	Optional 2nd DNS record type. If no type is specified the default is no look up
ifn	Which interface to use -1 uses the default active DNS interface

Returns

fd The file descriptor of the udp socket. DNS Return Codes

See also

fd_dns_processresult(), fd_outstanding_Responses(), GetHostByName(),

21.22.2.3 fd_dns_processresult()

```
bool fd_dns_processresult (
            int fd,
            const char * name,
            IPADDR & addr_out,
            uint16_t TYPE = DNS_A,
            uint16_t TYPE2 = DNS_AAAA,
            uint32_t * ttl = 0 )
```

Process any responses on the UDP socket opened for DNS.
This should only be called when the socket has data, otherwise it will block.

Parameters

fd	The file descriptor opened with fd_dns_part1
name	Pointer to domain name to resolve
addr_out	Reference to IPADDR variable to hold the result
TYPE	DNS record type. If no type is specified the default is IPv4 DNS_A
TYPE2	2nd DNS record type. If no type is specified the default is no look up
ttl	Optional pointer to return variable for TimeToLive value for DNS record

Returns

true if resolution was successful

See also

fd_dns_part1(), fd_outstanding_Responses(), GetHostByName(),

21.22.2.4 fd_outstanding_Responses()

```
int fd_outstanding_Responses (
            int fd )
```

Check to see if there are any outstanding DNS requests.
If this returns 0 then all DNS responses have been processed and one can assume ther eis no such name.

Parameters

fd	The file descriptor opened with fd_dns_part1

Returns

The number of outstanding DNS requests that have not been answered.

See also

fd_dns_part1(), fd_dns_processresult(), GetHostByName(),

21.22.2.5 GetHostByName()

```
int GetHostByName (
            const char * name,
            IPADDR * pIpaddr,
            const IPADDR & dns_server,
            const TickTimeout tout,
            uint16_t TYPE1 = DNS_A,
            uint16_t TYPE2 = extra_dns_t,
            uint32_t * ttl = NULL )  [inline]
```

Get the IP address associated with the specified domain name.
Calling GetHostByName() in dual stack mode will automatically call the correct IPv4 or IPv6 function.
The function will attempt to retrieve record type 1 first. If successful the function returns. If if the attempt using type 1 fails, the function will attempt to get a record using the type 2 parameter. For example, lets say you wish your code to use IPv6, but its OK to fall back to IPv4. You can set type 1 to DNS_AAAA and type2 to DNS_A.

Parameters

∗name	Pointer to domain name to resolve
∗plpaddr	Pointer to variable of type IPADDR (or IPADDR4) to store resultant IP address.
&dns_server	Specified the DNS server to use. Pass INADDR_ANY to use the DNS server associated with the interface.
tout	Number of time ticks to wait of type TickTimeout
TYPE1	Optional record type. If not specified will default to IPv4 DNS_A
TYPE2	Optional record type. If not specified will default to IPv6 DNS_AAAA
ttl	Optional pointer to return variable for TimeToLive value for DNS record

Returns

DNS Return Codes

See also

GetHostByName4()

GetHostByName6()

21.22.2.6 GetHostByNameViaIfNum()

```
int GetHostByNameViaIfNum (
            const char * name,
            IPADDR * pIpaddr,
            const IPADDR & dns_server,
            int ifn,
            const TickTimeout & tout,
            uint16_t TYPE1 = DNS_A,
            uint16_t TYPE2 = extra_dns_t,
            uint32_t * ttl = NULL )  [inline]
```

Get the IP address associated with the specified domain name on a specific interface.
Calling GetHostByName() in dual stack mode will automatically call the correct IPv4 or IPv6 function.
The function will attempt to retrieve record type 1 first. If successful the function returns. If if the attempt using type 1 fails, the function will attempt to get a record using the type 2 parameter. For example, lets say you wish your code to use IPv6, but its OK to fall back to IPv4. You can set type 1 to DNS_AAAA and type2 to DNS_A.

Parameters

∗name	Pointer to domain name to resolve
∗plpaddr	Pointer to variable of type IPADDR to store resultant IP address
&dns_server	Specified the DNS server to use. Pass INADDR_ANY to use the DNS server associated with the interface.
tout	A reference to the number of time ticks to wait of type TickTimeout
TYPE1	Optional record type. If not specified will default to IPv4 DNS_A
TYPE2	Optional record type. If not specified will default to IPv6 DNS_AAAA
ifn	Interface number
ttl	Optional pointer to return variable for TimeToLive value for DNS record

Returns

DNS Return Codes

See also

GetHostByName4ViaIfNum()

GetHostByNameViaIfNum()

21.22.2.7 IsNameIPAddress()

```
bool IsNameIPAddress (
            const char * name )
```

Determine if the name is a valid IP Address and does not need to be looked up.

Parameters

**name*	Pointer to domain name to check

Returns

true if it is a valid IP Address

See also

GetHostByName(), GetHostByName4ViaIfNum(), GetHostByName6()

21.23 DNS Record Types

Macros

- #define **DNS_A** 1
  32-bit IPv4 address
- #define **DNS_CNAME** 5
  Canonical name record.
- #define **DNS_MB** 7
  Mailing list subscriber list.
- #define **DNS_MG** 8
  Mailing list subscriber list.
- #define **DNS_MX** 15
  Mail exchange record.
- #define **DNS_AAAA** 28
  128-bit IPv6 address

21.23.1 Detailed Description

DNS record types from RFC1035

21.24 DNS Return Codes

Macros

- #define **DNS_OK** (0)
  Success.
- #define **DNS_TIMEOUT** (1)
  Request timed out.
- #define **DNS_NOSUCHNAME** (2)
  Name not found.
- #define **DNS_ERR** (3)
  Other error.

21.24.1 Detailed Description

21.25 DSPI state

21.25.1 Detailed Description

21.26 DspiModuleNumber

DSPI Peripheral Module.

Macros

- #define **DEFAULT_DSPI_MODULE** 0

  Default DSPI module.
- #define **DSPI_MODULE_COUNT** 1

  Number of modules: 0, 1.

21.26.1 Detailed Description

DSPI Peripheral Module.

21.27 DspiState

DSPI Bus State .

Macros

- #define **DSPI_OK** (0)

  DSPI OK.
- #define **DSPI_BUSY** (1)

  DSPI Busy.
- #define **DSPI_ERROR** (2)

  DSPI Error.

21.27.1 Detailed Description

DSPI Bus State .

21.28 EFFS - Embedded Flash File System

Modules

- EFFS-STD Flash File System
- FAT File System

21.28.1 Detailed Description

Two Embedded Flash File Systems are supported:

- FAT32 File System (EFFS-FAT), a FAT32 file system used on external flash memory cards and ram drives.
- Standard File System (EFFS-STD), a power-fail safe file system that resides in the flash on the module.

21.29 EFFS-STD Flash File System

Modules

- STD File System Seek Codes

Macros

- #define fs_getfreespace(drivenum, space) fsm_getfreespace(drivenum, space)

  Provides information about the drive space usage.
- #define fs_mkdir(dirname) fsm_mkdir(dirname)

  Makes a new directory.
- #define fs_chdir(dirname) fsm_chdir(dirname)

  Change the directory.
- #define fs_rmdir(dirname) fsm_rmdir(dirname)

  Removes a directory.
- #define fs_delete(filename) fsm_delete(filename)

  Deletes a file.
- #define fs_findfirst(filename, find) fsm_findfirst(filename, find)

  Find the first file or subdirectory in a specified directory.
- #define fs_findnext(find) fsm_findnext(find)

  Finds the next file or subdirectory in a specified directory after a previous call to `fs_findfirst()` or `fs_findnext()`.
- #define fs_open(filename, mode) fsm_open(filename, mode)

  Opens a file in the file system.
- #define fs_close(filehandle) fsm_close(filehandle)

  Closes an opened file.
- #define fs_write(buf, size, size_st, filehandle) fsm_write(buf, size, size_st, filehandle)

  Write data to the file at the current position.
- #define fs_read(buf, size, size_st, filehandle) fsm_read(buf, size, size_st, filehandle)

  Read data from the current position in a file.
- #define fs_seek(filehandle, offset, whence) fsm_seek(filehandle, offset, whence)

  Move the stream position of an open file.
- #define fs_eof(filehandle) fsm_eof(filehandle)

  Check whether the current position in the open target file is the end of the file.
- #define fs_rewind(filehandle) fsm_rewind(filehandle)

  Sets the file position in the open target file to the start of the file.
- #define fs_settimedate(filename, ctime, cdate) fsm_settimedate(filename, ctime, cdate)

  Set the time and date of a file or directory.
- #define fs_gettimedate(filename, pctime, pcdate) fsm_gettimedate(filename, pctime, pcdate)

  Get the time and date of a file or directory.

21.29.1 Detailed Description

The Standard Embedded Flash File System, EFFS-STD

21.29.2 Macro Definition Documentation

21.29.2.1 fs_chdir

```
#define fs_chdir(
              dirname ) fsm_chdir(dirname)
```

Change the directory.

Parameters

dirname	The directory to change to.

Return values

FS_NOERR	If able to change to the directory
!FS_NOERR	If unable to change to the directory

See also

fs_mkdir()

fs_rmdir()

21.29.2.2 fs_close

```
#define fs_close(
              filehandle ) fsm_close(filehandle)
```

Closes an opened file.

Parameters

filehandle	A handle to the file to close.

Return values

FS_NOERR	If able to close the file
!FS_NOERR	If unable to close the file

21.29.2.3 fs_delete

```
#define fs_delete(
              filename ) fsm_delete(filename)
```

Deletes a file.
A read-only or open file cannot be deleted.

Parameters

filename	The name of the file to be deleted.

Return values

FS_NOERR	If able to delete the file
!FS_NOERR	If unable to delete the file

21.29.2.4 fs_eof

```
#define fs_eof(
```

```
            filehandle ) fsm_eof(filehandle)
```

Check whether the current position in the open target file is the end of the file.

Parameters

filehandle	A handle to the file to check.

Return values

0	Nothing at the end of the file
!0	The end of the file, or an error

21.29.2.5 fs_findfirst

```
#define fs_findfirst(
            filename,
            find ) fsm_findfirst(filename, find)
```

Find the first file or subdirectory in a specified directory.
Note: If this is called with "∗.∗" and this is not the root directory, the first entry found will be ".", which is the current directory.

Parameters

filename	The name of the file to find.
find	Where to store the found information.

Return values

FS_NOERR	If the file was found
!FS_NOERR	If the file wasn't found or there are errors

See also

fs_findnext()

21.29.2.6 fs_findnext

```
#define fs_findnext(
            find ) fsm_findnext(find)
```

Finds the next file or subdirectory in a specified directory after a previous call to `fs_findfirst()` or `fs_findnext()`.
Note: If this is called with "∗.∗" and this is not the root directory, the first entry found will be ".", which is the current directory.

Parameters

find	Where to store the found information.

Return values

FS_NOERR	If the file was found

Return values

!FS_NOERR	If the file wasn't found or there are errors

See also

fs_findfirst()

21.29.2.7 fs_getfreespace

```
#define fs_getfreespace(
              drivenum,
              space ) fsm_getfreespace(drivenum, space)
```

Provides information about the drive space usage.

Parameters

drivenum	The drive number to get space usage for.
space	The structure to store the usage information in.

Return values

FS_NOERR	If able to get the drive space usage
!FS_NOERR	If unable to get the drive space usage

21.29.2.8 fs_gettimedate

```
#define fs_gettimedate(
              filename,
              pctime,
              pcdate ) fsm_gettimedate(filename, pctime, pcdate)
```

Get the time and date of a file or directory.

Parameters

filename	The name of the file or directory to modify.
pctime	A pointer to store the creation time of the file or directory.
pcdate	A pointer to store the creation date of the file or directory.

Return values

FS_NOERR	If able to get time and date
!FS_NOERR	If unable to get time and date

21.29.2.9 fs_mkdir

```
#define fs_mkdir(
              dirname ) fsm_mkdir(dirname)
```

Makes a new directory.

Parameters

dirname	The name to use for the new directory.

Return values

FS_NOERR	If able to make the directory
!FS_NOERR	If unable to make the directory

See also

fs_chdir()

fs_rmdir()

21.29.2.10 fs_open

```
#define fs_open(
              filename,
              mode ) fsm_open(filename, mode)
```

Opens a file in the file system.

Parameters

filename	The name of the file to open.
mode	The mode to open the file in. The following options are available: • "r": Reading. The stream is positioned at the beginning of the file. • "r+": Reading and writing. The stream is positioned at the beginning of the file. • "w": Truncate file to 0 length or create for writing. The stream is positioned at the beginning of the file. • "w+": Reading and writing. Create if it doesn't exist, or truncate if it does. The stream is positioned at the beginning of the file. • "a": Append, and create if it doesn't exist. The stream is positioned at the end of the file. • "a+": Reading and appending, and create if it doesn't exist. The stream is positioned at the end of the file.

Return values

*FS_FILE**	Pointer to the file if it could be opened
0	If it a file could not be opened

See also

fs_close()

21.29.2.11 fs_read

```
#define fs_read(
```

```
            buf,
            size,
            size_st,
            filehandle ) fsm_read(buf, size, size_st, filehandle)
```

Read data from the current position in a file.

Parameters

buf	A pointer where the data should be read.
size	The size of the items to be read.
size_st	The number of items to be read.
filehandle	A handle to the file to read to.

Returns

The number of items read.

21.29.2.12 fs_rewind

```
#define fs_rewind(
            filehandle ) fsm_rewind(filehandle)
```

Sets the file position in the open target file to the start of the file.

Parameters

filehandle	A handle to the file operate on.

Return values

FS_NOERR	If successful
!FS_NOERR	If there was an error

21.29.2.13 fs_rmdir

```
#define fs_rmdir(
            dirname ) fsm_rmdir(dirname)
```

Removes a directory.
Note: The target directory must be empty, otherwise an error code is returned.

Parameters

dirname	The name of the directory to remove.

Return values

FS_NOERR	If able to get the drive space usage
!FS_NOERR	If unable to get the drive space usage

See also

fs_chdir()

fs_mkdir()

21.29.2.14 fs_seek

```
#define fs_seek(
              filehandle,
              offset,
              whence ) fsm_seek(filehandle, offset, whence)
```

Move the stream position of an open file.

Parameters

filehandle	A handle to the file to move the position of.
offset	The relative byte position according to whence.
whence	Where to calculate the offset from. Can be a value from fSeekCodeGroup.

Return values

FS_NOERR	If the seek operation was successful
!FS_NOERR	If there was an error

21.29.2.15 fs_settimedate

```
#define fs_settimedate(
              filename,
              ctime,
              cdate ) fsm_settimedate(filename, ctime, cdate)
```

Set the time and date of a file or directory.

Parameters

filename	The name of the file or directory to modify.
ctime	The creation time of the file or directory.
cdate	The creation date of the file or directory.

Return values

FS_NOERR	If able to set time and date
!FS_NOERR	If unable to set time and date

21.29.2.16 fs_write

```
#define fs_write(
              buf,
              size,
              size_st,
```

filehandle) fsm_write(buf, size, size_st, filehandle)

Write data to the file at the current position.

Parameters

buf	A pointer to the data to write.
size	The size of the items to write.
size_st	The number of items to write.
filehandle	A handle to the file to write to.

Returns

The number of items written.

21.30 Ethernet

Modules

- Ethernet Interface Types

Macros

- #define NO_AUTOMATIC_2ND_ETHERNET extern const bool bAutomatic2ndEther = false;

  Disable automatic initialization of second Ethernet interface.

Functions

- void AddEthernetInterfaces ()

  Add an Ethernet interface.
- void ManualEthernetConfig (int interface, BOOL speed100Mbit, BOOL fullDuplex, BOOL autoNegotiate)

  Manually configure Ethernet speed and duplex settings.
- void DisablePHY (int ifn)

  Disable the specified Ethernet PHY.
- void EnablePHY (int ifn)

  Disable the specified Ethernet PHY.

21.30.1 Detailed Description

Functions to add, configure and disable Ethernet interfaces

21.30.2 Macro Definition Documentation

21.30.2.1 NO_AUTOMATIC_2ND_ETHERNET

```
#define NO_AUTOMATIC_2ND_ETHERNET extern const bool bAutomatic2ndEther = false;
```

Disable automatic initialization of second Ethernet interface.
Include this macro in UserMain to prevent automatic system addition and initialization of the second Ethernet interface on dual Ethenet devices.

21.30.3 Function Documentation

21.30.3.1 AddEthernetInterfaces()

```
void AddEthernetInterfaces ( )
```

Add an Ethernet interface.
The default system behavior is all Ethernet interfaces will automatically be added and initialized. Interface numbers range from 1 to (MAX_INTERFACES - 1)

21.30.3.2 DisablePHY()

```
void DisablePHY (
            int ifn )
```

Disable the specified Ethernet PHY.
Typically used for low power mode.

Parameters

ifn	Interface number to disable

See also

EnablePHY

21.30.3.3 EnablePHY()

```
void EnablePHY (
            int ifn )
```

Disable the specified Ethernet PHY.
Enable an Ethernet PHY previously disabled by DisablePHY()

Parameters

ifn	Interface number to ensable

See also

DisablePHY()

21.30.3.4 ManualEthernetConfig()

```
void ManualEthernetConfig (
            int interface,
            BOOL speed100Mbit,
            BOOL fullDuplex,
            BOOL autoNegotiate )
```

Manually configure Ethernet speed and duplex settings.
The default setting to establish an Ethernet link is autonegotiate. This function can be used to disable autonegotiate and set the Ethernet link speed and duplex.

Warning

With autonegotiate disabled, the other Ethernet host must also be configured manually with the same settings. If the other host is set to autonegotiate, the link will have undefined behavior and data loss.

Parameters

interface	The network interface to modify. The first interface number is 1

Parameters

speed100Mbit	True = 100Mbps, False = 10Mbps
fullDuplex	True = full duplex, False = half duplex
autoNegotiate	True = enabled, False = disabled. If enabled, will override manual settings.

21.31 Ethernet Interface Types

Macros

- #define **ETHERNET_ETHERTYPE_IPv4** (uint16_t)(0x0800)

  Internet Protocol, Version 4 (IPv4)
- #define **ETHERNET_ETHERTYPE_ARP** (uint16_t)(0x0806)

  Address Resolution Protocol (ARP)
- #define **ETHERNET_ETHERTYPE_IPv6** (uint16_t)(0x86DD)

  Internet Protocol, Version 6 (IPv6)
- #define **ETHERNET_ETHERTYPE_AARP** (uint16_t)(0x80F3)

  AppleTalk Address Resolution Prot. (AARP)
- #define **ETHERNET_ETHERTYPE_IPX** (uint16_t)(0x8137)

  Novell Internet Packet Exchange (IPX) (alt.)
- #define **ETHERNET_ETHERTYPE_EAPOL** (uint16_t)(0x888E)

  Extensible Authorization Protocol (EAP) over LAN.
- #define **ETHERNET_ETHERTYPE_VLAN** (uint16_t)(0x8100)

  Virtual Private Network (VLAN)

21.31.1 Detailed Description

EtherType Field (Ethernet Version II)

21.32 External Bus Interface (EBI)

Classes

- struct EBI_CS_cfg_t

  Configuration structure for an External Bus Interface (EBI) chip select.

Enumerations

- enum EBI_CS_BusWidth_t { EBI_BUS_WIDTH_8 = 0 , EBI_BUS_WIDTH_16 = 1 }
- enum EBI_CS_ByteAccess_t { EBI_BYTE_ACCESS_SELECT = 0 , EBI_BYTE_ACCESS_WRITE = 1 }
- enum EBI_CS_NWait_t { EBI_NWAIT_DISABLED = 0 , EBI_NWAIT_FROZEN = 2 , EBI_NWAIT_READY = 3 }
- enum EBI_CS_WrMode_t
- enum EBI_CS_RdMode_t

Functions

- void ConfigureEBI_CSPin (int csNum)

  Configure the I/O pin for a given Chip Select for the external data bus.
- void **ConfigureEBI_NWRPin** ()

  Configure the I/O pin for the active low write/read (NWR) bus signal.
- void **ConfigureEBI_NRDPin** ()

Configure the I/O pin for the active low read (NRD) bus signal.

- void ConfigureEBI_CS (uint32_t csNum, const EBI_CS_cfg_t &&cfg)

  Configure the given Chip Select for the external data bus.
- void ConfigureEBI_CS (uint32_t csNum, const EBI_CS_cfg_t &cfg)

  Configure the given Chip Select for the external data bus.

Variables

- uint8_t EBI_CS_cfg_t::ncs_rd_setup
- uint8_t EBI_CS_cfg_t::nrd_setup
- uint8_t EBI_CS_cfg_t::ncs_wr_setup
- uint8_t EBI_CS_cfg_t::nwe_setup
- uint8_t EBI_CS_cfg_t::ncs_rd_pulse
- uint8_t EBI_CS_cfg_t::nrd_pulse
- uint8_t EBI_CS_cfg_t::ncs_wr_pulse
- uint8_t EBI_CS_cfg_t::nwe_pulse
- uint16_t EBI_CS_cfg_t::nrd_cycles
- uint16_t EBI_CS_cfg_t::nwe_cycles
- uint8_t EBI_CS_cfg_t::tdf_cycles
- EBI_CS_BusWidth_t EBI_CS_cfg_t::busWidth
- EBI_CS_ByteAccess_t EBI_CS_cfg_t::byteAccess
- EBI_CS_NWait_t EBI_CS_cfg_t::nWait
- EBI_CS_WrMode_t EBI_CS_cfg_t::wrMode
- EBI_CS_RdMode_t EBI_CS_cfg_t::rdMode

21.32.1 Detailed Description

The Microchip SAME70 EBI peripheral is a parallel communication port that can only operate in "host" mode to communicate with "slave" devices. The EBI peripheral consists of a set of up to 16 data signals, up to 24 address signals and up to 10 different control signals. Some of the benefits of using the EBI bus are that it has high performance in both throughput and latency. It is directly memory mapped to the processor so it is very easy to use in software once configured. The downsides are mostly that it has many signals which will greatly complicate hardware designs and increase the size of external peripheral chips.
The EBI peripheral is highly configurable to fit the specifications of many types of external devices.
A consequence to this high configurablity is that there are many timing parameters which makes configuration complicated. The NetBurner examples specify a conservative set of timing parameters with the goal of being compatible with a wide range of peripherals such as FPGAs, CPLDs, buffers, latches displays, flash memory and SRAM. Once successful communication is obtained with the slave target device, these default settings should be refined to increase the performance to match the device you are interfacing with.
Advanced EBI functionality is currently not supported by this examples or this driver. Advaced functionality includes interfacing with SDRAM or NAND memories. Page mode, burst operations and scrambling functionality are all not currently configurable via this driver.

In most applications you will only need to verify the last 5 parameters meet your peripheral requirements: bus width (8 or 16), byte Access 16-bit mode only), timing parameters, read mode (chip select or read signal) and write mode (chip select or write signal).
There are 3 sources of information that are very useful for understanding the EBI:

1. The EBI section of the Microchip SAME70 manual located in "\nburn\docs". It is very important to do this as the first step to understand the signal terminology in the examples and header files. Additionally the Static Memory Controller chapter in this processor manual is essential as it explains all the settings and timing parameters exposed by this driver.

2. The NetBurner EBI examples described in the manual and located in "\nburn\examples\PlatformSpecific\↩ MODM7AE70"

3. The SAME70 ebi.h header file located in "\nburn\arch\cortex-m7\cpu\SAME70\include"

The EBI supports Chip Selects as well as Read and Write signals to interface with peripherals. The basic procedure for configuring the bus is:

- As noted in the examples, the MODM7AE70 has onboard bus buffers to reduce noise and help signal integrity. The following 4 signals must always be configured for EBI functionality as they are control the enable/disable and direction control of the bus buffer ICs:

```
P1[5].function(PINP1_5_NCS2);       // chip select 2, active low
P1[6].function(PINP1_6_NCS0);       // chip select 0, active low
P1[7].function(PINP1_7_NCS3);       // chip select 3, active low
P1[8].function(PINP1_8_NRD);        // Bus read, active low
```

- Declare a Configuration Structure of type EBI_CS_cfg_t. An easy way is to copy one from the examples. Then modify any parameters necessary to meet your specific peripheral requirements, such as bus width, byte access in 16-bit mode, timing parameters, etc.

- Configure the desired chip select using the ConfigureEBI_CS()function. Note that when enabling the EBI, the MODM7AE70 module requires all three chip selects (NCS0, NCS02, NCS03) to be assigned as chip selects. Addtionally the NRD signal must also be configured for NRD bus operation. All of these signals are used for controlling the enable/direction logic of the external bus buffer hardware - they cannot use their alternate functions. Each chip select has a predefined fixed 16MB address range (MODM7AE70 Memory Map). The NetBurner API supplied predefined base address variables to make access easier as demonstrated in the EBI examples:

```
extern volatile uint8_t ebi_0_base[];  // Chip select 0
extern volatile uint8_t ebi_1_base[];  // Chip select 1
extern volatile uint8_t ebi_2_base[];  // Chip select 2
extern volatile uint8_t ebi_3_base[];  // Chip select 3
```

  In this way you can do a bus write with code such as (please refer to EBI examples for latest code):

```
const uint8_t  EBITestData8 = 0x5A;                  // Data byte for testing
const uint32_t EBITestAddressOffset = 0xA5A5;        // Address offset from chip select base address

ebi_0_base[EBITestAddressOffset] = EBITestData8;     // Write data to EBI, chip select 0
ebi_0_base[0xA5A5] = 0X5A;                           // Same write with hard coded values

uint32_t EBIdata = ebi_0_base[EBITestAddressOffset]; // Read data from EBI, chip select 0
```

Warning

When enabling the EBI for the MODM7AE70 module, NCS0, NCS2, NCS3 and NRD must all be configured for their bus functionality as chip selects or nRead signals, even if not used by your application - they cannot be assigned their alternate functions. Additionally the "void EnableExtBusBuff(bool enable);" function from bsp.h must be used to enable the external bus buffer hardware.

21.32.2 Enumeration Type Documentation

21.32.2.1 EBI_CS_BusWidth_t

`enum EBI_CS_BusWidth_t`

Data bus width, 8 or 16 bits.

Enumerator

EBI_BUS_WIDTH_8	8-bit bus
EBI_BUS_WIDTH_16	16-bit bus

21.32.2.2 EBI_CS_ByteAccess_t

`enum EBI_CS_ByteAccess_t`

Byte Access. Used only in 16-bit mode to access low or high byte on data bus.

Enumerator

EBI_BYTE_ACCESS_SELECT	Use chip select for 8-bit access.
EBI_BYTE_ACCESS_WRITE	Use write signal for 8-bit access.

21.32.2.3 EBI_CS_NWait_t

`enum EBI_CS_NWait_t`

NWAIT signal mode. Used to extend read/write pulse by the bus device. Normally used with slow flash memory, otherwise disabled.

Any access can be extended by an external device using the NWAIT input signal of the SMC. The SMC_MODE.↩ EXNW_MODE field on the corresponding chip select must be set either to “10” (Frozen mode) or “11” (Ready mode). When SMC_MODE.EXNW_MODE is set to “00” (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the Read and Write modes of the corresponding chip select.

Enumerator

EBI_NWAIT_DISABLED	Disabled.
EBI_NWAIT_FROZEN	Frozen mode.
EBI_NWAIT_READY	Ready mode.

21.32.2.4 EBI_CS_RdMode_t

`enum EBI_CS_RdMode_t`

Read signal selection. NCS for chip select, or NRD for write enable signal.

21.32.2.5 EBI_CS_WrMode_t

`enum EBI_CS_WrMode_t`

Write signal selection. NCS for chip select, or NWE for write enable signal.

21.32.3 Function Documentation

21.32.3.1 ConfigureEBI_CS() [1/2]

```
void ConfigureEBI_CS (
            uint32_t csNum,
            const EBI_CS_cfg_t && cfg )
```

Configure the given Chip Select for the external data bus.

Parameters

csNum	The EBI Chip Select to configure.
cfg	The structure containing the configuration to use.

21.32.3.2 ConfigureEBI_CS() [2/2]

```
void ConfigureEBI_CS (
```

```
            uint32_t csNum,
            const EBI_CS_cfg_t & cfg )
```

Configure the given Chip Select for the external data bus.

Parameters

csNum	The EBI Chip Select to configure.
cfg	The structure containing the configuration to use.

21.32.3.3 ConfigureEBI_CSPin()

```
void ConfigureEBI_CSPin (
            int csNum )
```

Configure the I/O pin for a given Chip Select for the external data bus.

Parameters

csNum	The chip select to configure.

21.32.4 Variable Documentation

21.32.4.1 busWidth

`EBI_CS_cfg_t::busWidth`

Data bus width

21.32.4.2 byteAccess

`EBI_CS_cfg_t::byteAccess`

Byte Access mode, only used for 16-bit bus

21.32.4.3 ncs_rd_pulse

`EBI_CS_cfg_t::ncs_rd_pulse`

NCS pulse length = (256∗ ncs_rd_pulse[6] + ncs_rd_pulse[5:0]) clock cycles

21.32.4.4 ncs_rd_setup

`EBI_CS_cfg_t::ncs_rd_setup`

NCS setup length = (128∗ ncs_rd_setup[5] + ncs_rd_setup[4:0]) clock cycles

21.32.4.5 ncs_wr_pulse

`EBI_CS_cfg_t::ncs_wr_pulse`

NCS pulse length = (256∗ ncs_wr_pulse[6] + ncs_wr_pulse[5:0]) clock cycles

21.32.4.6 ncs_wr_setup

`EBI_CS_cfg_t::ncs_wr_setup`

NCS setup length = (128∗ ncs_wr_setup[5] + ncs_wr_setup[4:0]) clock cycles

21.32.4.7 nrd_cycles

EBI_CS_cfg_t::nrd_cycles
The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as: Read cycle length = (nrd_cycles[8:7]∗256 + nrd_cycles[6:0]) clock cycles

21.32.4.8 nrd_pulse

EBI_CS_cfg_t::nrd_pulse
NRD pulse length = (256∗ nrd_pulse[6] + nrd_pulse[5:0]) clock cycles

21.32.4.9 nrd_setup

EBI_CS_cfg_t::nrd_setup
NRD setup length = (128∗ nrd_setup[5] + nrd_setup[4:0]) clock cycles

21.32.4.10 nWait

EBI_CS_cfg_t::nWait
NWAIT signal mode. Used to extend read/write pulse by the bus device.

21.32.4.11 nwe_cycles

EBI_CS_cfg_t::nwe_cycles
The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as: Write cycle length = (nwe_cycles[8:7]∗256 + nwe_cycles[6:0]) clock cycles

21.32.4.12 nwe_pulse

EBI_CS_cfg_t::nwe_pulse
NWE pulse length = (256∗ nwe_pulse[6] + nwe_pulse[5:0]) clock cycles

21.32.4.13 nwe_setup

EBI_CS_cfg_t::nwe_setup
NWE setup length = (128∗ nwe_setup[5] + nwe_setup[4:0]) clock cycles

21.32.4.14 rdMode

EBI_CS_cfg_t::rdMode
Configures which signal is used to signal a bus read.

21.32.4.15 tdf_cycles

EBI_CS_cfg_t::tdf_cycles
The number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. Note: Valid range => 0-15

21.32.4.16 wrMode

EBI_CS_cfg_t::wrMode
Configures which signal is used to signal a bus write.

21.33 Extra File Descriptors

Functions

- int GetExtraFD (void *extra_data, struct IoExpandStruct *pFuncs)

  Returns a file descriptor for the structure passed as the `IoExpandStruct`. FreeExtraFd( ) will release the fd back to the pool of available fds.
- void * GetExtraData (int fd)

  Returns the `extra` structure value from `IoExpandStruct` associated with the file descriptor.
- void FreeExtraFd (int fd)

  Free a file descriptor and associated resources.
- int GetFreeExtraFDCount ()

  Returns the number of free file descriptors.
- int GetFreeSocketCount (void)

  Returns the number of free sockets.

21.33.1 Detailed Description

The NetBurner environment integrates the RTOS, TCP/IP stack, and other peripherals with a file I/O system based on file descriptors. A file descriptor can be described as a handle to a network socket, serial port, system peripheral, or any other object that can be read or written to. Most of the API functions pass a file descriptor as a parameter to such an object. These function are available if you wish to use the file descritpor system capabilities for your own application purposes. Please refer to the File Descriptor section in the Programmers Guide.

21.33.2 Function Documentation

21.33.2.1 FreeExtraFd()

```
void FreeExtraFd (
            int fd )
```

Free a file descriptor and associated resources.

Parameters

fd	File descriptor

21.33.2.2 GetExtraData()

```
void * GetExtraData (
            int fd )
```

Returns the `extra` structure value from `IoExpandStruct` associated with the file descriptor.

Parameters

fd	File descriptor

Return values

Value	of the extra member of the structure

21.33.2.3 GetExtraFD()

```
int GetExtraFD (
            void * extra_data,
            struct IoExpandStruct * pFuncs )
```

Returns a file descriptor for the structure passed as the `IoExpandStruct`. FreeExtraFd( ) will release the fd back to the pool of available fds.

Parameters

extra_data	The optional extra_data void pointer can be used to pass data into the file descriptor
∗pFuncs	Structure containing callback functions for such things as read, write, close, and the extra_data variable

Return values

Value	greater than 0, representing the file descriptor on success
-1	on failure

21.33.2.4 GetFreeExtraFDCount()

```
int GetFreeExtraFDCount ( )
```

Returns the number of free file descriptors.

Return values

Number	of free extra file descriptors

21.33.2.5 GetFreeSocketCount()

```
int GetFreeSocketCount (
            void  )
```

Returns the number of free sockets.

Return values

Number	of free sockets

21.34 FAT File System

Modules

- FAT File System Seek Codes

Macros

- #define f_delvolume(drvnumber) fm_delvolume(drvnumber)

  Un-mounts a flash card.
- #define f_getfreespace(drivenum, pspace) fm_getfreespace(drivenum, pspace)

  Provides information about the drive space usage.
- #define f_chdir(dirname) fm_chdir(dirname)

Change the directory.

- #define f_mkdir(dirname) fm_mkdir(dirname)

  Makes a new directory.
- #define f_rmdir(dirname) fm_rmdir(dirname)

  Removes a directory.
- #define f_findfirst(filename, find) fm_findfirst(filename, find)

  Find the first file or subdirectory in a specified directory.
- #define f_findnext(find) fm_findnext(find)

  Finds the next file or subdirectory in a specified directory after a previous call to `f_findfirst()` or `f_findnext()`.
- #define f_close(filehandle) fm_close(filehandle)

  Closes an opened file.
- #define f_open(filename, mode) fm_open(filename, mode)

  Opens a file in the file system.
- #define f_read(buf, size, size_st, filehandle) fm_read(buf, size, size_st, filehandle)

  Read data from the current position in a file.
- #define f_write(buf, size, size_st, filehandle) fm_write(buf, size, size_st, filehandle)

  Write data to the file at the current position.
- #define f_seek(filehandle, offset, whence) fm_seek(filehandle, offset, whence)

  Move the stream position of an open file.
- #define f_rewind(filehandle) fm_rewind(filehandle)

  Sets the file position in the open target file to the start of the file.
- #define f_eof(filehandle) fm_eof(filehandle)

  Check whether the current position in the open target file is the end of the file.
- #define f_gettimedate(filename, pctime, pcdate) fm_gettimedate(filename, pctime, pcdate)

  Get the time and date of a file or directory.
- #define f_settimedate(filename, ctime, cdate) fm_settimedate(filename, ctime, cdate)

  Set the time and date of a file or directory.
- #define f_delete(filename) fm_delete(filename)

  Deletes a file.

Functions

- int f_enterFS (void)

  Adds a new task priority to the task list used by the file system.
- void f_releaseFS (void)

  Removes a task priority from the task list used by the file system.

21.34.1 Detailed Description

The FAT32 Embedded Flash File System for Flash Cards, EFFS-FAT

21.34.2 Macro Definition Documentation

21.34.2.1 f_chdir

```
#define f_chdir(
              dirname ) fm_chdir(dirname)
```

Change the directory.

Parameters

dirname	The directory to change to.

Return values

F_NO_ERROR	If able to change to the directory
!F_NO_ERROR	If unable to change to the directory

See also

f_mkdir()

f_rmdir()

21.34.2.2 f_close

```
#define f_close(
              filehandle ) fm_close(filehandle)
```

Closes an opened file.

Parameters

filehandle	A handle to the file to close.

Return values

F_NO_ERROR	If able to close the file
!F_NO_ERROR	If unable to close the file

21.34.2.3 f_delete

```
#define f_delete(
              filename ) fm_delete(filename)
```

Deletes a file.
A read-only or open file cannot be deleted.

Parameters

filename	The name of the file to be deleted.

Return values

F_NO_ERROR	If able to delete the file
!F_NO_ERROR	If unable to delete the file

21.34.2.4 f_delvolume

```
#define f_delvolume(
```

```
            drvnumber ) fm_delvolume(drvnumber)
```

Un-mounts a flash card.

Parameters

drvnumber	The drive to be un-mount.

21.34.2.5 f_eof

```
#define f_eof(
            filehandle ) fm_eof(filehandle)
```

Check whether the current position in the open target file is the end of the file.

Parameters

filehandle	A handle to the file to check.

Return values

0	Nothing at the end of the file
!0	The end of the file, or an error

21.34.2.6 f_findfirst

```
#define f_findfirst(
            filename,
            find ) fm_findfirst(filename, find)
```

Find the first file or subdirectory in a specified directory.
Note: If this is called with "∗.∗" and this is not the root directory, the first entry found will be ".", which is the current directory.

Parameters

filename	The name of the file to find.
find	Where to store the found information.

Return values

F_NO_ERROR	If the file was found
!F_NO_ERROR	If the file wasn't found or there are errors

See also

f_findnext()

21.34.2.7 f_findnext

```
#define f_findnext(
            find ) fm_findnext(find)
```

Finds the next file or subdirectory in a specified directory after a previous call to f_findfirst() or f_findnext().
Note: If this is called with "*.*" and this is not the root directory, the first entry found will be ".", which is the current directory.

Parameters

find	Where to store the found information.

Return values

F_NO_ERROR	If the file was found
!F_NO_ERROR	If the file wasn't found or there are errors

See also

f_findfirst()

21.34.2.8 f_getfreespace

```
#define f_getfreespace(
            drivenum,
            pspace ) fm_getfreespace(drivenum, pspace)
```

Provides information about the drive space usage.

Parameters

drivenum	The drive number to get space usage for.
pspace	The structure to store the usage information in.

Return values

F_NO_ERROR	If able to get the drive space usage
!F_NO_ERROR	If unable to get the drive space usage

21.34.2.9 f_gettimedate

```
#define f_gettimedate(
            filename,
            pctime,
            pcdate ) fm_gettimedate(filename, pctime, pcdate)
```

Get the time and date of a file or directory.

Parameters

	filename	The name of the file or directory to modify.
out	*pctime*	A pointer to store the creation time of the file or directory.
out	*pcdate*	A pointer to store the creation date of the file or directory.

Return values

F_NO_ERROR	If able to get time and date
!F_NO_ERROR	If unable to get time and date

21.34.2.10 f_mkdir

```
#define f_mkdir(
              dirname ) fm_mkdir(dirname)
```

Makes a new directory.

Parameters

dirname	The name to use for the new directory.

Return values

F_NO_ERROR	If able to make the directory
!F_NO_ERROR	If unable to make the directory

See also

f_chdir()

f_rmdir()

21.34.2.11 f_open

```
#define f_open(
              filename,
              mode ) fm_open(filename, mode)
```

Opens a file in the file system.

Parameters

filename	The name of the file to open.
mode	The mode to open the file in. The following options are available: • "r": Reading. The stream is positioned at the beginning of the file. • "r+": Reading and writing. The stream is positioned at the beginning of the file. • "w": Truncate file to 0 length or create for writing. The stream is positioned at the beginning of the file. • "w+": Reading and writing. Create if it doesn't exist, or truncate if it does. The stream is positioned at the beginning of the file. • "a": Append, and create if it doesn't exist. The stream is positioned at the end of the file. • "a+": Reading and appending, and create if it doesn't exist. The stream is positioned at the end of the file.

Return values

*F_FILE**	Pointer to the file if it could be opened
0	If it a file could not be opened

See also

f_close()

21.34.2.12 f_read

```
#define f_read(
              buf,
              size,
              size_st,
              filehandle ) fm_read(buf, size, size_st, filehandle)
```

Read data from the current position in a file.

Parameters

buf	A pointer where the data should be read.
size	The size of the items to be read.
size_st	The number of items to be read.
filehandle	A handle to the file to read to.

Returns

The number of items read.

21.34.2.13 f_rewind

```
#define f_rewind(
              filehandle ) fm_rewind(filehandle)
```

Sets the file position in the open target file to the start of the file.

Parameters

filehandle	A handle to the file operate on.

Return values

F_NO_ERROR	If successful
!F_NO_ERROR	If there was an error

21.34.2.14 f_rmdir

```
#define f_rmdir(
              dirname ) fm_rmdir(dirname)
```

Removes a directory.
Note: The target directory must be empty, otherwise an error code is returned.

Parameters

dirname	The name of the directory to remove.

Return values

F_NO_ERROR	If able to remove the directory
!F_NO_ERROR	If unable to remove the directory

See also

f_chdir()

f_mkdir()

21.34.2.15 f_seek

```
#define f_seek(
              filehandle,
              offset,
              whence ) fm_seek(filehandle, offset, whence)
```

Move the stream position of an open file.

Parameters

filehandle	A handle to the file to move the position of.
offset	The relative byte position according to whence.
whence	Where to calculate the offset from. Can be a value from FAT File System Seek Codes.

Return values

F_NO_ERROR	If the seek operation was successful
!F_NO_ERROR	If there was an error

21.34.2.16 f_settimedate

```
#define f_settimedate(
              filename,
              ctime,
              cdate ) fm_settimedate(filename, ctime, cdate)
```

Set the time and date of a file or directory.

Parameters

filename	The name of the file or directory to modify.
ctime	The creation time of the file or directory.
cdate	The creation date of the file or directory.

Return values

F_NO_ERROR	If able to set time and date

Return values

IF_NO_ERROR	If unable to set time and date

21.34.2.17 f_write

```
#define f_write(
              buf,
              size,
              size_st,
              filehandle ) fm_write(buf, size, size_st, filehandle)
```

Write data to the file at the current position.

Parameters

buf	A pointer to the data to write.
size	The size of the items to write.
size_st	The number of items to write.
filehandle	A handle to the file to write to.

Returns

The number of items written.

21.34.3 Function Documentation

21.34.3.1 f_enterFS()

```
int f_enterFS (
             void  )
```

Adds a new task priority to the task list used by the file system.
This function needs to be called when a new task wants to use the file system, before any other calls to other file system functions. Up to 10 tasks can be assigned to use the file system at any given time.
Note: The task priority level is associated with the task list, not the task itself. This means that a task switched priority levels and still wants to use the file system, it will need to be called again.

Return values

0	If successful
!0	If there were errors

See also

f_releaseFS()

21.34.3.2 f_releaseFS()

```
void f_releaseFS (
             void  )
```

Removes a task priority from the task list used by the file system.
When a task no longer needs to use the file system, or prior to task changing priorities, this should be called. In the case of a task changing priorities, if access to the file system is still desired, another call to `f_enterFS()` should be made.

See also

f_releaseFS()

21.35 FAT File System Seek Codes

Macros

- #define **F_SEEK_SET** FN_SEEK_SET

  Beginning of file.
- #define **F_SEEK_END** FN_SEEK_END

  End of file.
- #define **F_SEEK_CUR** FN_SEEK_CUR

  Current position of the file pointer.

21.35.1 Detailed Description

The codes used when calling `f_seek()`.

21.36 FD Change Type

Enumerations

- enum FDChangeType { eReadSet , eWriteSet , eErrorSet }

  The notifications that a registered FD monitor can receive.

21.36.1 Detailed Description

The notifications that a registered FD monitor can receive.

21.36.2 Enumeration Type Documentation

21.36.2.1 FDChangeType

`enum FDChangeType`
The notifications that a registered FD monitor can receive.

Enumerator

eReadSet	When read data changes from not available to available.
eWriteSet	When an FD changes from not writable to writeable.
eErrorSet	When an FD goes from no error to having an error.

21.37 FTP Client

Modules

- FTP Client Return Codes

Functions

- int FTP_InitializeSession (IPADDR4 server_address, uint16_t port, PCSTR UserName, PCSTR Password, uint32_t time_out)

Initialize a FTP session with a FTP server.

- int FTP_CloseSession (int session)

  Close a FTP session.
- int FTPGetDir (int ftp_Session, char *dir_buf, int nbytes, uint16_t timeout)

  Get the current working directory.
- int FTPSetDir (int ftp_Session, const char *new_dir, uint16_t timeout)

  Set the current working directory.
- int FTPDeleteDir (int ftp_Session, const char *dir_to_delete, uint16_t timeout)

  Delete a directory.
- int FTPMakeDir (int ftp_Session, const char *dir_to_make, uint16_t timeout)

  Create a new directory.
- int FTPUpDir (int ftp_Session, uint16_t timeout)

  Move up one directory level.
- int FTPDeleteFile (int ftp_Session, const char *file_name, uint16_t timeout)

  Delete a file.
- int FTPRenameFile (int ftp_Session, const char *old_file_name, const char *new_file_name, uint16_t timeout)

  Rename a file.
- int FTPSendFile (int ftp_Session, const char *full_file_name, BOOL bBinaryMode, uint16_t timeout)

  Initialize the process to send a file to s FTP server.
- int FTPGetFile (int ftp_Session, const char *full_file_name, BOOL bBinaryMode, uint16_t timeout)

  Initialize the process to get a file from a FTP server.
- int FTPGetList (int ftp_Session, const char *full_dir_name, uint16_t timeout)

  Initialize the process to receive a directory listing from a FTP server.
- int FTPGetFileNames (int ftp_Session, const char *full_dir_name, uint16_t timeout)

  Initialize the process to receive just the file names in a directory from a FTP server.
- int FTPRawCommand (int ftp_Session, const char *cmd, char *cmd_buf, int nbytes, uint16_t timeout)

  Send a FTP command to the FTP server.
- int FTPGetCommandResult (int ftp_Session, char *cmd_buf, int nbytes, uint16_t timeout)

  Returns the result of the last FTP operation.
- int FTPRawStreamCommand (int ftp_Session, const char *cmd, int *pResult, char *cmd_buf, int nbytes, uint16_t timeout)

  Send a command and receive a response over a stream connection.
- void FTPActiveMode (int ftp_Session)

  Set mode to active.
- void FTPPassiveMode (int ftp_Session)

  Set mode to passive.

21.37.1 Detailed Description

The NetBurner FTP Client Group

21.37.2 Function Documentation

21.37.2.1 FTP_CloseSession()

```
int FTP_CloseSession (
            int session )
```

Close a FTP session.

Parameters

session	FTP session number to close

Returns

FTP Client Return Codes

See also

FTP_InitializeSession()

21.37.2.2 FTP_InitializeSession()

```
int FTP_InitializeSession (
            IPADDR4 server_address,
            uint16_t port,
            PCSTR UserName,
            PCSTR Password,
            uint32_t time_out )
```

Initialize a FTP session with a FTP server.

Open a connection to a FTP server and log in with the specified username and password. The session number returned from this call can be used with the file and directory functions.

Parameters

server_address	The IP address of the FTP server
port	The listen port number of the FTP server. Typically 21.
UserName	Log-in username
Password	Log-in password
time_out	Timeout in system Time Ticks

Returns

A value greater than 0 is a FTP session handle, otherwise FTP Client Return Codes

See also

FTP_CloseSession()

21.37.2.3 FTPActiveMode()

```
void FTPActiveMode (
            int ftp_Session )
```

Set mode to active.

Parameters

ftp_Session	FTP session number

21.37.2.4 FTPDeleteDir()

```
int FTPDeleteDir (
            int ftp_Session,
            const char * dir_to_delete,
            uint16_t timeout )
```

Delete a directory.

Parameters

ftp_Session	FTP session number
dir_to_delete	Directory to delete
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.37.2.5 FTPDeleteFile()

```
int FTPDeleteFile (
            int ftp_Session,
            const char * file_name,
            uint16_t timeout )
```

Delete a file.

Parameters

ftp_Session	FTP session number
file_name	Name of file to delete
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.37.2.6 FTPGetCommandResult()

```
int FTPGetCommandResult (
            int ftp_Session,
            char * cmd_buf,
            int nbytes,
            uint16_t timeout )
```

Returns the result of the last FTP operation.
Queries the FTP server for the last FTP operation status.
Must be called after the following functions are used:

- FTPGetList()
- FTPFileNames()
- FTPGetFile()
- FTPSendFile()

Parameters

ftp_Session	FTP session number
cmd_buf	The buffer to hold the ASCII FTP server response. See FTP Server return codes for details.
nbytes	The maximum number of bytes cmd_buf can hold
timeout	Timeout in system Time Ticks

Returns

The FTP server response number on success, otherwise ftpClientReturnCodes

21.37.2.7 FTPGetDir()

```
int FTPGetDir (
            int ftp_Session,
            char * dir_buf,
            int nbytes,
            uint16_t timeout )
```

Get the current working directory.

Parameters

ftp_Session	FTP session number
dir_buf	The buffer to copy the directory information into
nbytes	Maximum number of bytes in the dir_buf
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.37.2.8 FTPGetFile()

```
int FTPGetFile (
            int ftp_Session,
            const char * full_file_name,
            BOOL bBinaryMode,
            uint16_t timeout )
```

Initialize the process to get a file from a FTP server.
This function opens a TCP connection to a FTP server so that a file may be read. it will return a file descriptor that can then be used to read the file data using file I/O functions such as: read() and ReadWithTimeout().
IMPORTANT After reading the file data you must do 2 things:

- The TCP connection must be closed with the close() function.
- FTPGetCommandResult() must be called to check the result of the operation.

Parameters

ftp_Session	FTP session number
full_file_name	Full file name being read, including the path
bBinaryMode	True = read file as binary data, False = read file as ASCII data
timeout	Timeout in system Time Ticks

Returns

TCP file descriptor if greater than 0, otherwise FTP Client Return Codes

```
int fd = FTPGetFile(ftp, "testfile.txt", FALSE, 5 * TICKS_PER_SECOND);    // Send file in ASCII mode

if (fd > 0)
{
    // Code to read all data from the file descriptor goes here
    close(fd);

    int rv = FTPGetCommandResult(ftpSession, resultBuffer, 80, 5 * TICKS_PER*SECOND );

    if ( rv != 226)   // Return code 226 = Closing data connection. Requested file action successful.
       iprintf("Read error result = %d %s\r\n", rv, resultBuffer);
}
else
{
    iprintf("Failed to read file\r\n");
}
```

21.37.2.9 FTPGetFileNames()

```
int FTPGetFileNames (
              int ftp_Session,
              const char * full_dir_name,
              uint16_t timeout )
```

Initialize the process to receive just the file names in a directory from a FTP server.
This function opens a TCP connection to a FTP server so that a list of file names may be read. it will return a file descriptor that can then be used to read the file data using file I/O functions such as: read() and ReadWithTimeout(). IMPORTANT After reading the file data you must do 2 things:

- The TCP connection must be closed with the close() function.
- FTPGetCommandResult() must be called to check the result of the operation.

Parameters

ftp_Session	FTP session number
full_dir_name	Full directory name to read, including the path
timeout	Timeout in system Time Ticks

Returns

TCP file descriptor if greater than 0, otherwise ftpClientReturnCodes

21.37.2.10 FTPGetList()

```
int FTPGetList (
              int ftp_Session,
              const char * full_dir_name,
              uint16_t timeout )
```

Initialize the process to receive a directory listing from a FTP server.
This function opens a TCP connection to a FTP server so that a directory list may be read. it will return a file descriptor that can then be used to read the file data using file I/O functions such as: read() and ReadWithTimeout(). IMPORTANT After reading the file data you must do 2 things:

- The TCP connection must be closed with the close() function.
- FTPGetCommandResult() must be called to check the result of the operation.

Parameters

ftp_Session	FTP session number
full_dir_name	Full directory name to read, including the path
timeout	Timeout in system Time Ticks

Returns

TCP file descriptor if greater than 0, otherwise ftpClientReturnCodes

21.37.2.11 FTPMakeDir()

```
int FTPMakeDir (
            int ftp_Session,
            const char * dir_to_make,
            uint16_t timeout )
```

Create a new directory.

Parameters

ftp_Session	FTP session number
dir_to_make	Directory to create
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.37.2.12 FTPPassiveMode()

```
void FTPPassiveMode (
            int ftp_Session )
```

Set mode to passive.

Parameters

ftp_Session	FTP session number

21.37.2.13 FTPRawCommand()

```
int FTPRawCommand (
            int ftp_Session,
            const char * cmd,
            char * cmd_buf,
            int nbytes,
            uint16_t timeout )
```

Send a FTP command to the FTP server.

Parameters

ftp_Session	FTP session number.
cmd	The command to send. Do not include termination such as \r\n.

Parameters

cmd_buf	The buffer to hold the FTP server response, in ASCII.
nbytes	The maximum number of bytes the cmd_buf can hold.
timeout	Timeout in system Time Ticks.

Returns

The FTP server response code if greater than 0, otherwise ftpClientReturnCodes

21.37.2.14 FTPRawStreamCommand()

```
int FTPRawStreamCommand (
            int ftp_Session,
            const char * cmd,
            int * pResult,
            char * cmd_buf,
            int nbytes,
            uint16_t timeout )
```

Send a command and receive a response over a stream connection.
The FTP server will close the connection after all the data has been read. This function is the basis for such functions as FTPGetList() and FTPGetFiles().
IMPORTANT After reading the file data you must call FTPGetCommandResult() to check the result of the operation.

Parameters

ftp_Session	FTP session number
cmd	Command to send
pResult	The server response code integer value
cmd_buf	The server response code ASCII value (including \r\n)
nbytes	Maximum number of bytes cmd_buf can hold
timeout	Timeout in system Time Ticks

Returns

TCP file descriptor if greater than 0, otherwise ftpClientReturnCodes

21.37.2.15 FTPRenameFile()

```
int FTPRenameFile (
            int ftp_Session,
            const char * old_file_name,
            const char * new_file_name,
            uint16_t timeout )
```

Rename a file.

Parameters

ftp_Session	FTP session number
old_file_name	Name of file to rename
new_file_name	New name of file
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.37.2.16 FTPSendFile()

```
int FTPSendFile (
              int ftp_Session,
              const char * full_file_name,
              BOOL bBinaryMode,
              uint16_t timeout )
```

Initialize the process to send a file to s FTP server.
This function opens a TCP connection to a FTP server so that a file may be sent. it will return a file descriptor that can then be used to send the file data using file I/O write functions such as: write(), writeall(), writestring(), etc.
IMPORTANT After sending the file data you must do 2 things:

- The TCP connection must be closed with the close() function.
- FTPGetCommandResult() must be called to check the result of the operation.

Parameters

ftp_Session	FTP session number
full_file_name	Name of the file being sent
bBinaryMode	True = send file as binary data, False = send file as ASCII data
timeout	Timeout in system Time Ticks

Returns

TCP file descriptor if greater than 0, otherwise FTP Client Return Codes

```
int fd = FTPSendFile(ftp, "testfile.txt", FALSE, 5 * TICKS_PER_SECOND);    // Send file in ASCII mode

if (fd > 0)
{
    writestring(fd, "This is a test file\r\n");
    writestring(fd, "This is line 2 of the test file\r\n");
    writestring(fd, "Last Line\r\n");
    close(fd);

    int rv = FTPGetCommandResult(ftpSession, resultBuffer, 80, 5 * TICKS_PER*SECOND );

    if ( rv != 226)   // Return code 226 = Closing data connection. Requested file action successful.
       iprintf("Write error result = %d %s\r\n", rv, resultBuffer);
}
else
{
    iprintf("Failed to send file\r\n");
}
```

21.37.2.17 FTPSetDir()

```
int FTPSetDir (
              int ftp_Session,
              const char * new_dir,
              uint16_t timeout )
```

Set the current working directory.

Parameters

ftp_Session	FTP session number
new_dir	Directory path to set
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.37.2.18 FTPUpDir()

```
int FTPUpDir (
            int ftp_Session,
            uint16_t timeout )
```

Move up one directory level.

Parameters

ftp_Session	FTP session number
timeout	Timeout in system Time Ticks

Returns

FTP Client Return Codes

21.38 FTP Client Return Codes

Macros

- #define **FTP_OK** (0)

  OK.
- #define **FTP_TIMEOUT** (-1)

  Timeout.
- #define **FTP_PASSWORDERROR** (-2)

  Password error.
- #define **FTP_CONNECTFAIL** (-3)

  Connection failed.
- #define **FTP_COMMANDFAIL** (-4)

  Command failed.
- #define **FTP_COMMANDERROR** (-4)

  Command error.
- #define **FTP_BADSESSION** (-5)

  Bad session.
- #define **FTP_NETWORKERROR** (-6)

  Network error.

21.38.1 Detailed Description

FTP Client Return Codes

21.39 HAL - Hardware Abstraction Layer

Functions

- void HardwareSetup ()

  Initializes the system hardware such as the timer, cache and clock speed.
- void PostConfigHardwareInit ()

  Initializes the system hardware that depend on config variables such as the watchdog.

- void ForceReboot (bool bFromException=false)

  Forces the system hardware to perform a soft reset.
- int HalStorage_Save (uint8_t area, void *pData, int len, int offset=0)

  Save a blob to a specific persistent storage area as defined by the platform. This routine will perform all storage maintenance routines necessary to ensure a valid write.
- int HalStorage_SavePartial (uint8_t area, void *pData, int len, int offset)

  Save a blob to a specific persistent storage area as defined by the platform. This routine requires that any maintenance required for a valid write be explicitly performed prior to being called.
- int HalStorage_Prepare (uint8_t area, int len, int offset=0)

  Prepare a storage area for writing new data. For platforms with direct flash mapping this function is an alias for HalStorage_Erase.
- int HalStorage_Finalize (uint8_t area)

  Finalize a storage area after writing new data. This will perform any final completion or cleanup routines required to persist the previously saved data from HalStorage_SavePartial calls.
- int HalStorage_Erase (uint8_t area, int len=-1, int offset=0)

  Erase all or part of a storage area. Note: due to physical storage granularities, the total area erased may extend beyond the requested area.
- int HalStorage_GetAllocated (uint8_t area)

  Obtain the total size allocated to the given persistent storage area.
- int HalStorage_GetMaxAllocation (uint8_t area)

  Obtain the maximum size that may be allocated to the given persistent storage area. For direct flash mapped platforms, this is aliased to HalStorage_GetAllocated.
- int HalStorage_RemainingSpace (uint8_t area)

  Obtain the number of remaining bytes available to be written in the given persistent storage area.
- int HalStorage_WriteOffset (uint8_t area)

  Get the offset of the next byte that can be written in the given persistent storage area.
- int HalStorage_AddressOffset (uint8_t area, void *pWhere)

  Get the offset within a persistent storage area of an address. A platform may map any address to any offset of its choosing for a given storage area, and my modify its behavior on an area by area basis.
- void FlashErase (void *pWhere, int len)

  Erases the flash memory.
- void FlashProgram (void *pWhere, void *pWhat, int len)

  Program flash memory.
- void FlashProgramAppImage (void *pWhere, void *pWhat, int len)

  Write an application image to flash memory.
- void DisableCache ()

  Disable the instruction and data cache.
- void EnableCache ()

  Enable the instruction and data cache.
- uint32_t spaceleft ()

  Report how much free unallocated space is left in dynamic memory.
- uint16_t HalGetTickFraction (void)

  Returns the fraction of the current system time tick.
- void StdioCheckIntc (void)

  Check STDIO interrupt sources.
- void SysLogCheckIntc (void)

  This is just like the StdioCheckIntc() function, except that the results are displayed via UDP.
- bool HalDeviceCertValid ()

  Determine if the stored certificate is valid.
- uint8_t * HalGetDeviceCert ()

  Get a pointer to the stored certificate.
- uint16_t HalGetDeviceCertLen ()

Get the length of the stored certificate.

- uint8_t * HalGetDeviceKey ()

  Get a pointer to the stored certificate.

- uint16_t HalGetDeviceKeyLen ()

  Get the length of the stored key.

- bool HalSaveNewDeviceCert (const uint8_t *cert, uint16_t certlen, uint8_t format=SSL_FILETYPE_PEM)

  Save a device certificate in persistent storage.

- bool HalSaveNewDeviceKey (const uint8_t *key, uint16_t keylen, uint8_t format=SSL_FILETYPE_PEM)

  Save a device key in persistent storage.

- void HalEraseDeviceCertAndKey ()

  Clear the device certificate and key from persistent storage.

Variables

- void(* watchdog_service_function)(void)

  Watchdog callback service function.

- uint32_t HalTickMaxCount

  Rollover value for the system hardware tick timer.

21.39.1 Detailed Description

The HAL provides a common function name interface to the various NetBurner hardware platforms. Other than the ForceReboot() and spaceleft() functions, these are advanced functions that should only be used by developers experienced with NetBurner hardware, firmware and memory maps.

21.39.2 Function Documentation

21.39.2.1 DisableCache()

```
void DisableCache ( )
```

Disable the instruction and data cache.

21.39.2.2 EnableCache()

```
void EnableCache ( )
```

Enable the instruction and data cache.

21.39.2.3 FlashErase()

```
void FlashErase (
            void * pWhere,
            int len )
```

Erases the flash memory.

Parameters

pWhere	The starting location in flash memory to begin the erasure
len	Specified the number of bytes to erase

21.39.2.4 FlashProgram()

```
void FlashProgram (
            void * pWhere,
            void * pWhat,
            int len )
```

Program flash memory.

Parameters

pWhere	Pointer to the starting location in flash to begin programming
pWhat	Pointer to the content that will be programmed into flash
len	Number of bytes to program

21.39.2.5 FlashProgramAppImage()

```
void FlashProgramAppImage (
            void * pWhere,
            void * pWhat,
            int len )
```

Write an application image to flash memory.
On some platforms writing the application image may require calling ProgramImage even though it may only call FlashErase and FlashProgram. As a result, ProgramImage should always be used when updating the application image.

Parameters

pWhere	Pointer to the starting location in flash to begin the programming
pWhat	Pointer to the content that will be programmed into flash
len	Number of bytes to write

21.39.2.6 ForceReboot()

```
void ForceReboot (
            bool bFromException = false )
```

Forces the system hardware to perform a soft reset.

21.39.2.7 HalDeviceCertValid()

```
bool HalDeviceCertValid ( )
```

Determine if the stored certificate is valid.
This is the only function recommended to be used to determine the validity of the stored certificate. Is is not recommended to use HalGetDeviceCert() nor HalGetDeviceCertLen() to determine validity. For safest use of the cert/key getter functions, use HalDeviceCertValid() prior.

Return values

true	if the certificate is valid false otherwise

21.39.2.8 HalEraseDeviceCertAndKey()

```
void HalEraseDeviceCertAndKey ( )
```

Clear the device certificate and key from persistent storage.
HalEraseDeviceCertAndKey() is recommended instead of HalStorage_Erase() to clear the stored certificate and key.

21.39.2.9 HalGetDeviceCert()

```
uint8_t * HalGetDeviceCert ( )
```

Get a pointer to the stored certificate.
It is recommended to use HalDeviceCertValid() prior to calling this function.

Return values

pointer	to the certificate storage location in memory.

21.39.2.10 HalGetDeviceCertLen()

```
uint16_t HalGetDeviceCertLen ( )
```

Get the length of the stored certificate.
It is recommended to use HalDeviceCertValid() prior to calling this function.

Return values

length	of the stored certificate. 0 or 65535 if no certificates is stored.

21.39.2.11 HalGetDeviceKey()

```
uint8_t * HalGetDeviceKey ( )
```

Get a pointer to the stored certificate.
It is recommended to use HalDeviceCertValid() prior to calling this function.

Return values

pointer	to the certificate storage location in memory.

21.39.2.12 HalGetDeviceKeyLen()

```
uint16_t HalGetDeviceKeyLen ( )
```

Get the length of the stored key.
It is recommended to use HalDeviceCertValid() prior to calling this function.

Return values

length	of the stored key. 0 or 65535 if no key is stored.

21.39.2.13 HalGetTickFraction()

```
uint16_t HalGetTickFraction (
```

```
            void  )
```

Returns the fraction of the current system time tick.
return - The current counter value used to generate tick count

21.39.2.14 HalSaveNewDeviceCert()

```
bool HalSaveNewDeviceCert (
            const uint8_t * cert,
            uint16_t certlen,
            uint8_t format = SSL_FILETYPE_PEM )
```

Save a device certificate in persistent storage.

Parameters

cert	Pointer to the buffer containing the certificate
certlen	length of the certificate
format	filetype (SSL_FILETYPE_PEM, or SSL_FILETYPE_ASN1)

Return values

true	if the certificate was successfully stored
false	otherwise

21.39.2.15 HalSaveNewDeviceKey()

```
bool HalSaveNewDeviceKey (
            const uint8_t * key,
            uint16_t keylen,
            uint8_t format = SSL_FILETYPE_PEM )
```

Save a device key in persistent storage.

Parameters

key	Pointer to the buffer containing the key
keylen	length of the key
format	filetype (SSL_FILETYPE_PEM, or SSL_FILETYPE_ASN1)

Return values

true	if the key was successfully stored
false	otherwise

21.39.2.16 HalStorage_AddressOffset()

```
int HalStorage_AddressOffset (
            uint8_t area,
            void * pWhere )
```

Get the offset within a persistent storage area of an address. A platform may map any address to any offset of its choosing for a given storage area, and my modify its behavior on an area by area basis.

Parameters

area	The storage area being queried
pWhere	An address value to be translated

Returns

Offset of the requested address within the storage area. Returns negative on error.

21.39.2.17 HalStorage_Erase()

```
int HalStorage_Erase (
            uint8_t area,
            int len = -1,
            int offset = 0 )
```

Erase all or part of a storage area. Note: due to physical storage granularities, the total area erased may extend beyond the requested area.

Parameters

area	The storage area to be erased
len	Number of bytes to be erased (a negative length will erase the entire area)
offset	The location within the area that the erasure should begin

Returns

Number of bytes erased. Returns negative on error.

See also

HalEraseDeviceCertAndKey()

21.39.2.18 HalStorage_Finalize()

```
int HalStorage_Finalize (
            uint8_t area )
```

Finalize a storage area after writing new data. This will perform any final completion or cleanup routines required to persist the previously saved data from HalStorage_SavePartial calls.

Parameters

area	The storage area to finalize

Returns

Returns negative on error.

21.39.2.19 HalStorage_GetAllocated()

```
int HalStorage_GetAllocated (
            uint8_t area )
```

Obtain the total size allocated to the given persistent storage area.

Parameters

area	The storage area being queried

Returns

Number of bytes allocated. Returns negative on error.

21.39.2.20 HalStorage_GetMaxAllocation()

```
int HalStorage_GetMaxAllocation (
            uint8_t area )
```

Obtain the maximum size that may be allocated to the given persistent storage area. For direct flash mapped platforms, this is aliased to HalStorage_GetAllocated.

Parameters

area	The storage area being queried

Returns

Number of bytes allocated. Returns negative on error.

21.39.2.21 HalStorage_Prepare()

```
int HalStorage_Prepare (
            uint8_t area,
            int len,
            int offset = 0 )
```

Prepare a storage area for writing new data. For platforms with direct flash mapping this function is an alias for HalStorage_Erase.

Parameters

area	The storage area to be readied
len	Number of bytes to be written by later HalStorage_SavePartial calls
offset	The location within the area that the readying should begin

Returns

Number of bytes readied. Returns negative on error.

21.39.2.22 HalStorage_RemainingSpace()

```
int HalStorage_RemainingSpace (
            uint8_t area )
```

Obtain the number of remaining bytes available to be written in the given persistent storage area.

Parameters

area	The storage area being queried

Returns

Number of bytes available. Returns negative on error.

21.39.2.23 HalStorage_Save()

```
int HalStorage_Save (
            uint8_t area,
            void * pData,
            int len,
            int offset = 0 )
```

Save a blob to a specific persistent storage area as defined by the platform. This routine will perform all storage maintenance routines necessary to ensure a valid write.

Parameters

area	The storage area to be used. A platform is only *required* to support the areas as defined by HalStorage_t, however it may define additional areas as needed
pData	Pointer to the data to be written
len	Number of bytes to be written to the storage area
offset	Where in the storage area the first byte should be written. Default is 0.

Returns

Number of bytes written. Returns negative on error.

21.39.2.24 HalStorage_SavePartial()

```
int HalStorage_SavePartial (
            uint8_t area,
            void * pData,
            int len,
            int offset )
```

Save a blob to a specific persistent storage area as defined by the platform. This routine requires that any maintenance required for a valid write be explicitly performed prior to being called.

Parameters

area	The storage area to be used. A platform is only *required* to support the areas as defined by HalStorage_t, however it may define additional areas as needed.
pData	Pointer to the data to be written
len	Number of bytes to be written to the storage area
offset	Where in the storage area the first byte should be written.

Returns

Number of bytes written. Returns negative on error.

21.39.2.25 HalStorage_WriteOffset()

```
int HalStorage_WriteOffset (
            uint8_t area )
```

Get the offset of the next byte that can be written in the given persistent storage area.

Parameters

area	The storage area being queried

Returns

Offset of the next byte that can be written. Returns negative on error.

21.39.2.26 HardwareSetup()

```
void HardwareSetup ( )
```

Initializes the system hardware such as the timer, cache and clock speed.

21.39.2.27 PostConfigHardwareInit()

```
void PostConfigHardwareInit ( )
```

Initializes the system hardware that depend on config variables such as the watchdog.

21.39.2.28 spaceleft()

```
uint32_t spaceleft ( )
```

Report how much free unallocated space is left in dynamic memory.

Returns

Number of bytes unallocated in dynamic memory

21.39.2.29 StdioCheckIntc()

```
void StdioCheckIntc (
            void  )
```

Check STDIO interrupt sources.

Checks the interrupt sources used in the system along with their associated interrupt level and priority, and also checks for any conflicts in using the same level and priority between different sources. These results are sent to stdout.

See also

SysLogCheckIntc()

21.39.2.30 SysLogCheckIntc()

```
void SysLogCheckIntc (
            void  )
```

This is just like the StdioCheckIntc() function, except that the results are displayed via UDP.

See also

StdioCheckIntc()

21.39.3 Variable Documentation

21.39.3.1 HalTickMaxCount

```
uint32_t HalTickMaxCount  [extern]
```

Rollover value for the system hardware tick timer.

Returns

The max/rollover counter value used to generate the tick count

21.39.3.2 watchdog_service_function

```
void(* watchdog_service_function) (void) (
            void  )  [extern]
```

Watchdog callback service function.
If the following function pointer is set to point at a function of the form:
void YourFunc(void);
Then the system will call this function throughout the AutoUpdate/TCPUpdate process to make sure that the watchdog timer is serviced.

21.40 HTTP and HTML Functions

Classes

- struct HTTP_Request

  HTTP Request Structure.
- class HtmlPageHandler

  Base class for all GET handlers. To handle GET requests for a specific URL in your application, build a GET handler object for that specifif URL. A NULL name will be a catch all for all GET requests.
- class CallBackFunctionPageHandler

  Implements the HtmlPageHandler class as a function pointer callback for GET requests.
- class CallBackFunctionPostHandler

  Implements the HtmlPostHandler class as a function pointer callback for POST requests.
- class HtmlPostVariableListCallback

  Implements the HtmlPostVariableListHandler class as a function pointer callback for HTTP POST submissions.

Typedefs

- typedef int() http_gethandlerfunc(int sock, HTTP_Request &pd)

  Implements the HtmlPageHandler class as a function pointer callback for GET requests.
- typedef int() http_posthandler(int sock, HTTP_Request &httpReqInfo)

  Type definition of the HtmlPostHandler callback for POST requests.

Enumerations

- enum HTTP_RequestTypes { tUnknown , tGet , tPost , tHead }

  HTTP request types for HTTP page handler callback functions.
- enum HTTP_ACCESS

  HTTP page access return values.

Functions

- int SendEmailResponse (int sock, const char *name, const char *attachment)

  Send an email with HTML formatting.
- int SendFullResponse (char const *name, int fd)

Send a file with the proper HTTP header, file specified by file name.

- int SendFullResponse (HTML_FILE_RECORD *fr, int fd, const char *pUrl)

  Send a file with the proper HTTP header, file specified by HTML_FILE_RECORD.
- int SendHeaderResponse (char const *name, int fd)

  Send a HTTP header response for the specified file type.
- int SendHeaderResponse (HTML_FILE_RECORD *fr, int fd)

  Send a HTTP header response for the specified HTML_FILE_RECORD type.
- int32_t SendFileFragment (char const *name, int32_t fd, PCSTR url=NULL)

  Send a file fragment without a header.
- HTML_FILE_RECORD * GetRecordFromName (char const *name)

  Returns a pointer to a HTML_FILE_RECORD for the specified file name.
- CONFIG_RENDER_OBJ ConfigRenderFunc (int mode, const char *pobj, int len=20, const char *extra=0)

  Render a configuration object as HTML for the specified configuration object.
- CONFIG_RENDER_OBJ ConfigRenderFunc (int mode, config_leaf &cl, int len=20, const char *extra=0)

  Render a configuration object as HTML for the specified configuration leaf object.
- void WriteHtmlVariable (int fd, CONFIG_RENDER_OBJ co)

  Send a CONFIG_RENDER_OBJ to the client.
- HTTP_Request * **GetActiveHttpRequest** ()

  Get thc curent active running http request. Only valid fromwithing http get or post handeling. Also valid during page fill in operations.
- HTTP_ACCESS CheckHttpAccess (int sock, int access_level, HTTP_Request &Req)

  All HTTP requests go though this function.
- void StartHttp (uint16_t port, bool RunConfigMirror)

  Start the HTTP web server. Further documentation in the Initialization section Initialization - System Initialization Functions.
- void StopHttp ()

  Stop the HTTP web server.
- void SendHTMLHeader (int sock)

  Send a HTML response header.
- void SendHTMLHeaderWCookie (int sock, char *cookie)

  Send a HTML response header and cookie.
- void SendTextHeader (int sock)

  Send a HTML plain text header.
- void SendGifHeader (int sock)

  Send a HTML GIF header.
- void EmptyResponse (int sock)

  Send an empty response back.
- void NoContentResponse (int sock)

  Send a no content response back.
- void RedirectResponse (int sock, PCSTR new_page)

  Redirect a HTTP request to a different page.
- void NotFoundResponse (int sock, PCSTR new_page)

  Send a page not found response.
- void ForbiddenResponse (int sock, PCSTR new_page)

  Send a page is forbidden response.
- void NotAvailableResponse (int sock, PCSTR new_page)

  Send a response indicating that the requested resource is not available at this time.
- void BadRequestResponse (int sock, PCSTR url, PCSTR data)

  Send a response indicating that the client request itself is faulty in some manner.
- int writeallsafestring (int fd, PCSTR str)

  Send a string and escape all special characters.
- int writesafestring (int fd, PCSTR str, size_t strLength)

Send a string with a specified length and escape all special characters.

- int httpstricmp (PCSTR str1, PCSTR strIsUpper2)

  Special string compare. Returns 1 if the strings match until one string ends with a null (0).

- void StartHttps (uint16_t ssl_port, uint16_t http_port)

  Start the HTTPS secure web server.

21.40.1 Detailed Description

API to process and delivery files from the web server to clients. In particular, processing of files in a project's html folder.
For the majority of applications these functions will not be necessary; the web server system will automatically handle both static and dynamic content. This API is available if more advances processing is required, such as sending custom response headers or very large amounts of data.
In addition to the functions to start a web server, StartHttp(), this module provides functions used to create a custom web interface. Please refer to the Programmers Guide for additional details on web interface programming and dynamic content.

See also

StartHttps()

21.40.2 Typedef Documentation

21.40.2.1 http_gethandlerfunc

```
typedef int() http_gethandlerfunc(int sock, HTTP_Request &pd)
```

Implements the HtmlPageHandler class as a function pointer callback for GET requests.

Return values

0	Not handled, no changes are made
1	Handled
2	TCP socket will be kept. Socket will get closed by some other method. For example, a large file could be sent from a different task. Any return value other than 2 means the page was processed normally.

21.40.2.2 http_posthandler

```
typedef int() http_posthandler(int sock, HTTP_Request &httpReqInfo)
```

Type definition of the HtmlPostHandler callback for POST requests.
This should only be used if the specific variable callback implementions will not work in your application. This is mostly an internal use function.

Parameters

sock	The socket the request came in on.
httpReqInfo	Information about the request HTTP_Request.

Return values

0	Not handled, no changes are made
1	Handled
2	TCP socket will be kept. Socket will get closed by some other method. For example, a large file could be sent from a different task. Any return value other than 2 means the page was processed normally.

See also

HtmlPostVariableListCallback

HtmlPostVariableListHandler

21.40.3 Enumeration Type Documentation

21.40.3.1 HTTP_RequestTypes

```
enum HTTP_RequestTypes
```

HTTP request types for HTTP page handler callback functions.

Enumerator

tUnknown	The type of request is not yet known.
tGet	GET request.
tPost	POST request.
tHead	Header request, does not include content.

21.40.4 Function Documentation

21.40.4.1 BadRequestResponse()

```
void BadRequestResponse (
            int sock,
            PCSTR url,
            PCSTR data )
```

Send a response indicating that the client request itself is faulty in some manner.

Parameters

sock	Socket to send response
url	The URL of the page to display
data	An explanation or segment of data to display indicating the fault

See also

RedirectResponse(), NotFoundResponse(), ForbiddenResponse()

NotAvailableResponse()

21.40.4.2 CheckHttpAccess()

```
HTTP_ACCESS CheckHttpAccess (
            int sock,
            int access_level,
            HTTP_Request & Req )
```

All HTTP requests go though this function.
By default the library provides a weak reference implimentaion that approves all transactions. To provide your own just add an implmentaion of this function to your code.

Parameters

sock	The socket the request came in on.
access_level	The user defined access level for this request. See HtmlPageHandler
Req	The HTTP request where one can extract the URL , access type, password, etc

Returns

The desired response.

21.40.4.3 ConfigRenderFunc() [1/2]

```
CONFIG_RENDER_OBJ ConfigRenderFunc (
            int mode,
            config_leaf & cl,
            int len = 20,
            const char * extra = 0 )
```

Render a configuration object as HTML for the specified configuration leaf object.

Parameters

mode	CONFIG_RENDER_OBJ eMode value
cl	Reference to the config_leaf object to render
len	Optional length parameter for text input elements
extra	Pointer to any additional information to be added to the HTML input element. Will be added to the end of the <input> tag.

Returns

A CONFIG_RENDER_OBJ that can be sent to the client using WriteHtmlVariable()

21.40.4.4 ConfigRenderFunc() [2/2]

```
CONFIG_RENDER_OBJ ConfigRenderFunc (
            int mode,
            const char * pobj,
            int len = 20,
            const char * extra = 0 )
```

Render a configuration object as HTML for the specified configuration object.

Parameters

mode	CONFIG_RENDER_OBJ eMode value
pobj	Pointer to a configuration object. Can be specified as the JSON configuration path. For example, "Config/Sys/NetIf/Ethernet0/StaticAddr".
len	Optional length parameter for text input elements
extra	Pointer to any additional information to be added to the HTML input element. Will be added to the end of the <input> tag.

Returns

A CONFIG_RENDER_OBJ that can be sent to the client using WriteHtmlVariable()

21.40.4.5 EmptyResponse()

```
void EmptyResponse (
            int sock )
```

Send an empty response back.

Parameters

sock	Socket to send response

See also

ForbiddenResponse(), ForbiddenResponse(), SendHTMLHeaderWCookie()
NotAvailableResponse(), BadRequestResponse(), RedirectResponse()

21.40.4.6 ForbiddenResponse()

```
void ForbiddenResponse (
            int sock,
            PCSTR new_page )
```

Send a page is forbidden response.

Parameters

sock	Socket to send response
new_page	The URL of the page to display

See also

RedirectResponse(), NotFoundResponse()
NotAvailableResponse(), BadRequestResponse()

21.40.4.7 GetRecordFromName()

```
HTML_FILE_RECORD * GetRecordFromName (
            char const * name )
```

Returns a pointer to a HTML_FILE_RECORD for the specified file name.

Parameters

name	Name of file in htmldata.cpp to search for

Return values

A	pointer to a HTML_FILE_RECORD if found
NULL	if file name not found

21.40.4.8 httpstricmp()

```
int httpstricmp (
            PCSTR str1,
            PCSTR strIsUpper2 )
```

Special string compare. Returns 1 if the strings match until one string ends with a null (0).
strIsUpper2 must be specified in upper case
Used to match URL with stored file prefixes.

Parameters

str1	The string to search
strIsUpper2	The string to search for, must be specified in upper case

Return values

0	if the prefixes do not match
1	if the prefixes match

Example: httpstricmp(str1, "LED.HTML");

str1	Return Value
LED	1
led.HTML	1
led.html	1
LED.HTM?	0

21.40.4.9 NoContentResponse()

```
void NoContentResponse (
            int sock )
```

Send a no content response back.

Parameters

sock	Socket to send response

See also

ForbiddenResponse(), ForbiddenResponse(), SendHTMLHeaderWCookie()
NotAvailableResponse(), BadRequestResponse(), RedirectResponse()

21.40.4.10 NotAvailableResponse()

```
void NotAvailableResponse (
            int sock,
            PCSTR new_page )
```

Send a response indicating that the requested resource is not available at this time.

Parameters

sock	Socket to send response
new_page	The URL of the page to display

See also

RedirectResponse(), NotFoundResponse(), ForbiddenResponse()

BadRequestResponse()

21.40.4.11 NotFoundResponse()

```
void NotFoundResponse (
            int sock,
            PCSTR new_page )
```

Send a page not found response.

Parameters

sock	Socket to send response
new_page	The URL of the page to display

See also

ForbiddenResponse(), RedirectResponse()

NotAvailableResponse(), BadRequestResponse()

21.40.4.12 RedirectResponse()

```
void RedirectResponse (
            int sock,
            PCSTR new_page )
```

Redirect a HTTP request to a different page.

Parameters

sock	Socket to send response
new_page	The URL of the new page

See also

ForbiddenResponse(), ForbiddenResponse(), SendHTMLHeaderWCookie()

NotAvailableResponse(), BadRequestResponse(), EmptyResponse()

21.40.4.13 SendEmailResponse()

```
int SendEmailResponse (
            int sock,
            const char * name,
            const char * attachment )
```

Send an email with HTML formatting.

Parameters

sock	Network socket in which to send the data
name	Name of file from project's html folder, this is the email body
attachment	

Return values

1	Success, the file specified the name parameter was found
0	File does not exist

21.40.4.14 SendFileFragment()

```
int32_t SendFileFragment (
              char const * name,
              int32_t fd,
              PCSTR url = NULL )
```

Send a file fragment without a header.
Search through the files stored in the system by comphtml in the project's html folder for the specified file name. If found, the file content is sent as a file fragment. A SendHeaderRespose function must be sent prior to sending file fragments.
This function is useful when building HTML responses with large amounts of data. If the file has embedded dynamic HTML function tags and/or variable tags, such a CPPCALL and VARIABLE, they will be processed as the content is delivered.

Parameters

name	Name of the file
fd	File descriptor used to send the file fragment
url	Optional URL string in fragment header

Return values

1	if the file was found
0	otherwise

21.40.4.15 SendFullResponse() [1/2]

```
int SendFullResponse (
              char const * name,
              int fd )
```

Send a file with the proper HTTP header, file specified by file name.
This function searches through the files stored in the system by comphtml in the project's html folder for the specified file name. If found, it sends the proper HTTP header and file to the client. If the file has embedded dynamic HTML function tags and/or variable tags, such a CPPCALL and VARIABLE, they will be processed as the content is delivered.

Parameters

name	Name of the file
fd	File descriptor used to send the file

Return values

1	Succes, file was found
0	File does not exist

See also

SendFullResponse(HTML_FILE_RECORD *fr, int fd, const char *pUrl)

21.40.4.16 SendFullResponse() [2/2]

```
int SendFullResponse (
            HTML_FILE_RECORD * fr,
            int fd,
            const char * pUrl )
```

Send a file with the proper HTTP header, file specified by HTML_FILE_RECORD.
While the SendFullResponse(char const *name, int fd) function will resolve the file name, this lower level function requires knowledge of the file pointer to the HTML_FILE_RECORD in htmldata.cpp.

Parameters

fr	Pointer to the HTML_FILE_RECORD in htmldata.cpp
fd	File descriptor used to send the file
pUrl	Pointer to a URL string used by response functions such a CPPCALL and VARIABLE callbacks. In default web server processing the URL is automatically set to what the client requested. When processing web requests manually you must specify the URL string. If you are not using dynamic content that relies on the URL, this can be set to NULL.

Return values

1	if the file was found
0	otherwise

See also

SendFullResponse(char const *name, int fd)

21.40.4.17 SendGifHeader()

```
void SendGifHeader (
            int sock )
```

Send a HTML GIF header.
Sends a GIF response header. Can be useful when dynamically creating your own GIF images.

Parameters

sock	Socket to send response

See also

SendHTMLHeader(), SendTextHeader(), SendHTMLHeaderWCookie()

21.40.4.18 SendHeaderResponse() [1/2]

```
int SendHeaderResponse (
            char const * name,
            int fd )
```

Send a HTTP header response for the specified file type.
This function searches through the files stored in the system by comphtml in the project's html folder for the specified file name. If found, it sends the proper HTTP header only to the client. Unlike a full response, only the header is sent.

Parameters

name	Name of the file
fd	File descriptor used to send the file

Return values

1	Succes, file was found
0	File does not exist

See also

SendHeaderResponse(HTML_FILE_RECORD *fr, int fd)

21.40.4.19 SendHeaderResponse() [2/2]

```
int SendHeaderResponse (
            HTML_FILE_RECORD * fr,
            int fd )
```

Send a HTTP header response for the specified HTML_FILE_RECORD type.
While the SendFullResponse(HTML_FILE_RECORD *fr, int fd, const char *pUrl); function will resolve the file name, this lower level function requires knowledge of the file pointer to the HTML_FILE_RECORD array in htmldata.cpp.

Parameters

fr	Pointer to the HTML_FILE_RECORD in htmldata.cpp
fd	File descriptor used to send the file

Return values

1	Succes, file was found
0	File does not exist

See also

SendHeaderResponse(char const *name, int fd)

21.40.4.20 SendHTMLHeader()

```
void SendHTMLHeader (
            int sock )
```

Send a HTML response header.
Used to build your own HTML response

Parameters

sock	Socket to send response

See also

SendHTMLHeaderWCookie(), SendTextHeader(), SendGifHeader()

21.40.4.21 SendHTMLHeaderWCookie()

```
void SendHTMLHeaderWCookie (
            int sock,
            char * cookie )
```

Send a HTML response header and cookie.
Sends a header as well as a cookie to be stored by the client browser.

Parameters

sock	Socket to send response
cookie	Pointer to the cookie

See also

SendHTMLHeader(), SendGifHeader(), SendTextHeader()

21.40.4.22 SendTextHeader()

```
void SendTextHeader (
            int sock )
```

Send a HTML plain text header.
Use as the first part of building your own plain text response.

Parameters

sock	Socket to send response

See also

SendHTMLHeader(), SendGifHeader(), SendHTMLHeaderWCookie()

21.40.4.23 StartHttp()

```
void StartHttp (
            uint16_t port,
            bool RunConfigMirror )
```

Start the HTTP web server. Further documentation in the Initialization section Initialization - System Initialization Functions.
If no parameters are specified it will listen on port 80, and enable the configuration mirror feature so an application can create its own custom configuration web page.

Parameters

port	Listen port, defaults to port 80
RunConfigMirror	Enable configuration mirror, default is true

See also

StopHttp(), StartHttps()

21.40.4.24 StartHttps()

```
void StartHttps (
            uint16_t ssl_port,
            uint16_t http_port )
```

Start the HTTPS secure web server.
If no parameters are specified, the web server will listen on ports 443 and 80, and enable the configuration mirror feature so an application can create its own custom configuration web page. If no user certificate/key is installed, the default system key will be used.
Default parameter values defined in init.h

Parameters

ssl_port	Secure listen port, defaults to port 443
http_port	Non-secure listen port, defaults to port 80

See also

StartHttp(), StopHttp()

21.40.4.25 StopHttp()

```
void StopHttp ( )
```

Stop the HTTP web server.

See also

StartHttp()

21.40.4.26 writeallsafestring()

```
int writeallsafestring (
            int fd,
            PCSTR str )
```

Send a string and escape all special characters.
When sending a HTML text response, certain characters (e.g. '<') are interpreted by the browser as formatting. This function properly escapes the text so it will appear as intended.

Parameters

fd	File descriptor used to send response
str	String to send

Returns

The number of characters sent

21.40.4.27 WriteHtmlVariable()

```
void WriteHtmlVariable (
            int fd,
            CONFIG_RENDER_OBJ co )
```

Send a CONFIG_RENDER_OBJ to the client.
The CONFIG_RENDER_OBJ must have been previously created with a ConfigRenderFunc() function.

Parameters

fd	File descriptor to send data to
co	CONFIG_RENDER_OBJ to send

21.40.4.28 writesafestring()

```
int writesafestring (
            int fd,
            PCSTR str,
            size_t strLength )
```

Send a string with a specified length and escape all special characters.
When sending a HTML text response, certain characters (e.g. '<') are interpreted by the browser as formatting. This function properly escapes the text so it will appear as intended.

Parameters

fd	File descriptor used to send response
str	String to send
strLength	number of characters from str to send

Returns

The number of characters sent

21.41 Helper Macros

Macros

- #define **CAN_EXTENDED_ID_BIT** (0x20000000)

  The single bit used by this API to indicate an extended ID.
- #define **ExtToNbId**(id) (id | CAN_EXTENDED_ID_BIT)

  Convert an ID to an extended ID.
- #define **NormToNbId**(id) (id & 0x7ff)

  Convert a normal ID in the range of 0 to 2048 to the range of 0 to 1024.
- #define IsNBIdExt(id) ((id & CAN_EXTENDED_ID_BIT)!=0)

  Check for an extened MCAN ID.
- #define **NbToExtId**(id) (id & 0x1FFFFFFF)

  Remove the API extended flage from the ID.
- #define NbToNormId(id) (id & 0x7FF)

  Shift a Normal ID so it has a value from 0 to 1023.

21.41.1 Detailed Description

21.41.2 Macro Definition Documentation

21.41.2.1 IsNBIdExt

```
#define IsNBIdExt(
              id ) ( (id & CAN_EXTENDED_ID_BIT)!=0)
```

Check for an extened MCAN ID.
Returns non zero if the ID is an extended ID

21.41.2.2 NbToNormId

```
#define NbToNormId(
              id ) (id & 0x7FF)
```

Shift a Normal ID so it has a value from 0 to 1023.
Some CAN systems will treat normal ID's as an integer from 0 to 2048. Other systems may treat normal ID's as 28 bit values where the bottom 17 bits are zero. This macro will convert the NetBurner normal ID format to the 0 to 2048 format.

21.42 High Resolution Delay Timer

Classes

- class DelayObject

  Microsecond Delay Class.

21.42.1 Detailed Description

The OSTimeDly() system function has a 20ms tick count, but if you need more precision timing The High Resolution Delay timer provides delays in microseconds.
When a DelayObject is created, it allocates an available hardware system timer. Once the delay time has expired, and when the object goes out of scope, the hardware timer is freed.
Example Usage:

```
static DelayObject myHiResDelay;    // Create the object
myHiDrsDelay.DelayUsec(100);        // Delay 100 us
myHiDrsDelay.DelayUsec(200);        // Delay 200 us
myHiDrsDelay.DelayUsec(300);        // Delay 300 us
```

See also

Interval Timer

Stopwatch Timer

OSTimeDly()

21.43 I/O Control Command Flags

Macros

- #define **IOCTL_SET** (0x4000)

  Set an option.
- #define **IOCTL_CLR** (0x2000)

  Clear an option.

21.43.1 Detailed Description

The command flags to use in contuction with ioctl() and I/O Control Options

21.44 I/O Control Options

Macros

- #define **IOCTL_TX_CHANGE_CRLF** (1)

  When set, transmitted char \n gets converted to \r\n
- #define **IOCTL_RX_CHANGE_CRLF** (2)

  When set, received \r\n get turned into \n
- #define **IOCTL_RX_PROCESS_EDITS** (4)

  When set, process backspace and do simple line editing.
- #define **IOCTL_RX_ECHO** (8)

  When set, echo chars received to tx.
- #define **IOCTL_TX_NO_BLOCK** (32)

  When set, stdout and stderr will drop output instead of blocking.
- #define **IOCTL_ALL_OPTIONS** (15)

  When set, turns on all options.

21.44.1 Detailed Description

The legal options for use with ioctl() and in conjunction with I/O Control Command Flags

21.45 I2C

Modules

- MCF5441x I2C
- SAME70 I2C

21.45.1 Detailed Description

I2C API for NetBurner Platforms

21.46 IOSYS - I/O System

Modules

- FD Change Type
- I/O Control Command Flags
- I/O Control Options

Typedefs

- typedef void FDCallBack(int fd, FDChangeType change, void *pData)

  Define the function signature for file descriptor notification callbacks.

Functions

- int close (int fd)

  Close the specified file descriptor and free the associated resources.
- int read (int fd, char *buf, int nbytes)

  Read data from a file descriptor (fd). This function will block forever until at least one byte is available to be read (as opposed to the ReadWithTimeout() function which reads data from a file descriptor with a specified time-out value). This function can be used to read from stdio, TCP sockets, or Serial ports.
- int peek (int fd, char *c)

Peek at the data for the specified file descriptor (fd). Will block forever until at least one byte is available to be read. The byte returned is not removed from the fd buffer, it will be the first byte of data returned on any subsequent read operation.

- int write (int fd, const char *buf, int nbytes)

  Write data to the stream associated with a file descriptor (fd). Can be used to write data to stdio, a TCP socket, or a Serial port.
- int writestring (int fd, const char *str)

  Write a null terminated ascii string to the stream associated with a file descriptor (fd). Can be used to write data to stdio, a TCP socket, or a Serial port.
- int writeall (int fd, const char *buf, int nbytes=0)

  Write the specified number of bytes to a file descriptor. Will block until all bytes are sent, or a file descriptor error occurs (such as a TCP socket error). Can be used to write data to stdio, a TCP socket, or a Serial port.
- int ReadWithTimeout (int fd, char *buf, int nbytes, unsigned long timeout)

  Read data from a file descriptor(fd), or return if at least one byte is not read within the specified timeout. Can be used instead of the read() function, which will block forever until at least one byte is read. File descriptors include such things as stdio, TCP sockets, TLS sockets, and Serial ports.
- int ReadWithTickTimeout (int fd, char *buf, int nbytes, TickTimeout &timeout)

  Same as ReadWithTimeout(), except the timeout value parameter is of type TickTimeout instead of an unsigned long. The TickTimeout object is more robust in terms of sequential timeout calls and rollover protection.
- int ReadAllWithTimeout (int fd, char *buf, int nbytes, unsigned long timeout)

  Attempt to read the specified number of bytes from a file descriptor, or return with the number of bytes read if the timeout value has expired.
- int ReadAllWithTickTimeout (int fd, char *buf, int nbytes, TickTimeout &timeout)

  Same as ReadWithTimeout(), except the timeout value parameter is of type TickTimeout instead of an unsigned long. The TickTimeout object is more robust in terms of sequential timeout calls and rollover protection.
- int readall (int fd, char *buf, int nbytes)

  Read the specified number of bytes from a file descriptor (fd). This function will block until either the requested number of bytes have been read, or a file descriptor error occurs. It can be used to read from stdio, TCP sockets, or Serial ports.
- int PeekWithTimeout (int fd, char *c, unsigned long timeout)

  This function peeks at data from a file descriptor (fd), with a specified timeout value (as opposed to the peek function which will block forever until at least one byte is available to be read). This function can be used to peek from stdio, TCP sockets, or Serial ports.
- int dataavail (int fd)

  Check the specified file descriptor to detmeine if data is available to be read.
- int writeavail (int fd)

  Check the specified file descriptor to determine data can be written.
- int haserror (int fd)

  Check if a file descriptor has an error.
- int charavail ()

  Checks to see if data is available for read on stdin. By default, stdin is the boot/debug serial port.
- void RegisterFDCallBack (int fd, FDCallBack *fp, void *pData)

  Register a callback function to receive notification when an FD state changes. Register a NULL fp to remove the notification.
- void FD_ZERO (fd_set *pfds)

  Clear (set to 0) a fd_set (file descriptor set) so no file descriptors (fds) are selected.
- void FD_CLR (int fd, fd_set *pfds)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function clears or removes a specific file descriptor in an fd_set.
- void FD_SET (int fd, fd_set *pfds)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function sets or adds a specific file descriptor to an fd_set.
- int FD_ISSET (int fd, fd_set *pfds)

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function checks whether or not a particular fd is set in the specified fd_set.

- int FD_OVERLAP (const fd_set *f1, const fd_set *f2)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function determines whether there exists a common file descriptor between the two sets.
- void FD_COPY (const fd_set *from, fd_set *to)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function copies one file descriptor set to another.
- void FD_SETFROMSET (const fd_set *from, fd_set *to)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function adds the file descriptors from one set to another.
- void FD_CLRFROMSET (const fd_set *from, fd_set *to)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function remove the file descriptors in one set from another.
- int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, unsigned long timeout)

  Wait for events to occur on one or more I/O resources associated with a set of file descriptors (fds), such as data available to be read, a resource is available to write data, or an error has occurred.
- int ZeroWaitSelect (int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds)

  Returns whether events have occurred on one or more I/O resources associated with a set of file descriptors (fds), and returns immediately.
- int ioctl (int fd, int cmd)

  This function controls the selection of input/output control options for stdio: stdin = 0, stdout = 1 and stderr = 2.
- int ReplaceStdio (int stdio_fd, int new_fd)

  Maps stdio to any file descriptor (fd).
- int CurrentStdioFD (int stdio_fd)

  Returns the current file descriptor mapped to the stdio file descriptor.
- void **IrqStdio** ()

  Open the system default serial port in interrupt mode using the system default baud rate, and assign this serial port to stdin, stdout and stderr.

21.46.1 Detailed Description

I/O system functions use file descriptors for communication and status of network interfaces, serial interfaces, and any custom file descriptors created by an application.

21.46.2 Typedef Documentation

21.46.2.1 FDCallBack

```
typedef void FDCallBack(int fd, FDChangeType change, void *pData)
```

Define the function signature for file descriptor notification callbacks.

Parameters

fd	The file descriptor number
change	The reason for the call back, FDChangeType
**pData*	A void pointer passed into the register function. Can be used to reference objects or other data

See also

FDChangeType

RegisterFDCallBack

21.46.3 Function Documentation

21.46.3.1 charavail()

```
int charavail ( )
```

Checks to see if data is available for read on stdin. By default, stdin is the boot/debug serial port.

Return values

1	Data available
0	No data available

See also

dataavail()

writeavail()

read()

21.46.3.2 close()

```
int close (
            int fd )
```

Close the specified file descriptor and free the associated resources.

Parameters

fd	The file descriptor number

Returns

0 on success, or a resource specific error on failure.

See also

peek()

PeekWithTimeout()

read()

ReadWithTimeout()

write()

21.46.3.3 CurrentStdioFD()

```
int CurrentStdioFD (
            int stdio_fd )
```

Returns the current file descriptor mapped to the stdio file descriptor.

Parameters

stdio_fd	The stdio file descriptor to check. Options are stdin (0), stdout (1), and stderr (2).

Returns

The current file descriptor mapped to stdio_fd

21.46.3.4 dataavail()

```
int dataavail (
            int fd )
```

Check the specified file descriptor to detmeine if data is available to be read.

Parameters

fd	The file descriptor number

Return values

1	Data available
0	No data available

See also

charavail()

peek()

read()

ReadWithTimeout()

21.46.3.5 FD_CLR()

```
void FD_CLR (
            int fd,
            fd_set * pfds )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function clears or removes a specific file descriptor in an fd_set.

Parameters

fd	The file descriptor number
**pfds*	Pointer to the fd_set to modify

See also

FD_ZERO()

FD_SET()

FD_ISSET()

select()

21.46.3.6 FD_CLRFROMSET()

```
void FD_CLRFROMSET (
            const fd_set * from,
            fd_set * to )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function remove the file descriptors in one set from another.

Parameters

∗from	Pointer to the fd_set defining which file descriptors to remove.
∗to	Pointer to the fd_set containing file descritpors to remove.

See also

FD_ZERO()

FD_CLR()

FD_ISSET()

select()

21.46.3.7 FD_COPY()

```
void FD_COPY (
            const fd_set * from,
            fd_set * to )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function copies one file descriptor set to another.

Parameters

∗from	Pointer to the file descriptor set being copied from.
∗to	Pointer to the fd_set to copy to.

See also

FD_ZERO()

FD_CLR()

FD_ISSET()

select()

21.46.3.8 FD_ISSET()

```
int FD_ISSET (
            int fd,
            fd_set * pfds )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function checks whether or not a particular fd is set in the specified fd_set.
For example:

```
if ( FD_ISSET( fdListen, &readFds ) )
{
   // do processing for fdListen
}
```

Parameters

fd	The file descriptor number
∗pfds	A pointer to the fd_set to test

Return values

0	File descriptor is not set in the file descriptor set
>0	File descriptor is set in the file descriptor set

See also

FD_ZERO()

FD_CLR()

FD_SET()

FD_OVERLAP()

select()

21.46.3.9 FD_OVERLAP()

```
int FD_OVERLAP (
            const fd_set * f1,
            const fd_set * f2 )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function determines whether there exists a common file descriptor between the two sets.

Parameters

∗f1	Pointer to the first file descriptor set.
∗f2	Pointer to the second file descriptor set.

See also

FD_ZERO()

FD_CLR()

FD_ISSET()

select()

21.46.3.10 FD_SET()

```
void FD_SET (
            int fd,
            fd_set * pfds )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function sets or adds a specific file descriptor to an fd_set.

Parameters

fd	The file descriptor number
∗pfds	Pointer to the fd_set to modify

See also

FD_ZERO()

FD_CLR()

FD_ISSET()

select()

21.46.3.11 FD_SETFROMSET()

```
void FD_SETFROMSET (
            const fd_set * from,
            fd_set * to )
```

A fd_set (file descriptor set) holds a set of file descriptors (fds). This function adds the file descriptors from one set to another.

Parameters

∗from	Pointer to the file descriptor set being copied from.
∗to	Pointer to the fd_set to add to.

See also

FD_ZERO()

FD_CLR()

FD_ISSET()

select()

21.46.3.12 FD_ZERO()

```
void FD_ZERO (
            fd_set * pfds )
```

Clear (set to 0) a fd_set (file descriptor set) so no file descriptors (fds) are selected.

Parameters

∗pfds	Pointer to the file descriptor set

See also

FD_CLR()

FD_SET()

FD_ISSET()

select()

21.46.3.13 haserror()

```
int haserror (
            int fd )
```

Check if a file descriptor has an error.

Parameters

fd	File descriptor number

Return values

1	The file descriptor has an error
0	The file descriptor does not have an error

See also

dataavail()

writeavail()

charavail()

21.46.3.14 ioctl()

```
int ioctl (
            int fd,
            int cmd )
```

This function controls the selection of input/output control options for stdio: stdin = 0, stdout = 1 and stderr = 2. The legal options are listed in I/O Control Options.

Parameters

fd	The file descriptor number. The three options are stdin (0), stdout (1), and stderr (2).
cmd	The ioctl command options are I/O Control Command Flags and the bit of the associated options as defined by I/O Control Options.

Returns

The old option value.

```
ioctl(0,IOCTL_SET| IOCTL_ALL_OPTIONS); // Sets all options
ioctl(1,IOCTL_CLR| IOCTL_ALL_OPTIONS); // Clears all options
ioctl(1,IOCTL_SET| IOCTL_RX_CHANGE_CRLF|IOCTL_RX_PROCESS_EDITS); // Sets some options
ioctl(2,IOCTL_CLR| IOCTL_RX_CHANGE_CRLF|IOCTL_RX_PROCESS_EDITS); // Clears some options
```

See also

ReplaceStdio()

21.46.3.15 peek()

```
int peek (
            int fd,
            char * c )
```

Peek at the data for the specified file descriptor (fd). Will block forever until at least one byte is available to be read. The byte returned is not removed from the fd buffer, it will be the first byte of data returned on any subsequent read operation.

Parameters

fd	File descriptor
∗c	Pointer to the read destination character string

Return values

1	Number of bytes peeked
<0	Error when reading from file descriptor

See also

TCP Socket Status

close()

read()

PeekWithTimeout()

ReadWithTimeout()

write()

21.46.3.16 PeekWithTimeout()

```
int PeekWithTimeout (
            int fd,
            char * c,
            unsigned long timeout )
```

This function peeks at data from a file descriptor (fd), with a specified timeout value (as opposed to the peek function which will block forever until at least one byte is available to be read). This function can be used to peek from stdio, TCP sockets, or Serial ports.
The byte returned is *not* removed from the fd. The byte returned will be the first byte of data returned on the first subsequent read().

Parameters

fd	The file descriptor number
∗c	A pointer to the read destination
timeout	The number if time ticks to wait for at least 1 byte of data

Return values

0	Timeout, but the file descriptor is still valid
1	Number of bytes peeked
<	0 Error when reading from file descriptor

See also

TCP Socket Status

close()

peek()

read()

ReadWithTimeout()

write()

21.46.3.17 read()

```
int read (
```

```
            int fd,
            char * buf,
            int nbytes )
```

Read data from a file descriptor (fd). This function will block forever until at least one byte is available to be read (as opposed to the ReadWithTimeout() function which reads data from a file descriptor with a specified time-out value). This function can be used to read from stdio, TCP sockets, or Serial ports.

Parameters

fd	The file descriptor number.
**buf*	A pointer to the read destination.
nbytes	Maximum number of bytes to read.

Return values

>0	Number of bytes read
<0	Error when reading from file descriptor

See also

TCP Socket Status

close()

peek()

PeekWithTimeout()

ReadWithTimeout()

ReadAllWithTimeout()

readall()

write()

21.46.3.18 readall()

```
int readall (
            int fd,
            char * buf,
            int nbytes )
```

Read the specified number of bytes from a file descriptor (fd). This function will block until either the requested number of bytes have been read, or a file descriptor error occurs. It can be used to read from stdio, TCP sockets, or Serial ports.

Parameters

fd	File descriptor
**buf*	Pointer to the read destination buffer
nbytes	Number of bytes to read

Return values

0	Timeout
>0	The number of bytes read
-1	File descriptor error, such a TCP socket error

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

ReadAllWithTimeout()

write()

21.46.3.19 ReadAllWithTickTimeout()

```
int ReadAllWithTickTimeout (
            int fd,
            char * buf,
            int nbytes,
            TickTimeout & timeout )
```

Same as ReadWithTimeout(), except the timeout value parameter is of type TickTimeout instead of an unsigned long. The TickTimeout object is more robust in terms of sequential timeout calls and rollover protection.
This function can be used to read from stdio, TCP sockets, or Serial ports. This function will block until on of the following conditions occurs:

- The requested number of bytes have been read.
- The timeout expires. The number of bytes read are returned, and buf contains the data.
- An error occurs on the file descripor, such as a TCP socet error.

Note

The return value must be checked to verify the number of bytes read and that a timeout or error did not occur.

Parameters

fd	The file descriptor number
∗buf	A pointer to the read destination
nbytes	Number of bytes to read
timeout	A reference to the timeout value of type TickTimeout.

Returns

- <0: File descriptor error, such as an invalid TCP socket
- >=0 but less than nbytes: Timeout occurred, returns number of bytes read.
- nbytes: Successfully read the specified number of bytes within the timeout period

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

ReadAllWithTimeout()

readall()

write()

21.46.3.20 ReadAllWithTimeout()

```
int ReadAllWithTimeout (
             int fd,
             char * buf,
             int nbytes,
             unsigned long timeout )
```

Attempt to read the specified number of bytes from a file descriptor, or return with the number of bytes read if the timeout value has expired.
This function can be used to read from stdio, TCP sockets, or Serial ports. This function will block until on of the following conditions occurs:

- The requested number of bytes have been read.
- The timeout expires. The number of bytes read are returned, and buf contains the data.
- An error occurs on the file descripor, such as a TCP socet error.

Note

The return value must be checked to verify the number of bytes read and that a timeout or error did not occur.

Parameters

fd	The file descriptor number
∗buf	A pointer to the read destination
nbytes	Number of bytes to read
timeout	The number of timer ticks to wait for data

Returns

- <0: File descriptor error, such as an invalid TCP socket
- >=0 but less than nbytes: Timeout occurred, returns number of bytes read.
- nbytes: Successfully read the specified number of bytes within the timeout period

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

ReadAllWithTickTimeout()

ReadWithTickTimeout()

readall()

write()

21.46.3.21 ReadWithTickTimeout()

```
int ReadWithTickTimeout (
            int fd,
            char * buf,
            int nbytes,
            TickTimeout & timeout )
```

Same as ReadWithTimeout(), except the timeout value parameter is of type TickTimeout instead of an unsigned long. The TickTimeout object is more robust in terms of sequential timeout calls and rollover protection.

Note

This function operates like the read function in that it reads all available bytes up to the maximum and returns. The addition of a timeout does not cause the function to block until the specified maximum number of bytes are available. As with read, the application must use the return value of the ReadWithTimeout function to determine how many bytes were read, and call the function again if necessary.

When using with a TCP socket, you may want to use the SockReadWithTimeout() function, which returns additional values related to network connections.

Parameters

fd	The file descriptor number.
∗buf	A pointer to the read destination.
nbytes	Maximum number of bytes to read.
timeout	A reference to the timeout value of type TickTimeout.

Return values

0	Timeout
>0	The number of bytes read
-1	Invalid file descriptor, such as if a TCP connection is no longer valid

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

ReadAllWithTimeout()

readall()

SockReadWithTimeout()

write()

21.46.3.22 ReadWithTimeout()

```
int ReadWithTimeout (
            int fd,
            char * buf,
            int nbytes,
            unsigned long timeout )
```

Read data from a file descriptor(fd), or return if at least one byte is not read within the specified timeout. Can be used instead of the read() function, which will block forever until at least one byte is read. File descriptors include such things as stdio, TCP sockets, TLS sockets, and Serial ports.

Note

This function operates like the read function in that it reads all available bytes up to the maximum and returns. The addition of a timeout does not cause the function to block until the specified maximum number of bytes are available. As with read, the application must use the return value of the ReadWithTimeout function to determine how many bytes were read, and call the function again if necessary.

When using with a TCP socket, you may want to use the SockReadWithTimeout() function, which returns additional values related to network connections.

Parameters

fd	The file descriptor number.
∗buf	A pointer to the read destination.
nbytes	Maximum number of bytes to read.
timeout	The number of timer ticks to wait for data.

Return values

0	Timeout
>0	The number of bytes read
-1	Invalid file descriptor, such as if a TCP connection is no longer valid

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTickTimeout()

ReadAllWithTimeout()

readall()

SockReadWithTimeout()

write()

21.46.3.23 RegisterFDCallBack()

```
void RegisterFDCallBack (
            int fd,
            FDCallBack * fp,
            void * pData )
```

Register a callback function to receive notification when an FD state changes. Register a NULL fp to remove the notification.

Parameters

fd	The file descriptor number
∗fp	The function pointer to call back
∗pData	A void ∗ passed to the callback function, can be used for any purpose, such as object references

See also

FDChangeType

FDCCallBack

21.46.3.24 ReplaceStdio()

```
int ReplaceStdio (
            int stdio_fd,
            int new_fd )
```

Maps stdio to any file descriptor (fd).
If the file descriptor generates an error (such as a closed TCP connection), then stdio will be remapped to a negative fd (this will cause stdio to generate errors). When this function is used to remap a stdio channel that has errored, then the error will be cleared.

Parameters

stdio_fd	The stdio file descriptor to map to. Options are stdin (0), stdout (1), and stderr (2).
new_fd	The file descriptor to replace stdio with. A value of 0 returns stdio to the default debug monitor based traps.

Return values

0	If stdio had not been previously mapped
>0	The value of the fd for the previous stdio override

See also

ioctl()

21.46.3.25 select()

```
int select (
            int nfds,
            fd_set * readfds,
            fd_set * writefds,
            fd_set * errorfds,
            unsigned long timeout )
```

Wait for events to occur on one or more I/O resources associated with a set of file descriptors (fds), such as data available to be read, a resource is available to write data, or an error has occurred.
The most common use for this function is to monitor multiple serial and/or TCP file descriptors at one time to read incoming data. For example, an application could use a single RTOS task and a select() to process incoming data on multiple TCP sockets at one time, instead of using multiple tasks or looping through each fd sequentially.
The file descriptors of interest are specified with file descriptor sets (fds) for read, write and error conditions. select() will block until one of the fd conditions are true, or the specified timeout occurs (a value of 0 blocks forever). For example, if the readfds is used for multiple TCP sockets, select() will return if data is available to read on one or more of them. The application would then use macros such as FD_ISSET() to handle reading the data.

Note

A timeout value can be used to cause select() to return periodically for any type of processing an application would like to do in the event no activity is occurring on the fd sets.

Parameters

nfds	The number of file descriptors to examine (This parameter is currently ignored)
∗readfds	Pointer to the fd_set to select for read events, or null if no read events are to be monitored. It is modified on exit to reflect the read availability of the selected fds in the set.
∗writefds	Pointer to the fd_set to select for write availability events, or null if no events are to be monitored. If non-null, it is modified on exit to reflect the write availability of the selected fds in the set.
∗errorfds	Pointer to the fd_set to select for error events, or null if no error events are to be monitored. It is modified on exit to reflect the error state of the selected fds in the set.
timeout	The number of system time ticks to wait before timing out if no events occurred in the selected fd sets. Note: The TICKS_PER_SECOND multiplier should be used to specify the timeout in seconds. For example, TICKS_PER_SECOND ∗ 5 to specify a timeout of 5 seconds.

Return values

0	Timeout occurred
>0	The number of fds in all of the non-null fd_sets

See also

FD_ZERO()

FD_CLR()

FD_SET()

FD_ISSET()

21.46.3.26 write()

```
int write (
            int fd,
            const char * buf,
            int nbytes )
```

Write data to the stream associated with a file descriptor (fd). Can be used to write data to stdio, a TCP socket, or a Serial port.

Note

The write function will block until at least one byte is written, but may not write all the bytes requested. For example, if you want to write 100 bytes, and there was only room in the buffer for 5, then the write() function would return 5.

Parameters

fd	File descriptor
∗buf	Pointer to the buffer to write
nbytes	Maximum number of bytes to write

Return values

0	Timeout
>0	The actual number of bytes written (**Note:** Can be less than the number of bytes requested)
<0	Error occurred, such as a TCP socket error

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

writestring()

writeall()

21.46.3.27 writeall()

```
int writeall (
            int fd,
            const char * buf,
            int nbytes = 0 )
```

Write the specified number of bytes to a file descriptor. Will block until all bytes are sent, or a file descriptor error occurs (such as a TCP socket error). Can be used to write data to stdio, a TCP socket, or a Serial port.

Parameters

fd	File descriptor
**buf*	Pointer to the buffer to write
nbytes	Maximum number of bytes to write

Return values

>=0	The number of bytes written
<0	Error occurred

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

write()

writestring()

21.46.3.28 writeavail()

```
int writeavail (
            int fd )
```

Check the specified file descriptor to determine data can be written.

Parameters

fd	File descriptor number

Return values

1	Data can be written
0	Data cannot be written, such as when an output buffer if full

See also

dataavail()

charavail()

haserror()

21.46.3.29 writestring()

```
int writestring (
            int fd,
            const char * str )
```

Write a null terminated ascii string to the stream associated with a file descriptor (fd). Can be used to write data to stdio, a TCP socket, or a Serial port.

Parameters

fd	File descriptor
∗str	Pointer to the null terminated string to write

Return values

0	Timeout occurred
>0	The number of bytes written (**Note:** Can be less than the number of bytes requested)
<0	Error occurred, such as a TCP socket error

See also

close()

peek()

read()

PeekWithTimeout()

ReadWithTimeout()

write()

writeall()

21.46.3.30 ZeroWaitSelect()

```
int ZeroWaitSelect (
            int nfds,
            fd_set * readfds,
            fd_set * writefds,
            fd_set * errorfds )
```

Returns whether events have occurred on one or more I/O resources associated with a set of file descriptors (fds), and returns immediately.

Parameters

nfds	The number of file descriptors to examine. **Note:** This parameter is currently ignored.
∗readfds	A pointer to the fd_set to select for read events. **Note:** This parameter can be NULL. It is modified on exit to reflect the read availability of the selected fds in the set.
∗writefds	A pointer to the fd_set to select for write availability events. **Note:** This parameter can be NULL. It is modified on exit to reflect the write availability of the selected fds in the set.
∗errorfds	A pointer to the fd_set to select for error events. **Note:** This parameter can be NULL. It is modified on exit to reflect the error state of the selected fds in the set.

Return values

0	If there is no valid fds
>0	The number of fds in all of the non null fd_sets

See also

select()

FD_ZERO()

FD_CLR()

FD_SET()

FD_ISSET()

21.47 IPADDR4 Class

Classes

- class MACADR

  Used to store and manipulate MAC addresses.
- class IPADDR4

  Used to store and manipulate IPv4 addresses in dual stack mode.

Typedefs

- typedef class MACADR MACADR

  Used to store and manipulate MAC addresses.
- typedef IPADDR6 IPADDR

  IPADDR Object Type (either v4 or v6)

Functions

- bool operator== (const MACADR &i, const MACADR &j)

  Check MAC equality.
- bool operator!= (const MACADR &i, const MACADR &j)

  Check MAC inequality.
- bool operator> (const MACADR &i, const MACADR &j)

  Check MAC greater than.

21.47.1 Detailed Description

The IPADDR C++ objects were created to provide a portable and easy implementation to support IPv4 and IPv6 IP addresses. There are 3 types of IPADDR objects:

- IPADDR4: Used exclusively for IPv4 addresses
- IPADDR: Used for either an IPv6 or IPv4 address
- **IPADDR:** The typedef IPADDR is defines a type IPADDR. Its purpose is to provide clarity that an IPADDR object can be used for both IPv4 and IPv6 addresses. Any functions called with a parameter of type IPADDR will automatically execute the correct IPv4 or IPv6 underling function.

21.47.2 Typedef Documentation

21.47.2.1 IPADDR

```
typedef IPADDR6 IPADDR
```

IPADDR Object Type (either v4 or v6)

21.47.2.2 MACADR

```
typedef class MACADR MACADR
```

Used to store and manipulate MAC addresses.

21.47.3 Function Documentation

21.47.3.1 operator"!=()

```
bool operator!= (
            const MACADR & i,
            const MACADR & j )  [inline]
```

Check MAC inequality.

21.47.3.2 operator==()

```
bool operator== (
            const MACADR & i,
            const MACADR & j )  [inline]
```

Check MAC equality.

21.47.3.3 operator>()

```
bool operator> (
            const MACADR & i,
            const MACADR & j )  [inline]
```

Check MAC greater than.

21.48 IPADDR4 Functions

Network Interface functions with IPADDR4 return types.

Functions

- IPADDR4 InterfaceIP (int interface)

  Returns the IPv4 IP address of the specified network interface.
- IPADDR4 InterfaceAutoIP (int interface)

  Returns the IPv4 IP AutoIP address of the specified network interface.
- IPADDR4 InterfaceDNS (int interface)

  Returns the IPv4 DNS address of the specified network interface.
- IPADDR4 InterfaceDNS2 (int interface)

  Returns the second IPv4 DNS address of the specified network interface.
- IPADDR4 InterfaceMASK (int interface)

  Returns the IPv4 network mask of the specified network interface.
- IPADDR4 InterfaceGate (int interface)

  Returns the IPv4 gateway address of the specified network interface.

21.48.1 Detailed Description

Network Interface functions with IPADDR4 return types.

Note

While the recommended method for getting and setting network interface parameters is to obtain a pointer to an InterfaceBlock, there are also non-member functions of the object that can be used for situations that are best served by a single function call, such as just getting a single IP address of an interface. When using those functions pay special attention to the return value type. While IPADDR should be used because it supports both IPADDR4 and IPADDR6 types automatically, some of these helper functions may specifically return an IPADDR4 type.

21.48.2 Function Documentation

21.48.2.1 InterfaceAutoIP()

```
IPADDR4 InterfaceAutoIP (
            int interface )
```

Returns the IPv4 IP AutoIP address of the specified network interface.

Parameters

interface	Interface number

Returns

IPv4 IP address of type IPADDR4

21.48.2.2 InterfaceDNS()

```
IPADDR4 InterfaceDNS (
            int interface )
```

Returns the IPv4 DNS address of the specified network interface.

Parameters

interface	Interface number

Returns

IPv4 IP address of type IPADDR4

21.48.2.3 InterfaceDNS2()

```
IPADDR4 InterfaceDNS2 (
            int interface )
```

Returns the second IPv4 DNS address of the specified network interface.

Parameters

interface	Interface number

Returns

IPv4 IP address of type IPADDR4

21.48.2.4 InterfaceGate()

```
IPADDR4 InterfaceGate (
            int interface )
```

Returns the IPv4 gateway address of the specified network interface.

Parameters

interface	Interface number

Returns

IPv4 IP address of type IPADDR4

21.48.2.5 InterfaceIP()

```
IPADDR4 InterfaceIP (
            int interface )
```

Returns the IPv4 IP address of the specified network interface.

Parameters

interface	Interface number

Returns

IPv4 IP address of type IPADDR4

21.48.2.6 InterfaceMASK()

```
IPADDR4 InterfaceMASK (
            int interface )
```

Returns the IPv4 network mask of the specified network interface.

Parameters

interface	Interface number

Returns

IPv4 network mask of type IPADDR4

21.49 IPADDR6 Class

Classes

- class IPADDR6

  Used to hold and manipulate IPv4 and IPv6 addresses in dual stack mode.

21.49.1 Detailed Description

The IPADDR C++ objects were created to provide a portable and easy implementation to support IPv4 and IPv6 IP addresses. There are 3 types of IPADDR objects:

- IPADDR4: Used exclusively for IPv4 addresses
- IPADDR6: Used for either an IPv6 or IPv4 address
- **IPADDR:** The typedef IPADDR is defines a type IPADDR6. Its purpose is to provide clarity that an IPADDR object can be used for both IPv4 and IPv6 addresses. Any functions called with a parameter of type IPADDR will automatically execute the correct IPv4 or IPv6 underling function.

21.50 Initialization - System Initialization Functions

Functions

- void init ()

  System initialization. Normally called at the beginning of all applications.
- void StartHttp (uint16_t port=80)

  Start the HTTP web server.
- void StartHttps (uint16_t ssl_port=443, uint16_t http_port=80)

  Start the HTTPS secure web server.
- bool WaitForActiveNetwork (uint32_t ticks_to_wait=120 ∗TICKS_PER_SECOND, int interface=-1)

  Wait for an active network connection on at least one interface.
- void EnableSystemDiagnostics ()

  Turn on the diagnostic reports from the config page.
- void EnableSecureConfigServer (bool bSec_Only)

  Enable the minimal http config server to operate over TLS.

21.50.1 Detailed Description

21.50.2 Function Documentation

21.50.2.1 EnableSecureConfigServer()

```
void EnableSecureConfigServer (
            bool bSec_Only )
```

Enable the minimal http config server to operate over TLS.
Due to the nature of the server operating on a non-standard HTTP port, a client (such as a webbrowser) cannot infer whether to utilize TLS unless explicitly told to do so. Therefore, the secure configuration server has two modes it can operate in: a less secure, more compliant manner that allows both plaintext and TLS secured transactions and a HSTS (Http Strict Transport Security) mode that will only respond to plaintext requests with a redirect indicating the TLS requirement.

Parameters

bSec_Only	Whether the SecureConfigServer should disallow non-secure requests

21.50.2.2 EnableSystemDiagnostics()

```
void EnableSystemDiagnostics ( )
```

Turn on the diagnostic reports from the config page.
This should be used under development. It production it has no performance impact, but could leak possibly sensitive information.

21.50.2.3 init()

```
void init ( )
```

System initialization. Normally called at the beginning of all applications.
Initialize the system including the following:

- Enable interrupt driven stdio
- Read the configuration record and set parameters accordingly. If the configuration speficies DHCP, start the DHCP Client. All interfaces will be checked.
- Initialize the network stack
- Start the configuration server for both web and serial
- Assign the UserMain() task priority to MAIN_PRIO
- Enable the Task Monitor diagnostic tool
- Initialize the GDB stub if in debug mode

21.50.2.4 StartHttp()

```
void StartHttp (
            uint16_t port = 80 )
```

Start the HTTP web server.
If no parameters are specified it will listen on port 80, and enable the configuration mirror feature so an application can create its own custom configuration web page.

Parameters

port	Listen port, defaults to port 80

See also

StartHttps(), StopHttp()

21.50.2.5 StartHttps()

```
void StartHttps (
            uint16_t ssl_port = 443,
            uint16_t http_port = 80 )
```

Start the HTTPS secure web server.
If no parameters are specified the web server will listen on ports 443 and 80, and enable the configuration mirror feature so an application can create its own custom configuration web page. If no user key is installed the default system key will be used.

Parameters

ssl_port	Secure listen port, defaults to port 443
http_port	Non-secure listen port, defaults to port 80

See also

StartHttp(), StopHttp()

21.50.2.6 WaitForActiveNetwork()

```
bool WaitForActiveNetwork (
            uint32_t ticks_to_wait = 120 *TICKS_PER_SECOND,
            int interface = -1 )
```

Wait for an active network connection on at least one interface.
If DHCP is enabled it will wait for a lease. If static IP parameters are configured it will wait for link.

Parameters

ticks_to_wait	Number of system time ticks to wait. Default is 120 seconds.
interface	Interface number to check. Default is -1, which checks all interfaces.

21.51 Interrupt Macro - ColdFire

Macro for ColdFire platforms to set up an interrupt function handler including entry and exit code.

21.51.1 Detailed Description

Macro for ColdFire platforms to set up an interrupt function handler including entry and exit code.

Parameters

x	need to add
y	need to add

Example Usage:
INTERRUPT(timetick) { // Interrupt handler code goes here.. }
The example exposes the symbol timetick that can be used as a function pointer to put in the vector base register array.

21.52 Interval Timer

Functions

- int IntervalOSSem (OS_SEM *p_toSem, int num_per_sec, int timer=FIRST_UNUSED_TIMER)

  Posts to a semaphore at the requested interval.
- int IntervalOSFlag (OS_FLAGS *p_toFlag, uint32_t flag_value, int num_per_sec, int timer=FIRST_UNUSED_TIMER)

  Sets a flag at requested interval.
- int IntervalInterruptCallback (void(*p_toCallbackFunc)(), int num_per_sec, int timer=FIRST_UNUSED_TIMER)

  Calls a function at requested interval. Note that the callback function is called from within the timer's interrupt handler so you should treat your callback function as an interrupt.
- void IntervalStop (int timer_number)

  Stops an existing Interval Timer and frees the resource.

21.52.1 Detailed Description

The Interval Timer is used to create a periodic interrupt (IRQ) that can trigger the following:

- Call an Interrupt Service Routine (ISR). An example of this is located in `\nburn\examples`
- Post an OS semaphore
- Post an OS flag

Example: Create an interval timer that posts 20 times per second

```
OS_SEM timerSem;
timerSem.Init();

int timerNumer = IntervalOSSem(&timerSem, 20);
if(timerNumer < 0)
{
   iprintf("Error %d\r\n", timerNumer);
   break;
}
```

When you have no further use for the interval timer, call IntervalStop() so that the hardware timer is released.

See also

High Resolution Delay Timer

Stopwatch Timer

OSTimeDly()

21.52.2 Function Documentation

21.52.2.1 IntervalInterruptCallback()

```
int IntervalInterruptCallback (
            void(*)() p_toCallbackFunc,
            int num_per_sec,
            int timer = FIRST_UNUSED_TIMER )
```

Calls a function at requested interval. Note that the callback function is called from within the timer's interrupt handler so you should treat your callback function as an interrupt.

Parameters

p_toCallbackFunc	Pointer to the callback function
num_per_sec	Number of posts per second, minimum is 20 posts per second.
timer	Optional parameter, timer number of -1 for first unused timer

Returns

The timer number, -1 if no timer is available, or -2 if num_per_sec is invalid.

21.52.2.2 IntervalOSFlag()

```
int IntervalOSFlag (
            OS_FLAGS * p_toFlag,
            uint32_t flag_value,
            int num_per_sec,
            int timer = FIRST_UNUSED_TIMER )
```

Sets a flag at requested interval.

Parameters

p_toFlag	Pointer to the semaphore
flag_value	OS Flag value to set
num_per_sec	Number of posts per second, minimum is 20 posts per second.
timer	Optional parameter, timer number of -1 for first unused timer

Returns

The timer number, -1 if no timer is available, or -2 if num_per_sec is invalid.

21.52.2.3 IntervalOSSem()

```
int IntervalOSSem (
            OS_SEM * p_toSem,
            int num_per_sec,
            int timer = FIRST_UNUSED_TIMER )
```

Posts to a semaphore at the requested interval.

Parameters

p_toSem	Pointer to the semaphore
num_per_sec	Number of posts per second, minimum is 20 posts per second
timer	Optional parameter, timer number of -1 for first unused timer

Returns

The timer number, -1 if no timer is available, or -2 if num_per_sec is invalid.

21.52.2.4 IntervalStop()

```
void IntervalStop (
            int timer_number )
```

Stops an existing Interval Timer and frees the resource.

Parameters

timer_number	Number of the timer to stop.

21.53 JSON Lexer

NetBurner's JSON library. See more details and examples below.

Classes

- struct JsonAllocString

  A list of large strings that are created with malloc.
- class JsonRef

  Represents a positional reference (pointer) of a location inside a ParsedJsonDataSet object

- class ParsedJsonDataSet

  A class to create, read, and modify a JSON object.

Typedefs

- typedef void() CharOutputFn(const char *chars, int len, void *blob)

  Helper function typedef for print functions.

Enumerations

- enum json_primitive_type {
  UNDEFINED , BEGIN_ARRAY , BEGIN_OBJECT , END_ARRAY ,
  END_OBJECT , NAME , STRING , VALUE_SEPERATOR ,
  NUMBER , FALSE_EL , TRUE_EL , NULL_EL ,
  STRING_TOO_BIG , ALLOC_STRING , NOTFOUND , EOF_EL }

  The following types define the basic building blocks that make up a JSON data set. These are the values that will be returned from the functions used to parse the data set. Member functions include operators to return specific data type, as well as type validity checks.

21.53.1 Detailed Description

NetBurner's JSON library. See more details and examples below.
The library enables rapid, performant parsing, traversal, and querying of JSON data. It also works seamlessly as a buffer_object for the Web Client's DoGet() function.
ParsedJsonDataSet is the root object that holds the document, and JsonRef is the result of each query function, serving as a pointer to a location in the document but also as a chainable query and inspection interface.
See also the JSON Lexer Example Applications.

Example

```
ParsedJsonDataSet pjd;
bool result = DoGet("https://www.example.com/users.json", pjd);
// pjd.PrintObject(true);
// pjd("users").PrintChildren(true);
if (!result ||
    !pjd("users")[0].IsObject() ||
    !pjd("users")[0]("id").IsNumber() ||
    !pjd("users")[0]("name").IsString()
) {
    return false;
}
printf("Got user %d name %s",
    (int)pjd("users")[0]("id"),
    (const char *)pjd("users")[0]("name")
);
```

21.53.2 Typedef Documentation

21.53.2.1 CharOutputFn

```
typedef void() CharOutputFn(const char *chars, int len, void *blob)
```
Helper function typedef for print functions.

Parameters

chars	The characters to print.
len	The length of the characters to print.
blob	A pointer to where they should be printed.

21.53.3 Enumeration Type Documentation

21.53.3.1 json_primitive_type

```
enum json_primitive_type
```
The following types define the basic building blocks that make up a JSON data set. These are the values that will be returned from the functions used to parse the data set. Member functions include operators to return specific data type, as well as type validity checks.

Enumerator

UNDEFINED	Not a defined primitive type. This indicates an error somewhere.
BEGIN_ARRAY	[- Signals the start of an array.
BEGIN_OBJECT	{ - Signals the start of an object.
END_ARRAY	] - Signals the end of an array.
END_OBJECT	} - Signals the end of an object.
NAME	"xxx": - Signals the name of a name/value pair.
STRING	"xxx" - An element with a string value.
VALUE_SEPERATOR	, - Signals the separation between to arrays, objects, or values.
NUMBER	An element with a number value.
FALSE_EL	An element with the value false.
TRUE_EL	An element with the value true.
NULL_EL	An element with the null value.
STRING_TOO_BIG	Error, we got a sting but it was too big to fit in the storage scheme (1500 bytes).
ALLOC_STRING	We allocated a large string.
NOTFOUND	Return value when we don't find what we are looking for.
EOF_EL	Last token in the data set.

21.54 Low Level Processing (NetDoRx)

Callback functions to add custom Ethenet handlers and pass Ethernet frames in the bottom of the TCP/IP stack.

Typedefs

- typedef int(∗ netDoRXFunc) (PoolPtr, uint16_t, int)

  Typedef interface for all network rx processing functions.

Functions

- netDoRXFunc SetCustomNetDoRX (netDoRXFunc customFunc)

  Registers a new custom ethernet handler to run prior to the primary handler.
- netDoRXFunc ClearCustomNetDoRX ()

  Clears the custom ethernet handler, resetting the handler to NULL.
- int NetDoRX (PoolPtr pp, uint16_t ocount, int if_num)

  Entry function for Ethernet frames into the system TCP/IP stack.

21.54.1 Detailed Description

Callback functions to add custom Ethenet handlers and pass Ethernet frames in the bottom of the TCP/IP stack.

Note

Custom handlers require that the system is(re)compiled with the ALLOW_CUSTOM_NET_DO_RX macro defined.

Warning

These functions are for advanced developers only! Code written can affect all receive network processing.

21.54.2 Typedef Documentation

21.54.2.1 netDoRXFunc

```
typedef int(* netDoRXFunc) (PoolPtr, uint16_t, int)
```

Typedef interface for all network rx processing functions.

```
Parameter Types:
PoolPtr   The buffer structure containing the received network data
uint16_t  The total number of receive bytes of network data
int       The network interface number of the interface that the frame was received from
```

Returns

Boolean integer value declaring whether the frame was handled or if it should continue to be processed 1 = proccessed and should be released 0 = not processed and should continue being handled

21.54.3 Function Documentation

21.54.3.1 ClearCustomNetDoRX()

```
netDoRXFunc ClearCustomNetDoRX ( )  [inline]
```

Clears the custom ethernet handler, resetting the handler to NULL.

Returns

Previously registered custom handler (NULL if there was none)

21.54.3.2 NetDoRX()

```
int NetDoRX (
            PoolPtr pp,
            uint16_t ocount,
            int if_num )
```

Entry function for Ethernet frames into the system TCP/IP stack.

Parameters

pp	PoolPtr to PoolBuffer containing the ethernet frame
ocount	A count of the number of octets (bytes) of ethernet payload
if_num	The interface number that the frame came in on.

Returns

The number of system time ticks to wait before calling again, call with null if timeout.

21.54.3.3 SetCustomNetDoRX()

```
netDoRXFunc SetCustomNetDoRX (
            netDoRXFunc customFunc )  [inline]
```

Registers a new custom ethernet handler to run prior to the primary handler.

Parameters

customFunc	The netDoRXFunc to call when processing an ethernet frame

Returns

Previously registered custom handler (NULL if there was none)

21.55 MCAN Constants

Macros

- #define **CAN_DATA_STORE_SIZE** (512)

  Receive OS_FIFO Buffer Size.

Variables

- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_FIFO_0_NUM = 32
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_FIFO_1_NUM = 1
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_BUFFER_NUM = 1
- const uint32_t mcanMODM7AE70::CONF_MCAN_TX_BUFFER_NUM = 8
- const uint32_t mcanMODM7AE70::CONF_MCAN_TX_FIFO_QUEUE_NUM = 1
- const uint32_t mcanMODM7AE70::CONF_MCAN_TX_EVENT_FIFO = 8
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_STANDARD_ID_FILTER_NUM = 32
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM = 32

21.55.1 Detailed Description

MCAN Constants

21.55.2 Variable Documentation

21.55.2.1 CONF_MCAN_RX_BUFFER_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_RX_BUFFER_NUM = 1
```

Range: 1..64

21.55.2.2 CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM = 32
```
Range: 1..64

21.55.2.3 CONF_MCAN_RX_FIFO_0_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_RX_FIFO_0_NUM = 32
```
Range: 1..64

21.55.2.4 CONF_MCAN_RX_FIFO_1_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_RX_FIFO_1_NUM = 1
```
Range: 1..64

21.55.2.5 CONF_MCAN_RX_STANDARD_ID_FILTER_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_RX_STANDARD_ID_FILTER_NUM = 32
```
Range: 1..128

21.55.2.6 CONF_MCAN_TX_BUFFER_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_TX_BUFFER_NUM = 8
```
Range: 1..16

21.55.2.7 CONF_MCAN_TX_EVENT_FIFO

```
const uint32_t mcanMODM7AE70::CONF_MCAN_TX_EVENT_FIFO = 8
```
Range: 1..32

21.55.2.8 CONF_MCAN_TX_FIFO_QUEUE_NUM

```
const uint32_t mcanMODM7AE70::CONF_MCAN_TX_FIFO_QUEUE_NUM = 1
```
Range: 1..16

21.56 MCF5441x (DSPI)

Modules

- DSPI state

Classes

- class DSPIModule

  DSPIModule is a SPI communications driver. It is an object based driver, which allows for low overhead multiplexing between peripherals with different bus configurations.

Enumerations

- enum csReturnType {
  DEASSERT_NEVER = 0 , DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 ,
  DEASSERT_NEVER = 0 ,
  DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 }

  Chip select return types.
- enum spiChipSelect {
  CHIP_SELECT_0 = 0xFE , CHIP_SELECT_1 = 0xFD , CHIP_SELECT_2 = 0xFB , CHIP_SELECT_3 = 0xF7 ,
  CHIP_SELECT_DISABLED = 0xFF , CHIP_SELECT_0 = 0 , CHIP_SELECT_1 = 1 , CHIP_SELECT_2 = 2 ,
  CHIP_SELECT_3 = 3 , CHIP_SELECT_DISABLED = 0xFF }

Chip select number.

- enum spiChipSelectPolarity { CS_ASSERT_LOW = 0x00 , CS_ASSERT_HIGH = 0xFF , CS_ASSERT_LOW = 0 , CS_ASSERT_HIGH = 1 }

  Chip select polarity.

Functions

- uint8_t DSPIInit (uint8_t SPIModule=DEFAULT_DSPI_MODULE, uint32_t Baudrate=2000000, uint8_t QueueBitSize=8, uint8_t CS=0x00, uint8_t CSPol=0x0F, uint8_t ClkPolarity=0, uint8_t ClkPhase=1, BOOL DoutHiz=TRUE, uint8_t QCD=0, uint8_t DTL=0)

  Initialize a DSPI module.
- uint8_t QSPIInit (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x0F, uint8_t chipSelectPolarity=1, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize Queued Serial Peripheral Interface (QSPI)
- uint8_t QSPIStart (puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL)

  Start QSPI Data Transfer.
- BOOL QSPIdone ()

  Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.

21.56.1 Detailed Description

Supported platforms:

- MOD54415
- MOD54417
- NANO54415
- SB800EX

The DMA serial peripheral interface (DSPI) interfaces (up to 3 depending on your platform) provide a synchronous serial bus for communication between the processor and an external peripheral device. DSPI interface modules are numbered from 0 to 3. DSPI 0 is reserved for system use. The DSPI supports up to 32 queued SPI transfers (16 receive and 16 transmit) in the DSPI resident FIFOs eliminating CPU intervention between transfers.
The DSPI specification allows for sharing the SPI bus with multiple devices. However, when designing your system, you must consider what devices are connected and the desired throughput. For example, if you are using the Embedded Flash File System (EFFS) with external Flash cards, or the WiFi module, they will require exclusive use of the DSPI interface.
The DSPI uses the following hardware signals:

- SCLK Serial clock output from master.
- MOSI Master data output (transmit), slave data input (receive).
- MISO Master data input (receive), slave data output (transmit).
- QSPI_CSn Optional QSPI chip selects. Note that you can also use GPIO signals and control them manually for the chip select functionality. In some cases, there are advantages to doinig it this way.

DSPI Configuration and Initialization
Use of the DSPI requires:

1. Identify which pins on your NetBurner device provide primary or secondary QSPI functionality. Use the NetBurner Pins class functions to configure these pins for QSPI operation.
2. Single task applications: you can choose to call the DSPIdone() function to determine when the DSPI transfer is complete, or use a semaphore and pend for completion.

3. Multi-task applications: If your application has more than one task that will access the same DSPI, you must create a semaphore to control access to the DSPI resource, just as you would with any shared resource.
4. Initialize the DSPI with the DSPIInit() function.
5. When you have data to send, or want to read data from the DSPI slave device, call the QSPIStart() function to initiate the data transfer.
6. The semaphore will post when the data transfer is complete.

21.56.2 Enumeration Type Documentation

21.56.2.1 csReturnType

`enum csReturnType`

Chip select return types.
Chip select deassertion modes. Used to determine when the driver should deassert chip selects during SPI transfer.

Enumerator

DEASSERT_NEVER	The chip select used for the transaction should remain asserted, even after the transaction is complete.
DEASSERT_AFTER_LAST	The chip select should remain asserted for the full duration of the transaction, and only be deasserted after the final transfer.
DEASSERT_EVERY_TRANSFER	The chip select should be deasserted between every transfer within the transaction.
DEASSERT_NEVER	The chip select used for the transaction should remain asserted, even after the transaction is complete.
DEASSERT_AFTER_LAST	The chip select should remain asserted for the full duration of the transaction, and only be deasserted after the final transfer.
DEASSERT_EVERY_TRANSFER	The chip select should be deasserted between every transfer within the transaction.Deassert chip select After every transfer.

21.56.2.2 spiChipSelect

`enum spiChipSelect`

Chip select number.
If supported by the SPI peripheral, these values can be used to configure the chip select to be used during a SPI transfer.

Enumerator

CHIP_SELECT_0	Configure bit mask for SPI chip select 0.
CHIP_SELECT_1	Configure bit mask for SPI chip select 1.
CHIP_SELECT_2	Configure bit mask for SPI chip select 2.
CHIP_SELECT_3	Configure bit mask for SPI chip select 3.
CHIP_SELECT_DISABLED	Configure bit mask to disable chip select.
CHIP_SELECT_0	Configure the SPI peripheral to use chip select 0.
CHIP_SELECT_1	Configure the SPI peripheral to use chip select 1.
CHIP_SELECT_2	Configure the SPI peripheral to use chip select 2.
CHIP_SELECT_3	Configure the SPI peripheral to use chip select 3.
CHIP_SELECT_DISABLED	Configure the SPI peripheral to disable chip select.

21.56.2.3 spiChipSelectPolarity

```
enum spiChipSelectPolarity
```

Chip select polarity.
These values can be used to configure the chip select's asserted/deasserted logic level. These values will configure all chip selects to the same configuration. If you intend on using an SPI peripheral with multiple chip selects with different polarities, these values cannot not be used.

Enumerator

CS_ASSERT_LOW	Bit mask configures all active chip selects as active low, inactive high.
CS_ASSERT_HIGH	Bit mask configures all active chip selects as inactive low, active high.
CS_ASSERT_LOW	Assert all chip selects to logic level LOW during a SPI transaction.
CS_ASSERT_HIGH	Assert all chip selects to logic level HIGH during a SPI transaction.

21.56.3 Function Documentation

21.56.3.1 DSPIInit()

```
uint8_t DSPIInit (
              uint8_t SPIModule = DEFAULT_DSPI_MODULE,
              uint32_t Baudrate = 2000000,
              uint8_t QueueBitSize = 8,
              uint8_t CS = 0x00,
              uint8_t CSPol = 0x0F,
              uint8_t ClkPolarity = 0,
              uint8_t ClkPhase = 1,
              BOOL DoutHiz = TRUE,
              uint8_t QCD = 0,
              uint8_t DTL = 0 )
```

Initialize a DSPI module.
Notes:

- The maximum baud rate is CPU_CLOCK / 3
- Will initialize to the highest available baud rate that does not exceed the maximum
- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for greater than 8 bits per transfer then the data must be uint16_t aligned
- If configured for greater than 16 bits per transfer then the data must be uint32_t aligned
- Chip select is based on chipSelectPolarity
- 0 data captured leading edge of DSPI_CLK, changed following edge.
- 1 data changed leading edge of DSPI_CLK, captured following edge.
- 0 default is as close to 1/2 DSPI_CLK without going under, keeping with the interface to SPI
- 0 default is 17/(system clock / 2), in keeping with interface to QSPI

Parameters

SPIModule	SPI module number, 0 - 1

Parameters

Baudrate	Maximum baud rate requested
QueueBitSize	Number of bits per transfer: 8, 16 or 32
CS	SPI chip selects to use for transfer
CSPol	0 = inactive logic level low, 1 = high
ClkPolarity	0 = inactive logic level low, 1 = high
ClkPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
DoutHiz	Data output high impedance between transfers
QCD	Delay from chip select to valid clock (default is 0). QCD is a value in the QDLYR register and will change the delay between the assertion of the chip select and the start of the DSPI clock. Default setting of one half DSPI clk will be used if parameter is specified as 0x0 or not included.
DTL	DTL is a value in the QDLYR register and will change the delay following a tranfer of a single uint16_t in the DSPI queue. Default reset value of 17/(fsys/2) will be used if parameter is specified 0 or not included.

Returns

Current stat of DSPI bus DSPI state

21.56.3.2 QSPIdone()

```
BOOL QSPIdone (
            void  )  [inline]
```

Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.

Return values

TRUE	When transfer is complete
FALSE	On error

See also

QSPIInit()

QSPIStart()

21.56.3.3 QSPIInit()

```
uint8_t QSPIInit (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x0F,
            uint8_t chipSelectPolarity = 1,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )  [inline]
```

Initialize Queued Serial Peripheral Interface (QSPI)

Notes: (3) Transfers > 8 must be word aligned (4) Select based on chipSelectPolarity (5) 0 data captured leading edge of QSPI_CLK, changed following edge. (6) 1 data changed leading edge of QSPI_CLK, captured following

edge. (7) 0 default is 1/2 QSPI_CLK or see MFCXXXX reference manual (8) 0 default is 8192/(system clock / 2) see MFCXXXX reference manual

Parameters

baudRateInBps	Maximum requested baud rate in bits per second. The maximum possible baud rate is determined by the system clock. The function will select the highest possible baud rate if the specified value cannot be achieved.
transferSizeInBits	Size of data values to be transferred. Values can be 8, 16 or 32 bits. Note that 16 and 32 bit size values must be word (16-bit) aligned (16-bit).
peripheralChipSelects	Peripheral chip select drive level. Used to select an external device for serial data transfer. More than one chip select may be active at once, and more than one device can be connected to each chip select. Bits 3-0 map directly to QSPI_CS[3:0], respectively. For each bit: • 0 = Chip select is 0 during a transfer. • 1 = Chip select is 1 during a transfer.
chipSelectPolarity	Peripheral chip select inactive level (no transfer in progress). Applies to all chip selects. • 0 = 0 when inactive • 1 = 1 when inactive
clockPolarity	Clock polarity: • 0 = The inactive state value of QSPI_CLK is logic level 0. • 1 = The inactive state value of QSPI_CLK is logic level 1.
clockPhase	Clock phase: • 0 = Data captured on the leading edge of QSPI_CLK and changed on the following edge of QSPI_CLK. • 1 = Data changed on the leading edge of QSPI_CLK and captured on the following edge of QSPI_CLK.
doutHiz	Data output high impedance enable. Selects QSPI_DOUT mode of operation. • 0 = Default value after reset. QSPI_DOUT is actively driven between transfers. • 1 = QSPI_DOUT is high impedance between transfers.
csToClockDelay	QCD, QSPICLK delay. When the DSCK bit in the command RAM is set this field determines the length of the delay from assertion of the chip selects to valid QSPI_CLK transition.
delayAfterTransfer	Delay after transfer. When the DT bit in the command RAM is set this field determines the length of delay after the serial transfer.

Returns

groupQspiState

See also

QSPIInit()

QSPIStart()

21.56.3.4 QSPIStart()

```
uint8_t QSPIStart (
            puint8_t transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            OS_SEM * finishedSem = NULL )  [inline]
```

Start QSPI Data Transfer.

Parameters

transmitBufferPtr	Pointer to the buffer to use in the transfer. Specify NULL for receive only.
receiveBufferPtr	Pointer to buffer to store received data. NULL for transmit only.
byteCount	Number of bytes to send or receive. If the data size is greater than 8-bit, you must provide the total number of bytes.
finishedSem	Pointer to preinitialized semaphore to post to when transfer is complete. A value of NULL disables the semaphore function.

Returns

groupQspiState

See also

QSPIInit()

QSPIdone()

21.57 MCF5441x (QSPI)

Modules

- QSPI state

Classes

- struct QSPI_Record

  This struct contains the major variables/configurations used for a QSPI transfer.

Functions

- uint8_t QSPIInit (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChip↩ Selects=0x0F, uint8_t chipSelectPolarity=1, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL dout↩ Hiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize Queued Serial Peripheral Interface (QSPI)
- uint8_t QSPIStart (puint8_t transmitBufferPtr, volatile uint8_t ∗receiveBufferPtr, uint32_t byteCount, OS_SEM ∗finishedSem=NULL)

  Start QSPI Data Transfer.
- BOOL QSPIdone (void)

  Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.

21.57.1 Detailed Description

Supported platforms:

- MOD54415

- MOD54417
- NANO54415
- SB800EX

Note

The SPI interface modules (up to 4 depending on the platform) on the MCF5441x processor are termed DMA SPI (DSPI). Earlier ColdFire processors had a single SPI with a queue, and were termed QSPI. This header file is here as a guide for customers using earlier ColdFire processors to make porting code easier. However, we highly recommend using the DSPI driver instead. The calls are nearly identical.

Introduction

The Queued Serial Peripheral Interface (QSPI) is a full duplex synchronous serial data link that operates in master/slave mode. The QSPI is similar to SPI, but also provided a hardware queue for transmit data, receive data, and commands. Up to 16 transfers can be queued at one time, eliminating CPU intervention between transfers. The NetBurner QSPI driver and API provide interrupt driven QSPI communication.

The QSPI specification allows for sharing the QSPI bus with multiple devices. However, when designing your system you must consider what devices are connected and the desired throughput. For example, if you are using the Embedded Flash File System (EFFS) with external SD flash cards, or the WiFi module, they will require exclusive use of the QSPI.

The QSPI uses the following hardware signals:

- SCLK Serial clock output from master
- MOSI Master data output (transmit), slave data input (receive)
- MISO Master data input (receive), slave data output (transmit)
- QSPI_CS Optional QSPI chip selects

Programming tip: If you are new to QSPI and require a chip select for your device, we recommend using a GPIO signal during initial development to simplify your application code. Once proper operation has been achieved, modify the code to use the QSPI chip selects.

QSPI Configuration and Initialization

Use of the QSPI requires:

1. Identify which pins on your NetBurner device provide primary or secondary QSPI functionality. Use the NetBurner Pins class functions to configure these pins for QSPI operation.
2. Single task applications: you can choose to call the QSPIdone() function to determine when the QSPI transfer is complete, or use a semaphore and pend for completion.
3. Multi-task applications: If your application has more than one task that will access the QSPI, your must create a semaphore to control access to the QSPI resource.
4. Initialize the QSPI with the QSPIInit() function.
5. When you have data to send, or want to read data from the QSPI slave device, call the QSPIStart() function to initiate the data transfer.
6. The semaphore will post when the data transfer is complete.

21.57.2 Function Documentation

21.57.2.1 QSPIdone()

```
BOOL QSPIdone (
            void  )  [inline]
```

Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.

Return values

TRUE	When transfer is complete
FALSE	On error

See also

QSPIInit()

QSPIStart()

Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.
Note that the 'Q' stands for Queued SPI

Returns

true if DSPI is finished, false if active

21.57.2.2 QSPIInit()

```
uint8_t QSPIInit (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x0F,
            uint8_t chipSelectPolarity = 1,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )  [inline]
```

Initialize Queued Serial Peripheral Interface (QSPI)
Notes: (3) Transfers > 8 must be word aligned (4) Select based on chipSelectPolarity (5) 0 data captured leading edge of QSPI_CLK, changed following edge. (6) 1 data changed leading edge of QSPI_CLK, captured following edge. (7) 0 default is 1/2 QSPI_CLK or see MFCXXXX reference manual (8) 0 default is 8192/(system clock / 2) see MFCXXXX reference manual

Parameters

baudRateInBps	Maximum requested baud rate in bits per second. The maximum possible baud rate is determined by the system clock. The function will select the highest possible baud rate if the specified value cannot be achieved.
transferSizeInBits	Size of data values to be transferred. Values can be 8, 16 or 32 bits. Note that 16 and 32 bit size values must be word (16-bit) aligned (16-bit).
peripheralChipSelects	Peripheral chip select drive level. Used to select an external device for serial data transfer. More than one chip select may be active at once, and more than one device can be connected to each chip select. Bits 3-0 map directly to QSPI_CS[3:0], respectively. For each bit: • 0 = Chip select is 0 during a transfer. • 1 = Chip select is 1 during a transfer.
chipSelectPolarity	Peripheral chip select inactive level (no transfer in progress). Applies to all chip selects. • 0 = 0 when inactive • 1 = 1 when inactive

Parameters

clockPolarity	Clock polarity: • 0 = The inactive state value of QSPI_CLK is logic level 0. • 1 = The inactive state value of QSPI_CLK is logic level 1.
clockPhase	Clock phase: • 0 = Data captured on the leading edge of QSPI_CLK and changed on the following edge of QSPI_CLK. • 1 = Data changed on the leading edge of QSPI_CLK and captured on the following edge of QSPI_CLK.
doutHiz	Data output high impedance enable. Selects QSPI_DOUT mode of operation. • 0 = Default value after reset. QSPI_DOUT is actively driven between transfers. • 1 = QSPI_DOUT is high impedance between transfers.
csToClockDelay	QCD, QSPICLK delay. When the DSCK bit in the command RAM is set this field determines the length of the delay from assertion of the chip selects to valid QSPI_CLK transition.
delayAfterTransfer	Delay after transfer. When the DT bit in the command RAM is set this field determines the length of delay after the serial transfer.

Returns

groupQspiState

See also

QSPIInit()

QSPIStart()

Initialize Queued Serial Peripheral Interface (QSPI)
Note that the 'Q' stands for Queued SPI
Will use the default SPI module, 0.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.
- If DSPI_Finished points to a semaphore, then the DSPI will POST to it when the transfer is complete.
- The semaphore is optional, but it can increase efficiency.

Parameters

baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8, 16 or 32
peripheralChipSelects	SPI chip selects to use for transfer
chipSelectPolarity	0 = inactive logic level low, 1 = high
clockPolarity	0 = inactive logic level low, 1 = high

Parameters

clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	Delay from chip select to valid clock (default is 0)
delayAfterTransfer	Chip select mode spiChipSelect

Returns

The current state of the SPI bus dspiState

21.57.2.3 QSPIStart()

```
uint8_t QSPIStart (
            puint8_t transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            OS_SEM * finishedSem = NULL )  [inline]
```

Start QSPI Data Transfer.

Parameters

transmitBufferPtr	Pointer to the buffer to use in the transfer. Specify NULL for receive only.
receiveBufferPtr	Pointer to buffer to store received data. NULL for transmit only.
byteCount	Number of bytes to send or receive. If the data size is greater than 8-bit, you must provide the total number of bytes.
finishedSem	Pointer to preinitialized semaphore to post to when transfer is complete. A value of NULL disables the semaphore function.

Returns

groupQspiState

See also

QSPIInit()

QSPIdone()

Start QSPI Data Transfer.
Note that the 'Q' stands for Queued SPI

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.
- If DSPI_Finished points to a semaphore, then the DSPI will POST to it when the transfer is complete.
- The semaphore is optional, but it can increase efficiency.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
receiveBufferPtr	Pointer to the buffer to store the received data
byteCount	Number of bytes to transmit
finishedSem	Optional semaphore to post to when finished

Returns

The current state of the SPI bus dspiState

21.58 MCF5441x I2C

Modules

- Multichannel I2C Macros

  I2C Macros.
- Multichannel I2C Return Values

  Function return values.

Functions

- void MultiChannel_I2CInit (int moduleNum=DEFAULT_I2C_MODULE, uint8_t slave_Addr=0x08, uint8_t freqdiv=0x3C)

  Initialize the I2C peripheral module.
- uint8_t MultiChannel_I2CSendBuf (int moduleNum, uint8_t addr, puint8_t buf, int num, bool stop=true)

  Send a buffer of bytes to an I2C device.
- uint8_t MultiChannel_I2CReadBuf (int moduleNum, uint8_t addr, puint8_t buf, int num, bool stop=true)

  Read a number of bytes from an I2C device and store in the specified buffer.
- void I2CMultiChannelResetPeripheral (int moduleNum)

  Reset the specified I2C peripheral module.
- uint8_t MultiChannel_I2CRestart (int moduleNum, uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Restart communication with a I2C device.
- uint8_t MultiChannel_I2CStart (int moduleNum, uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait=I2C_START_TIMEOUT)

  Send an I2C start to an I2C device to begin communication.
- uint8_t MultiChannel_I2CStop (int moduleNum=DEFAULT_I2C_MODULE, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Issue an I2C stop terminate communication with an I2C device and release the bus.
- uint8_t MultiChannel_I2CSend (int moduleNum, uint8_t val, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Send a single byte on the I2C bus.
- uint8_t MultiChannel_I2CRead (int moduleNum, puint8_t val, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Read a single byte from the I2C bus.

21.58.1 Detailed Description

Supported platforms:

- MOD54415
- MOD54417
- NANO54415

- SB800EX

  The platforms have multiple I2C interfaces called "modules". The available I2C modules available on a particular platform are determined by the connector configuration of that platform. Available I2C module numbers are defined below. Since the processor pins are multiplexed with many other functions, the modules chosen are not sequential. The default module number is 0.

  I2C MODULE NUMBERS BY PLATFORM:

  – MOD5441x: 0, 1, 2, 4, 5.
  – NANO54415: 0, 1, 4, 5.

The most common I2C functions used are listed below. They automatically handle I2C start and stop conditions, as well as checking for I2C bus errors and recovery:

- MultiChannel_I2CInit ()
- MultiChannel_I2CSendBuf ()
- MultiChannel_I2CReadBuf ()
- I2CMultiChannelResetPeripheral()

21.58.2 Function Documentation

21.58.2.1 I2CMultiChannelResetPeripheral()

```
void I2CMultiChannelResetPeripheral (
            int moduleNum )
```

Reset the specified I2C peripheral module.
Clears all interrupts, I2C conditions, and releases the bus.

Parameters

moduleNum	I2C peripheral module number.

21.58.2.2 MultiChannel_I2CInit()

```
void MultiChannel_I2CInit (
            int moduleNum = DEFAULT_I2C_MODULE,
            uint8_t slave_Addr = 0x08,
            uint8_t freqdiv = 0x3C )
```

Initialize the I2C peripheral module.
The frequency is based on the internal bus clock of the processor. The formula is: Freq = (250MHZ (sys clock)/2) / freqdiv to give max baud rate of the master mode I2C bus clock. Values for freqdiv are found in the i2fdr section of the NXP MCF5441x User Manual.

Parameters

moduleNum	I2C peripheral module number.
slave_Addr	I2C address of the module. If not specified, default value is 0x08.
freqdiv	I2C module frequency divider. If not specified, default value is 0x3C, which is the maximum value of 100KHz.

21.58.2.3 MultiChannel_I2CRead()

```
uint8_t MultiChannel_I2CRead (
            int moduleNum,
            puint8_t val,
            uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT )
```

Read a single byte from the I2C bus.
Reads a single byte from the I2C bus and stores it in val. A start or restart must have been previously issued to become the bus master, and a stop must be issued to release the bus when you are finished reading the desired number of bytes.
Does not handle no Ack of last byte, which needs to have a I2C_SET_NO_ACK called before the last read.

Parameters

moduleNum	I2C peripheral module number.
val	Location to store the byte read.
ticks_to_wait	Amount of time to wait for a start transmission to complete, specified as system time ticks. The TICKS_PER_SECOND macro can be used with a multiplier or dividor. Includes the time to receive an ack from the I2C device. If not specified, the default time of I2C_RX_TX_TIMEOUT will be used.

Returns

The state of the I2C bus: I2C Return Values

21.58.2.4 MultiChannel_I2CReadBuf()

```
uint8_t MultiChannel_I2CReadBuf (
            int moduleNum,
            uint8_t addr,
            puint8_t buf,
            int num,
            bool stop = true )
```

Read a number of bytes from an I2C device and store in the specified buffer.
Sends an I2C start, reads the specified number of bytes into buf, and and finishes with our without an I2C stop (default is stop). If a stop is not issued, for purposes such as a restart, a stop must be called manually.

Parameters

moduleNum	I2C peripheral module number.
addr	I2C device address.
buf	The buffer to store the data.
num	The number of bytes to read.
stop	Send a stop signal at the end of the transaction. If not specified, the default value is true.

Returns

The state of the I2C bus: I2C Return Values

21.58.2.5 MultiChannel_I2CRestart()

```
uint8_t MultiChannel_I2CRestart (
            int moduleNum,
            uint8_t addr,
```

```
            bool Read_Not_Write,
            uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT )
```

Restart communication with a I2C device.

Restarts communication with an I2C device after completion of a read or write command in which a stop was not issued. Enables communicate on bus again without giving up control as the I2C bus master.

Parameters

moduleNum	I2C peripheral module number.
addr	I2C device address.
Read_Not_Write	Specifies the next bus operation: true = read, false = write.
ticks_to_wait	Amount of time to wait for a start transmission to complete, specified as system time ticks. The TICKS_PER_SECOND macro can be used with a multiplier or divider. Includes the time to receive an ack from the I2C device. If not specified, the default time of I2C_RX_TX_TIMEOUT will be used.

Returns

The state of the I2C bus: I2C Return Values

21.58.2.6 MultiChannel_I2CSend()

```
uint8_t MultiChannel_I2CSend (
            int moduleNum,
            uint8_t val,
            uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT )
```

Send a single byte on the I2C bus.

Sends a single byte on the I2C bus. A start or restart must have been previously issued to become the bus master, and a stop must be issued to release the bus when you are finished with writing the desired number of bytes.

Parameters

moduleNum	I2C peripheral module number.
val	Byte to send
ticks_to_wait	Amount of time to wait for a start transmission to complete, specified as system time ticks. The TICKS_PER_SECOND macro can be used with a multiplier or dividor. Includes the time to receive an ack from the I2C device. If not specified, the default time of I2C_RX_TX_TIMEOUT will be used.

Returns

The state of the I2C bus: I2C Return Values

21.58.2.7 MultiChannel_I2CSendBuf()

```
uint8_t MultiChannel_I2CSendBuf (
            int moduleNum,
            uint8_t addr,
            puint8_t buf,
            int num,
            bool stop = true )
```

Send a buffer of bytes to an I2C device.

Sends an I2C start, the bytes in buf, and and finishes with our without an I2C stop (default is stop). If a stop is not issued, for purposes such as a restart, a stop must be called manually.

Parameters

moduleNum	I2C peripheral module number.
addr	I2C device address.
buf	The buffer to send.
num	The number of bytes in the buffer.
stop	Send a stop signal at the end of transmission. If not specified, the default value is true.

Returns

The state of the I2C bus: I2C Return Values

21.58.2.8 MultiChannel_I2CStart()

```
uint8_t MultiChannel_I2CStart (
            int moduleNum,
            uint8_t addr,
            bool Read_Not_Write,
            uint32_t ticks_to_wait = I2C_START_TIMEOUT )
```

Send an I2C start to an I2C device to begin communication.
Restarts communication with an I2C device after completion of a read or write command in which a stop was not issued. Enables communicate on bus again without giving up control as the I2C bus master.

Parameters

moduleNum	I2C peripheral module number.
addr	I2C device address.
Read_Not_Write	Specifies the next bus operation: true = read, false = write.
ticks_to_wait	Amount of time to wait for a start transmission to complete, specified as system time ticks. The TICKS_PER_SECOND macro can be used with a multiplier or dividor. Includes the time to receive an ack from the I2C device. If not specified, the default time of I2C_START_TIMEOUT will be used.

Returns

The state of the I2C bus: I2C Return Values

21.58.2.9 MultiChannel_I2CStop()

```
uint8_t MultiChannel_I2CStop (
            int moduleNum = DEFAULT_I2C_MODULE,
            uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT )
```

Issue an I2C stop terminate communication with an I2C device and release the bus.
Terminate any communication and release the I2C bus.

Parameters

moduleNum	I2C peripheral module number.
ticks_to_wait	Amount of time to wait for a start transmission to complete, specified as system time ticks. The TICKS_PER_SECOND macro can be used with a multiplier or dividor. Includes the time to receive an ack from the I2C device. If not specified, the default time of I2C_RX_TX_TIMEOUT will be used.

Returns

The state of the I2C bus: I2C Return Values

21.59 MIME Content Types

21.60 MODM7AE70

Modules

- Helper Macros
- MCAN Constants

Classes

- class mcanMODM7AE70::mcan_module

  MCAN Module Class.
- class mcanMODM7AE70::mcan_config

  MCAN configuration structure.
- class mcanMODM7AE70::CanRxMessage

  Class to hold received CAN messages.

21.60.1 Detailed Description

Control Area Network (MCAN) Low Level Driver for MODM7AE70
The Controller Area Network (MCAN) performs communication according to ISO 11898-1:2015 and to Bosch CANFD specification. Additional transceiver hardware is required for connection to the physical layer. All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM, as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core, as well as providing transmit status information.
Acceptance filtering is implemented by a combination of up to 128 filter elements, where each element can be configured as a range, as a bit mask, or as a dedicated ID filter.
There are two types of CAN identifiers that define different formats of the message frame, with the main difference being the identifier length. The two CAN protocol versions are:

- Version 2.0A, Normal or Standard CAN, supporting messages with 11-bit identifiers.
- Version 2.0B, Extended CAN, supporting messages with 29-bit identifiers consisting of an 11-bit base identifier for compatibility with Version 2.0A and an 18-bit extension identifier.

Note

Detailed information on the MCAN operation and features is located in the Microchip SAME70 Family Data Sheet located in your `\nburn\docs\Microchip` folder, chapter 49.

21.61 Multicast

Functions

- void RegisterMulticastFifo4 (IPADDR4 group, uint16_t dest_port, OS_FIFO *pfifo, int interface=0)

  Register to join a Multicast group.
- void UnregisterMulticastFifo4 (IPADDR4 group, uint16_t destination_port, int interface=0)

  Unregister from a Multicast group.
- void RegisterMulticastFifo6 (IPADDR group, uint16_t dest_port, OS_FIFO *pfifo, int interface=0)

  Register to join a Multicast group.
- void UnregisterMulticastFifo6 (IPADDR group, uint16_t destination_port, int interface=0)

  Unregister from a Multicast group.

21.61.1 Detailed Description

The NetBurner Multicast API

21.61.2 Function Documentation

21.61.2.1 RegisterMulticastFifo4()

```
void RegisterMulticastFifo4 (
            IPADDR4 group,
            uint16_t dest_port,
            OS_FIFO * pfifo,
            int interface = 0 )
```

Register to join a Multicast group.
Calling RegisterMulticastFifo() in dual stack mode will automatically select the correct IPv4/IPv6 function.
IGMP Multicast is a method for distributing UDP packets within a group of hosts and servers. The NetBurner Multicast functions extend the NetBurner UDP interface. Instead of RegisterUDPFifo(), the RegisterMulticastFifo() function is used to listen for Multicast UDP packets. To transmit Multicast packets, use the UDP Send function with a multicast IP address.

Parameters

group	The IP address of the group to join
dest_port	The port number of the group to join
pfifo	Pointer to the FIFO to store incoming packets
interface	Optional interface number. If not specified or 0, the first system interface will be used.

See also

UnregisterMulticastFifo4()

21.61.2.2 RegisterMulticastFifo6()

```
void RegisterMulticastFifo6 (
            IPADDR group,
            uint16_t dest_port,
            OS_FIFO * pfifo,
            int interface = 0 )
```

Register to join a Multicast group.
Calling RegisterMulticastFifo() in dual stack mode will automatically select the correct IPv4/IPv6 function.
IGMP Multicast is a method for distributing UDP packets within a group of hosts and servers. The NetBurner Multicast functions extend the NetBurner UDP interface. Instead of RegisterUDPFifo(), the RegisterMulticastFifo() function is used to listen for Multicast UDP packets. To transmit Multicast packets, use the UDP Send function with a multicast IP address.

Parameters

group	The IP address of the group to join
dest_port	The port number of the group to join
pfifo	Pointer to the FIFO to store incoming packets
interface	Optional interface number. If not specified or 0, the first system interface will be used.

See also

UnregisterMulticastFifo4()

21.61.2.3 UnregisterMulticastFifo4()

```
void UnregisterMulticastFifo4 (
            IPADDR4 group,
            uint16_t destination_port,
            int interface = 0 )
```

Unregister from a Multicast group.
Calling UnregisterMulticastFifo() in dual stack mode will automatically select the correct IPv4/IPv6 function.

Parameters

group	The IP address of the group to leave
destination_port	The port number of the group to leave
interface	Specifies the interface number, the default is 0

See also

RegisterMulticastFifo4()

21.61.2.4 UnregisterMulticastFifo6()

```
void UnregisterMulticastFifo6 (
            IPADDR group,
            uint16_t destination_port,
            int interface = 0 )
```

Unregister from a Multicast group.
Calling UnregisterMulticastFifo() in dual stack mode will automatically select the correct IPv4/IPv6 function.

Parameters

group	The IP address of the group to leave
destination_port	The port number of the group to leave
interface	Optional interface number. If not specified or 0, the first system interface will be used.

See also

RegisterMulticastFifo4()

21.62 Multichannel I2C Macros

I2C Macros.

Macros

- #define **I2C_SR_BUSY** (((0x20 & I2C_SR) == 0x20))

  Bus is busy (bit 5 of I2SR)
- #define **I2C_CR_SLAVE** (((0x20 & I2C_CR) == 0x00))

  Bus set as slave (bit 5 of I2CR)
- #define **I2C_SR_ARB_LOST** (((0x10 & I2C_SR) == 0x10))

Bus arbitration was lost (bit 4 of I2SR)

- #define **I2C_SR_ADRES_AS_SLAVE** (((0x40 & I2C_SR) == 0x40))

  Addressed as a slave (bit 6 of I2SR)
- #define **I2C_SR_SLAVE_TX** (((0x04 & I2C_SR) == 0x04))

  State of read/write bit of the received address command (bit 2 of I2SR)
- #define **I2C_CR_TX** (((0x10 & I2C_CR) == 0x10))

  Configured for transmit (bit 4 of I2CR)
- #define **I2C_SR_RX_ACK** (((0x01 & I2C_SR) == 0x00))

  Received a RX ACK after last transmit (bit 0 of I2SR)
- #define **I2C_CR_RX_ACK** (((0x08 & I2C_CR) == 0x00))

  Configured to RX Ack.
- #define **I2C_SET_NO_ACK** ((I2C_CR |= 0x08))

  Configure I2C module not to send a RX ACK (bit 3 of I2CR)
- #define **I2C_SET_ACK** ((I2C_CR &= 0xF7))

  Configure I2C module to send a RX ACK (bit 3 of I2CR)
- #define **I2C_SET_TX** ((I2C_CR |= 0x10))

  Configure I2C module to be in TX mode (bit 4 of I2CR)
- #define **I2C_SET_RX** ((I2C_CR &= 0xEF))

  Configure I2C module to be in RX mode (bit 4 of I2CR)
- #define **I2C_SET_REPEAT_START** ((I2C_CR |= 0x04))

  Configure I2C to send a repeated start signal.
- #define **I2C_CLR_ARB_LOST** ((I2C_SR &= 0xEF))

  Clear Arbitration lost error condition.

21.62.1 Detailed Description

I2C Macros.

21.63 Multichannel I2C Return Values

Function return values.

Macros

- #define **I2C_OK** (0)

  Last instruction terminated correctly.
- #define **I2C_NEXT_WRITE_OK** (1)

  I2C bus is OK for a write.
- #define **I2C_NEXT_READ_OK** (2)

  I2C bus is OK for a read.
- #define **I2C_MASTER_OK** (3)

  I2C finished transmission but still owns but (need to stop or restart)
- #define **I2C_TIMEOUT** (4)

  A timeout occurred while trying communicate on I2C bus.
- #define **I2C_BUS_NOT_AVAIL** (5)

  A timeout occurred while trying gain I2C bus control.
- #define **I2C_NOT_READY** (6)

  A read or write was attempted before I2C ready or during a slave transmission.
- #define **I2C_LOST_ARB** (7)

  Lost arbitration during start.
- #define **I2C_LOST_ARB_ADD** (8)

  Lost arbitration and then winner addressed our slave address.
- #define **I2C_NO_LINK_RX_ACK** (9)

  We are in Master TX mode and received no ACK from slave device, possibly during start.

21.63.1 Detailed Description

Function return values.

21.64 Multihome and VLAN

Functions

- int AddVlanInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, const char *ParentName)

  Add a VLAN interface with a Parent Name.
- int AddVlanInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, InterfaceBlock &parent)

  Add a VLAN interface with a Partent InterfaceBlock reference.
- int AddVlanInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, int root_if=0)

  Add a VLAN interface with an interface number.
- int AddInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, const char *ParentName)

  Add an interface with a Parent Name.
- int AddInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, InterfaceBlock &parent)

  Add an interface with a Partent InterfaceBlock reference.
- int AddInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, int root_if=0)

  Add an interface with an interface number.

21.64.1 Detailed Description

Create Multihome and VLAN Interfaces

21.64.2 Function Documentation

21.64.2.1 AddInterface() [1/3]

```
int AddInterface (
            IPADDR4 addr,
            IPADDR4 mask,
            IPADDR4 gateway,
            const char * ParentName )  [inline]
```

Add an interface with a Parent Name.

Parameters

addr	IP address of new interface
mask	IP mask
gateway	IP gateway
ParentName	Name of parent physical interface, such as Ethernet0.

returns Interface number of newly created interface

21.64.2.2 AddInterface() [2/3]

```
int AddInterface (
            IPADDR4 addr,
            IPADDR4 mask,
            IPADDR4 gateway,
            int root_if = 0 )  [inline]
```

Add an interface with an interface number.

Parameters

addr	IP address of new interface
mask	IP mask
gateway	IP gateway
root_if	Optional root physical interface number. If not specified the default is 0, for Ethernt 0.

returns Interface number of newly created interface

21.64.2.3 AddInterface() [3/3]

```
int AddInterface (
            IPADDR4 addr,
            IPADDR4 mask,
            IPADDR4 gateway,
            InterfaceBlock & parent )  [inline]
```

Add an interface with a Partent InterfaceBlock reference.

Parameters

addr	IP address of new interface
mask	IP mask
gateway	IP gateway
parent	Reference to parent physical interface of type InterfaceBlock.

returns Interface number of newly created interface

21.64.2.4 AddVlanInterface() [1/3]

```
int AddVlanInterface (
            IPADDR4 addr,
            IPADDR4 mask,
            IPADDR4 gateway,
            uint16_t vlan_tag,
            const char * ParentName )
```

Add a VLAN interface with a Parent Name.

Parameters

addr	IP address of new interface
mask	IP mask
gateway	IP gateway
vlan_tag	VLAN tag/id for interface
ParentName	Name of parent physical interface, such as Ethernet0.

returns Interface number of newly created VLAN interface

21.64.2.5 AddVlanInterface() [2/3]

```
int AddVlanInterface (
            IPADDR4 addr,
            IPADDR4 mask,
            IPADDR4 gateway,
            uint16_t vlan_tag,
            int root_if = 0 )
```

Add a VLAN interface with an interface number.

Parameters

addr	IP address of new interface
mask	IP mask
gateway	IP gateway
vlan_tag	VLAN tag/id for interface
root_if	Optional root physical interface number. If not specified the default is 0 for Ethernet 0.

returns Interface number of newly created VLAN interface

21.64.2.6 AddVlanInterface() [3/3]

```
int AddVlanInterface (
            IPADDR4 addr,
            IPADDR4 mask,
            IPADDR4 gateway,
            uint16_t vlan_tag,
            InterfaceBlock & parent )
```

Add a VLAN interface with a Partent InterfaceBlock reference.

Parameters

addr	IP address of new interface
mask	IP mask
gateway	IP gateway
vlan_tag	VLAN tag/id for interface
parent	Reference to parent physical interface of type InterfaceBlock

returns Interface number of newly created VLAN interface

21.65 NBRTOS Error Codes

Macros

- #define **OS_NO_ERR** 0

  No error.
- #define **OS_TIMEOUT** 10

  Timeout.
- #define **OS_MBOX_FULL** 20

  Mailbox full.
- #define **OS_Q_FULL** 30

  Queue full.
- #define **OS_Q_EXISTS** 31

  Queue already exists.
- #define **OS_PRIO_EXIST** 40

  Task priority number already exists.
- #define **OS_PRIO_INVALID** 41

  Invalid task priority number.
- #define **OS_SEM_ERR** 50

  Semaphore error.
- #define **OS_SEM_OVF** 51

  Semaphore count overflow.

- #define **OS_CRIT_ERR** 60

  Critical section error.
- #define **OS_NO_MORE_TCB** 70

  No Task Control Blocks (TCB) available to create task.

21.65.1 Detailed Description

NBRTOS function return codes

21.66 NBRTOS Real Time Operating System

Modules

- NBRTOS Error Codes
- NBRTOS Task Status

Classes

- class TickTimeout

  TickTimeout objects are used to facilitate sequential function calls with timeout parameters that need to indexed from an initial start time, and to prevent TimeTick rollover errors.
- struct OS_SEM

  Semaphores are used to control access to shared resources or or to communicate between tasks in a multithreaded system or with interrupt service routines. Semaphores can be 0 or 1, or they can be counting semaphores that increment and decreement based on calls to Pend() and Post() functions.
- struct OS_MBOX

  Mailboxes single value storage locations used to communicate between tasks.
- struct OS_Q

  A message queue is an object that enables tasks and interrupt service routines to pend and post pointer sized messages. The pointer values typically point to some type of object or structure that contains the actual message or data. A queue functions as a fixed size First In First Out (FIFO) storage for 32-bit void pointers that can be used for communication between tasks.
- struct os_fifo_el

  OS_FIFO element definition.
- struct OS_FIFO
- struct OS_CRIT

  An OS_CRIT object is used to establish critical sections of code that can only be run by one task at a time. Tasks that try to claim a critical section which is currently claimed by another task will stop and wait for that task to release the critical section before continuing execution.
- struct OS_FLAGS

  OSFlags enables a function or task to pend on multiple flags or events.
- class OSLockObj

  A simple wrapper class that helps use OS locks effectively.
- class OSCriticalSectionObj

  A simple wrapper class that helps utilize OS_CRIT objects more effectively.
- class OSLockAndCritObj

  A simple wrapper class that helps utilize OS_CRIT objects to lock tasks and enter critical sections more effectively.
- class OSSpinCrit

  A simple wrapper class that uses an OS_CRIT object to try and claim a critical section, and will continue the attempt until it is able to do so.
- class USERCritObj

  User critial section object class.
- class NBRtosInitObj

  A simple class to derive from if you are creating tasks that are constructed at global scope and need to do RTOS initalization.

Macros

- #define **WAIT_FOREVER** 0

  Parameter macro used for timeout parameters that have a 0 value and wait forever.
- #define OSSimpleTaskCreatewName(x, p, n)

  Simpler form of creating a new task. Will automatically allocate the default task stack size.
- #define OSSimpleTaskCreateLambda(p, n, f) LambdaTask2(p,n,[ ](void * pv)f,__COUNTER__)

  This macro functions the same as OSTaskCreatewName().

Typedefs

- typedef struct os_fifo_el OS_FIFO_EL

  OS_FIFO element definition.

Functions

- void OSFlagSet (OS_FLAGS *flags, uint32_t bits_to_set)

  This function sets the corresponding bits asserted in `bits_to_set` of an OS_FLAGS object pointed to by `*flags`.
- void OSFlagClear (OS_FLAGS *flags, uint32_t bits_to_clr)

  *This function clears the bits asserted in `bits_to_clr` of an OS_FLAGS object pointed to by *flags..*
- uint8_t OSFlagPendAny (OS_FLAGS *flags, uint32_t bit_mask, uint16_t timeout)

  This function waits a number of time ticks specified by `timeout` until any of the flags indicated by `bit_mask` are set.
- uint8_t OSFlagPendAnyNoWait (OS_FLAGS *flags, uint32_t bit_mask)

  This function immediately checks to see if any of the flag bits indicated by `bit_mask` are set; it does not wait.
- uint8_t OSFlagPendAll (OS_FLAGS *flags, uint32_t bit_mask, uint16_t timeout)

  *This function waits a number of time ticks specified by `timeout` until **all** the flags indicated by `bit_mask` are set.*
- uint8_t OSFlagPendAllNoWait (OS_FLAGS *flags, uint32_t bit_mask)

  *This function immediately checks to see if **all** the flag bits indicated by `bit_mask` are set; it does not wait.*
- uint32_t OSFlagState (OS_FLAGS *flags)

  This function returns the current values of the flags stored in the OS_FLAGS object structure.
- uint8_t OSTaskCreatewName (void(*task)(void *dptr), void *data, void *pstktop, void *pstkbot, uint8_t prio, const char *name)

  Create a new task.
- void OSTimeWaitUntil (uint32_t systemTickValue)

  Delay the task until the specified value of the system timer tick. The number of system ticks per second is defined by the constant: `TICKS_PER_SECOND` in `<nburn_install>/nbrtos/include/constants.h`. The default value is 20 ticks per second.
- void OSTimeDly (uint32_t to_count)

  Delay the task until the specified value of the system timer ticks. The number of system ticks per second is defined by the constant: `TICKS_PER_SECOND` in `<nburn_install>/nbrtos/include/constants.h`. The default value is 20 ticks per second.
- void OSTaskDelete (void)

  This function deletes the current calling task, but we do not recommend the use of this function because it can cause memory leaks.
- uint8_t OSChangePrio (uint32_t newp)

  Set the priority of the calling task.
- void OSSetName (const char *cp)

  Set the name of the calling task.
- void OSLock (void)

  Calling the OSLock function will prevent the OS from changing tasks.
- void OSUnlock (void)

  This function unlocks the OS.

- uint8_t OSSemInit (OS_SEM *psem, long value)

  Initializes a semaphore.
- uint8_t OSSemPost (OS_SEM *psem)

  Increases the value of the semaphore by one. ***Note:*** *If any higher priority tasks were waiting on the semaphore - it releases them.*
- uint8_t OSSemPend (OS_SEM *psem, uint16_t timeout)

  Wait timeout ticks for the value of the semaphore to be non zero. ***Note:*** *A timeout value of 0 (zero) waits forever.*
- uint8_t OSSemPendNoWait (OS_SEM *psem)

  OSSemPendNoWait() is identical to OSSemPend(), but it does not wait.
- uint8_t OSMboxInit (OS_MBOX *pmbox, void *msg)

  This function is used to initialize an `OS_MBOX` structure.
- uint8_t OSMboxPost (OS_MBOX *pmbox, void *msg)

  This function posts a message to a Mail box.
- void * OSMboxPend (OS_MBOX *pmbox, uint16_t timeout, uint8_t *err)

  Wait timeout ticks for some other task to post to the Mailbox.
- void * OSMboxPendNoWait (OS_MBOX *pmbox, uint8_t *err)

  OSMboxPendNoWait() is identical to OSMboxPend(), but it does ***not*** *wait.*
- uint8_t OSQInit (OS_Q *pq, void **start, uint8_t size)

  A queue functions as a fixed size FIFO for communication between tasks. This function initializes an `OS_Q` structure.
- uint8_t OSQPost (OS_Q *pq, void *msg)

  This function posts a message to a Queue.
- uint8_t OSQPostFirst (OS_Q *pq, void *msg)

  This function posts a message like `OSQPost`, but posts the message at the head of the queue.
- uint8_t OSQPostUnique (OS_Q *pq, void *msg)

  This function posts a message like `OSQPost`, but only if the message isn't already in the queue The function performs a brute force check to see if the message is already in the queue.
- uint8_t OSQPostUniqueFirst (OS_Q *pq, void *msg)

  This function posts a message like `OSQPostFirst`, but only if the message isn't already in the queue The function performs a brute force check to see if the message is already in the queue.
- void * OSQPend (OS_Q *pq, uint16_t timeout, uint8_t *err)

  Wait timeout ticks for another task to post to the queue.
- void * OSQPendNoWait (OS_Q *pq, uint8_t *err)

  OSQPendNoWait() is identical to the OSQPend() function but it does not wait.
- uint8_t OSFifoInit (OS_FIFO *pFifo)

  Initialize a FIFO, which is used to pass structures from one task to another.
- uint8_t OSFifoPost (OS_FIFO *pFifo, OS_FIFO_EL *pToPost)

  This function posts to a FIFO.
- uint8_t OSFifoPostFirst (OS_FIFO *pFifo, OS_FIFO_EL *pToPost)

  This function is identical to OSFifoPost(), but the element posted is put at the beginning of the FIFO list.
- OS_FIFO_EL * OSFifoPend (OS_FIFO *pFifo, uint16_t timeout)

  This function pends on a FIFO.
- OS_FIFO_EL * OSFifoPendNoWait (OS_FIFO *pFifo)

  This function is identical to the OSFifoPen() function, but it does ***not*** *wait.*
- uint8_t OSCritInit (OS_CRIT *pCrit)

  This function initializes the critical section.
- uint8_t OSCritEnter (OS_CRIT *pCrit, uint16_t timeout)

  This function tries to enter or claim the critical section.
- uint8_t OSCritEnterNoWait (OS_CRIT *pCrit)

  This function tries to enter or claim the critical section.
- uint8_t OSCritLeave (OS_CRIT *pCrit)

  This function releases the critical section.

- uint8_t **OSTaskID** (void)

  Returns the current task's priority.
- const char * **OSTaskName** ()

  Returns the current task's name.
- void OSChangeTaskDly (uint16_t task_prio, uint32_t to_count)

  This function allows the User to modify the timeout delay for a task that is waiting.
- void **OSDumpStack** (void)

  Dump the task stack to the stdout.
- void OSDumpTCBStacks (void)

  This function dumps information about the UCOS stacks and tasks to stdout. This function is useful for debugging.
- void OSDumpTasks (void)

  Dump the state and call stack for every task to stdout. This function is useful for debugging.
- void OSStartTaskDumper (uint8_t prio, uint32_t reportInterval)

  This function creates a task that calls OSDumpTasks() at the specified system time tick interval. The task is intended for use when debugging run status of multiple tasks.
- void ShowTaskList (void)

  This functions dumps the current RTOS task states to stdio.
- bool OS_CRIT::OwnedByCurTask ()

  Check if critical section owned by the current task.

21.66.1 Detailed Description

The NetBurner Real-Time OS

21.66.2 Macro Definition Documentation

21.66.2.1 OSSimpleTaskCreateLambda

```
#define OSSimpleTaskCreateLambda(
              p,
              n,
              f ) LambdaTask2(p,n,[]( void * pv)f,__COUNTER__)
```

This macro functions the same as OSTaskCreatewName().

Parameters

p	The priority for this new task (`OS_MAX_PRIOS` is lowest priority and 1 is highest). Look in `<nburn_install>/nbrtos/include/constants.h` to see which priorities are used by the OS.
n	The optional name of the task.
f	The function to run as the task.

Example:

```
OSSimpleTaskCreateLambda(MAIN_PRIO-1,"MyLambdaTask",{ Code to run goes here });
```

See also

OSTaskCreatewName(), OSSimpleTaskCreatewName()

21.66.2.2 OSSimpleTaskCreatewName

```
#define OSSimpleTaskCreatewName(
              x,
```

```
            p,
            n )
```

Value:

```
{                                                                                   \
    static uint32_t func_##x_Stk[USER_TASK_STK_SIZE] __attribute__((aligned(4)));   \
    OSTaskCreatewName(x, NULL, (void *)&func_##x_Stk[USER_TASK_STK_SIZE], (void *)func_##x_Stk, p, n); \
}
```

Simpler form of creating a new task. Will automatically allocate the default task stack size.

Parameters

x	The address of the function where this task will start executing.
p	The priority for this new task (OS_MAX_PRIOS is lowest priority and 1 is highest). Look in <nburn_install>/nbrtos/include/constants.h to see which priorities are used by the OS.
n	The optional name of the task.

See also

OSTaskCreatewName(), OSTaskDelete(), OSChangePrio()

21.66.3 Typedef Documentation

21.66.3.1 OS_FIFO_EL

```
typedef struct os_fifo_el OS_FIFO_EL
```

OS_FIFO element definition.
This struture is a union of a pointer to the next OS_FIFO and a byte pointer to the same data.

21.66.4 Function Documentation

21.66.4.1 OSChangePrio()

```
uint8_t OSChangePrio (
            uint32_t newp )
```

Set the priority of the calling task.

Note

There can only be one task at each priority level. Task priorities can range from 1 to OS_MAX_PRIOS, where OS_MAX_PRIOS is the lowest priority level and 1 is highest priority level. Priorities 1-4 and the NetBurner system priority levels are reserved as described below. The recommended user priority levels for your application are in the range of 46 to OS_MAX_PRIOS-1. This avoids any conflicts with network communications.

System priorities are defined in <nburn_install>/nbrtos/include/constants.h for all platforms.

Parameters

newp	The new priority of the calling task

Return values

OS_NO_ERR	If successful
OS_PRIO_EXIST	If the requested priority already exists

See also

NBRTOS Error Codes, OSTaskCreatewName()

21.66.4.2 OSChangeTaskDly()

```
void OSChangeTaskDly (
            uint16_t task_prio,
            uint32_t to_count )  [inline]
```

This function allows the User to modify the timeout delay for a task that is waiting.

Warning

Use of this function is discouraged.

Parameters

task_prio	The task's priority
to_count	The new number of ticks to delay

See also

OSTimeDly()

21.66.4.3 OSCritEnter()

```
uint8_t OSCritEnter (
            OS_CRIT * pCrit,
            uint16_t timeout )  [inline]
```

This function tries to enter or claim the critical section.

Parameters

pCrit	A pointer to the critical section.
timeout	How many time ticks do we want to wait for this critical section? **Note:** A timeout of 0 (**zero**) waits forever.

Returns

OS_NO_ERR - If we were successful in claiming the critical section, or if our task owns it

OS_TIMEOUT - If we were unable to claim the section

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_CRIT for current usage.

21.66.4.4 OSCritEnterNoWait()

```
uint8_t OSCritEnterNoWait (
            OS_CRIT * pCrit )  [inline]
```

This function tries to enter or claim the critical section.

Parameters

pCrit	A pointer to the critical section.

Returns

OS_NO_ERR - If we were successful in claiming the critical section, or if our task owns it

OS_TIMEOUT - If we were unable to claim the section

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_CRIT for current usage.

21.66.4.5 OSCritInit()

```
uint8_t OSCritInit (
            OS_CRIT * pCrit )  [inline]
```

This function initializes the critical section.

Parameters

pCrit	A pointer to the critical section.

Returns

OS_NO_ERR - If successful

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_CRIT for current usage.

21.66.4.6 OSCritLeave()

```
uint8_t OSCritLeave (
            OS_CRIT * pCrit )  [inline]
```

This function releases the critical section.

Parameters

pCrit	A pointer to the critical section we want to leave/release.

Returns

OS_NO_ERR - If we were successful in releasing the critical section

OS_CRIT_ERR - If we are trying to release a critical section that we do not own

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_CRIT for current usage.

21.66.4.7 OSDumpTasks()

```
void OSDumpTasks (
            void  )
```

Dump the state and call stack for every task to stdout. This function is useful for debugging.

Note

This function can only be called if NBRTOS_STACKCHECK is defined in predef.h.

See also

OSDumpTCBStacks()

21.66.4.8 OSDumpTCBStacks()

```
void OSDumpTCBStacks (
            void  )
```

This function dumps information about the UCOS stacks and tasks to stdout. This function is useful for debugging.

Note

This function can only be called if NBRTOS_STACKCHECK is defined in predef.h.

See also

OSDumpTasks()

21.66.4.9 OSFifoInit()

```
uint8_t OSFifoInit (
            OS_FIFO * pFifo )  [inline]
```

Initialize a FIFO, which is used to pass structures from one task to another.

Parameters

pFifo	A pointer to an OS_FIFO structure.

Returns

OS_NO_ERR - If successful

OS_CRIT_ERR - If pFifo is NULL

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_FIFO for current usage.

21.66.4.10 OSFifoPend()

```
OS_FIFO_EL * OSFifoPend (
            OS_FIFO * pFifo,
            uint16_t timeout )  [inline]
```

This function pends on a FIFO.

Parameters

pFifo	A pointer to an OS_FIFO structure.
timeout	The number of ticks to wait on the FIFO.

Returns

A pointer to the posted structure if successful, or NULL if it timed out.

Deprecated This function is now deprecated. Please see OS_FIFO for current usage.

21.66.4.11 OSFifoPendNoWait()

```
OS_FIFO_EL * OSFifoPendNoWait (
            OS_FIFO * pFifo )  [inline]
```

This function is identical to the OSFifoPen() function, but it does **not** wait.

Parameters

pFifo	A pointer to an OS_FIFO structure.

Returns

A pointer to the posted structure if successful, or NULL if it timed out.

21.66.4.12 OSFifoPost()

```
uint8_t OSFifoPost (
            OS_FIFO * pFifo,
            OS_FIFO_EL * pToPost )  [inline]
```

This function posts to a FIFO.

Parameters

pFifo	A pointer to an OS_FIFO structure.
pToPost	A pointer to the user's structure cast as an OS_FIFO_EL to be posted to the FIFO.

Returns

OS_NO_ERR - If successful

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_FIFO for current usage.

21.66.4.13 OSFifoPostFirst()

```
uint8_t OSFifoPostFirst (
            OS_FIFO * pFifo,
            OS_FIFO_EL * pToPost )  [inline]
```

This function is identical to OSFifoPost(), but the element posted is put at the beginning of the FIFO list.

Parameters

pFifo	A pointer to an OS_FIFO structure.
pToPost	A pointer to the user's structure cast as an OS_FIFO_EL to be posted to the FIFO.

Returns

OS_NO_ERR - If successful

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_FIFO for current usage.

21.66.4.14 OSFlagClear()

```
void OSFlagClear (
            OS_FLAGS * flags,
            uint32_t bits_to_clr )  [inline]
```

This function clears the bits asserted in bits_to_clr of an OS_FLAGS object pointed to by *flags..

Parameters

flags	A pointer to the OS_FLAGS object to be configured.
bits_to_clr	A bit or set of bits to be cleared.

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.15 OSFlagPendAll()

```
uint8_t OSFlagPendAll (
            OS_FLAGS * flags,
            uint32_t bit_mask,
            uint16_t timeout )  [inline]
```

This function waits a number of time ticks specified by timeout until **all** the flags indicated by bit_mask are set.

Parameters

flags	A pointer to the OS_FLAGS object with the desired flag bits.
bit_mask	A bit or set of bits to wait on.
timeout	Number of time ticks to wait for all specified flag bits to be set.

Return values

OS_NO_ERR	If the flags condition is satisfied
OS_TIMEOUT	If the timeout expired

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.16 OSFlagPendAllNoWait()

```
uint8_t OSFlagPendAllNoWait (
            OS_FLAGS * flags,
            uint32_t bit_mask )  [inline]
```

This function immediately checks to see if **all** the flag bits indicated by `bit_mask` are set; it does not wait.

Parameters

flags	A pointer to the OS_FLAGS object with the desired flag bits.
bit_mask	A bit or set of bits to wait on.

Return values

OS_NO_ERR	All flags indicated by `bit_mask` are set.
OS_TIMEOUT	Not all of the flags indicated by `bit_mask` are set.

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.17 OSFlagPendAny()

```
uint8_t OSFlagPendAny (
            OS_FLAGS * flags,
            uint32_t bit_mask,
            uint16_t timeout )  [inline]
```

This function waits a number of time ticks specified by `timeout` until any of the flags indicated by `bit_mask` are set.

Parameters

flags	A pointer to the OS_FLAGS object with the desired flag bits.
bit_mask	A bit or set of bits to wait on.
timeout	Number of time ticks to wait for all specified flag bits to be set.

Return values

OS_NO_ERR	At least one of the flag bits are set before `timeout` expires.
OS_TIMEOUT	None of the flag bits are set before `timeout` expires.

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.18 OSFlagPendAnyNoWait()

```
uint8_t OSFlagPendAnyNoWait (
            OS_FLAGS * flags,
            uint32_t bit_mask )  [inline]
```

This function immediately checks to see if any of the flag bits indicated by `bit_mask` are set; it does not wait.

Parameters

flags	A pointer to the OS_FLAGS object with the desired flag bits.

Parameters

bit_mask	A bit or set of bits to wait on.

Return values

OS_NO_ERR	At least one of the flags indicated by `bit_mask` are set.
OS_TIMEOUT	None of the flags indicated by `bit_mask` are set.

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.19 OSFlagSet()

```
void OSFlagSet (
            OS_FLAGS * flags,
            uint32_t bits_to_set )  [inline]
```

This function sets the corresponding bits asserted in `bits_to_set` of an OS_FLAGS object pointed to by `*flags`.

Parameters

flags	A pointer to the OS_FLAGS object to be configured.
bits_to_set	A bit or set of bits to be set.

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.20 OSFlagState()

```
uint32_t OSFlagState (
            OS_FLAGS * flags )  [inline]
```

This function returns the current values of the flags stored in the OS_FLAGS object structure.

Parameters

flags	A pointer to the OS_FLAGS object whose flag states are to be returned.

Returns

The state of the OS_FLAGS object.

Deprecated This function is now deprecated. Please see OS_FLAGS for current usage.

21.66.4.21 OSLock()

```
void OSLock (
            void  )
```

Calling the OSLock function will prevent the OS from changing tasks.
This is used to protect critical variables that must be accessed one task at a time. Use the OSUnlock function to release your lock. **Important:** You must call OSUnlock() once for each call to OSLock.

Warning: Do not keep a task locked for a long period of time, or the performance of the network subsystem will degrade, and eventually loose packets. For example, use of I/O is not recommended inside an OSLock protected block.

See also

OSUnlock(), OSLockObj()

21.66.4.22 OSMboxInit()

```
uint8_t OSMboxInit (
            OS_MBOX * pmbox,
            void * msg )  [inline]
```

This function is used to initialize an OS_MBOX structure.

Parameters

pmbox	A pointer to the OS_MBOX structure to initialize.
msg	The initial mail box message (NULL) for none.

Returns

OS_NO_ERR - If successful

OS_CRIT_ERR - If pmbox is NULL

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_MBOX for current usage.

21.66.4.23 OSMboxPend()

```
void * OSMboxPend (
            OS_MBOX * pmbox,
            uint16_t timeout,
            uint8_t * err )  [inline]
```

Wait timeout ticks for some other task to post to the Mailbox.

Parameters

	pmbox	A pointer to the OS_MBOX structure.
	timeout	The number of time ticks to wait.
out	*err*	A variable to receive the result code (OS_NO_ERR if successful or OS_TIMEOUT if it fails).

Returns

The posted message if successful, or NULL if timed out.

Deprecated This function is now deprecated. Please see OS_MBOX for current usage.

21.66.4.24 OSMboxPendNoWait()

```
void * OSMboxPendNoWait (
```

```
            OS_MBOX * pmbox,
            uint8_t * err )  [inline]
```

OSMboxPendNoWait() is identical to OSMboxPend(), but it does **not** wait.

Parameters

pmbox	A pointer to the OS_MBOX structure.
err	A variable to receive the result code (OS_NO_ERR if successful or OS_TIMEOUT if it fails).

Returns

The posted message if successful, or NULL if it fails.

Deprecated This function is now deprecated. Please see OS_MBOX for current usage.

21.66.4.25 OSMboxPost()

```
uint8_t OSMboxPost (
            OS_MBOX * pmbox,
            void * msg )  [inline]
```

This function posts a message to a Mail box.

Parameters

pmbox	A pointer to the OS_MBOX structure.
msg	The message to post.

Returns

OS_NO_ERR - If successful

OS_MBOX_FULL - If the mailbox is full

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_MBOX for current usage.

21.66.4.26 OSQInit()

```
uint8_t OSQInit (
            OS_Q * pq,
            void ** start,
            uint8_t size )  [inline]
```

A queue functions as a fixed size FIFO for communication between tasks. This function initializes an OS_Q structure.

Parameters

pq	A pointer to the OS_Q structure.
start	A pointer to an array of (void *) pointers to hold queue messages.
size	The number of pointers in the Q data storage area.

Returns

OS_NO_ERR - If Successful

OS_CRIT_ERR - If start or pq is NULL

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.27 OSQPend()

```
void * OSQPend (
            OS_Q * pq,
            uint16_t timeout,
            uint8_t * err )  [inline]
```

Wait timeout ticks for another task to post to the queue.

Parameters

	pq	A pointer to the OS_Q structure.
	timeout	The number of ticks to wait.
out	*err*	A variable to receive the result code, either OS_NO_ERR on receiving a posted message, or OS_TIMEOUT on a timeout.

Returns

The posted message, or NULL if the function failed

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.28 OSQPendNoWait()

```
void * OSQPendNoWait (
            OS_Q * pq,
            uint8_t * err )  [inline]
```

OSQPendNoWait() is identical to the OSQPend() function but it does not wait.

Parameters

	pq	A pointer to the OS_Q structure.
out	*err*	A variable to receive the result code, either OS_NO_ERR on receiving a posted message, or OS_TIMEOUT on a timeout.

Returns

The posted message, or NULL if the function failed

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.29 OSQPost()

```
uint8_t OSQPost (
```

```
            OS_Q * pq,
            void * msg )  [inline]
```

This function posts a message to a Queue.

Parameters

pq	A pointer to the OS_Q structure.
msg	The message to be posted to the queue.

Returns

OS_NO_ERR - If Successful

OS_Q_FULL - If the queue is full and has no more room

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.30 OSQPostFirst()

```
uint8_t OSQPostFirst (
            OS_Q * pq,
            void * msg )  [inline]
```

This function posts a message like `OSQPost`, but posts the message at the head of the queue.

Parameters

pq	A pointer to the OS_Q structure.
msg	The message to post at the head of the queue.

Returns

OS_NO_ERR - If Successful

OS_Q_FULL - If the queue is full and has no more room

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.31 OSQPostUnique()

```
uint8_t OSQPostUnique (
            OS_Q * pq,
            void * msg )  [inline]
```

This function posts a message like `OSQPost`, but only if the message isn't already in the queue The function performs a brute force check to see if the message is already in the queue.

Parameters

pq	A pointer to the OS_Q structure.
msg	The message to post at the head of the queue.

Returns

OS_NO_ERR - If Successful

OS_Q_FULL - If the queue is full and has no more room

OS_Q_EXISTS - If the message already exists in the queue

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.32 OSQPostUniqueFirst()

```
uint8_t OSQPostUniqueFirst (
            OS_Q * pq,
            void * msg )  [inline]
```

This function posts a message like `OSQPostFirst`, but only if the message isn't already in the queue The function performs a brute force check to see if the message is already in the queue.

Parameters

pq	A pointer to the OS_Q structure.
msg	The message to post at the head of the queue.

Returns

OS_NO_ERR - If Successful

OS_Q_FULL - If the queue is full and has no more room

OS_Q_EXISTS - If the message already exists in the queue

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_Q for current usage.

21.66.4.33 OSSemInit()

```
uint8_t OSSemInit (
            OS_SEM * psem,
            long value )  [inline]
```

Initializes a semaphore.

Parameters

psem	A pointer to the OS_SEM structure to initialize.
value	The initial count value for the semaphore.

Returns

OS_NO_ERR - If successful

OS_SEM_ERR - If value is $<$ 0 (zero), it cannot initialize

OS_CRIT_ERR - If psem is NULL

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_SEM for current usage.

21.66.4.34 OSSemPend()

```
uint8_t OSSemPend (
            OS_SEM * psem,
            uint16_t timeout )  [inline]
```

Wait timeout ticks for the value of the semaphore to be non zero. **Note:** A timeout value of 0 (zero) waits forever.

Parameters

psem	A pointer to the OS_SEM structure.
timeout	The number of time ticks to wait

Returns

OS_NO_ERR - If successful

OS_TIMEOUT - If the function timed out or if the NoWait function failed

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_SEM for current usage.

21.66.4.35 OSSemPendNoWait()

```
uint8_t OSSemPendNoWait (
            OS_SEM * psem )  [inline]
```

OSSemPendNoWait() is identical to OSSemPend(), but it does not wait.

Parameters

psem	A pointer to the OS_SEM structure

Returns

OS_NO_ERR - If successful

OS_TIMEOUT - If it fails

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_SEM for current usage.

21.66.4.36 OSSemPost()

```
uint8_t OSSemPost (
            OS_SEM * psem )  [inline]
```

Increases the value of the semaphore by one. **Note:** If any higher priority tasks were waiting on the semaphore - it releases them.

Parameters

psem	A pointer to the OS_SEM structure.

Returns

OS_NO_ERR - If successful

OS_SEM_OVF - If the value of the semaphore overflows

See also

NBRTOS Error Codes

Deprecated This function is now deprecated. Please see OS_SEM for current usage.

21.66.4.37 OSSetName()

```
void OSSetName (
            const char * cp )
```

Set the name of the calling task.

Parameters

cp	Character pointer to new task name

21.66.4.38 OSStartTaskDumper()

```
void OSStartTaskDumper (
            uint8_t prio,
            uint32_t reportInterval )
```

This function creates a task that calls OSDumpTasks() at the specified system time tick interval. The task is intended for use when debugging run status of multiple tasks.

Note

This function can only be called if NBRTOS_STACKCHECK is defined in predef.h.

See also

OSDumpTasks()

21.66.4.39 OSTaskCreatewName()

```
uint8_t OSTaskCreatewName (
            void(*)(void *dptr) task,
            void * data,
            void * pstktop,
            void * pstkbot,
            uint8_t prio,
            const char * name )
```

Create a new task.

You must allocate storage for the task stack that this new task will use, and it must be 4 byte aligned.

Task priorities can range from 1 to OS_MAX_PRIOS, where OS_MAX_PRIOS is the lowest priority level and 1 is highest priority level. The recommended user priority levels for your application are in the range of 46 to OS↩_MAX_PRIOS-1. This avoids any conflicts with network communications. The maximum number of tasks your

application can run is defined by OS_MAX_TASKS. Both OS_MAX_PRIOS and OS_MAX_TASKS are defined in "\nburn\nbrtos\include\constants.h".

Note

The system can have only one task at each priority

It is good practice to check the function return value to ensure the task was created successfully.

Parameters

task	The address of the function where this task will start executing.
data	The data to pass to the task function.
pstktop	The highest address of the stack space.
pstkbot	The lowest address of the stack space.
prio	The priority for this new task (OS_MAX_PRIOS is lowest priority and 1 is highest). Look in <nburn_install>/nbrtos/include/constants.h to see which priorities are used by the OS.
name	The name of the task

Return values

OS_NO_ERR	If successful
OS_PRIO_EXIST	If the requested priority already exists, NBRTOS Error Codes

See also

OSTaskDelete(), OSChangePrio(), NBRTOS Error Codes

21.66.4.40 OSTaskDelete()

```
void OSTaskDelete (
            void  )
```

This function deletes the current calling task, but we do not recommend the use of this function because it can cause memory leaks.
The preferred method for terminating a task is to set a flag or semaphore that the task is listening for. The flag can then be set by an outside task, which enables the task to be deleted to free any resources and terminate gracefully by simply returning.

See also

OSTaskCreatewName(), OSChangePrio()

21.66.4.41 OSTimeDly()

```
void OSTimeDly (
            uint32_t to_count )  [inline]
```

Delay the task until the specified value of the system timer ticks. The number of system ticks per second is defined by the constant: TICKS_PER_SECOND in <nburn_install>/nbrtos/include/constants.h. The default value is 20 ticks per second.

Note

The TICKS_PER_SECOND macro is a great way to make code more readable, and will also adjust if the default of 20 ticks/second is changed.

Parameters

to_count	The number of system ticks to delay.

See also

OSChangeTaskDly(), OSTimeWaitUntil()

21.66.4.42 OSTimeWaitUntil()

```
void OSTimeWaitUntil (
            uint32_t systemTickValue )
```

Delay the task until the specified value of the system timer tick. The number of system ticks per second is defined by the constant: `TICKS_PER_SECOND` in `<nburn_install>/nbrtos/include/constants.h`. The default value is 20 ticks per second.

Parameters

systemTickValue	The system time tick value to wait for

```
uint32_t Now = TimeTick;
while(1)
{
   Now += 60 * TICKS_PER_SECOND;
   OSTimeWaitUntil(Now);
   // Do whatever every 60 seconds, no mater how long this takes to run the interval will have a consistant
      spacing
}
```

See also

OSChangeTaskDly(), OSTimeDly()

21.66.4.43 OSUnlock()

```
void OSUnlock (
            void  )
```

This function unlocks the OS.
Important: You must call OSUnlock() once for each call to OSLock().

See also

OSLock(), OSLockObj()

21.66.4.44 OwnedByCurTask()

```
bool OS_CRIT::OwnedByCurTask ( )  [inline]
```

Check if critical section owned by the current task.
Check if current task owns the critical section.

Return values

true	if owned
false	if not owned

21.66.4.45 ShowTaskList()

```
void ShowTaskList (
            void  )
```

This functions dumps the current RTOS task states to stdio.
The output takes on multiple lines of the following format for each logged state:

```
at t= [T] [Message]
```

Followed by a tally of the number of task states logged since system start:

```
Total messages:  [N]
```

`[T]` represents the number of ticks in hexadecimal since system start; `[N]` represents the number of task state messages in decimal logged since system start; `[Message]` represents one of the output messages listed in the below table.

Message	Description
`Wait for Semaphore`	Task is asleep and pending for semaphore
`Wake from Semaphore`	Task gets a semaphore and wakes up
`Task locked`	Task becomes locked
`Task lock++`	Task gets an added nested lock
`Task lock-`	Task get a nested lock unlocked
`Task unlocked`	Task becomes completely unlocked
`Task priority changed`	The task's priority level is changed
`Unknown flag [F]`	The flag value defining the task's state is undefined
`Switched to Task [P]`	Task priority `[P]` (in decimal) gets control
`Switched to Task [P] PC=[X]`	Task priority `[P]` gets control with the program counter containing the address `[X]` (in hexadecimal) of the instruction being executed

Note: Usage of this function is valid only when defining `NBRTOS_TASKLIST` in debug mode. In order to enable this macro definition, it must be uncommented in `\Nburn\include\predef.h`, followed by rebuilding the system files to incorporate the modification. Attempting to load a compiled non-debug application image with the macro defined will cause a trap error.

21.67 NBRTOS Task Status

Macros

- #define **OS_STAT_RDY** 0x00

  Ready to run.
- #define **OS_STAT_MBOX** 0x01

  Pending on mailbox.
- #define **OS_STAT_SEM** 0x02

  Pending on semaphore.
- #define **OS_STAT_Q** 0x04

  Pending on queue.
- #define **OS_STAT_FIFO** 0x08

  Pending on FIFO.
- #define **OS_STAT_CRIT** 0x10

Pending on Critical Section.

- #define **OS_STAT_DELAY** 0x20

  Reserved.
- #define **OS_STAT_RES4** 0x40

  Reserved.
- #define **OS_STAT_RES5** 0x80

  Reserved.

21.67.1 Detailed Description

Values for the current state of a running task

21.68 NBString - NetBurner String Class

Classes

- class NBString

  Lightweight alternative to C++ CString class.

21.68.1 Detailed Description

The NetBurner lightweight String Class alternative to Cstring.

21.69 NBUpdate Function Return Values

Macros

- #define **NBUP_ERR_NO_ERR** 0

  No errors.
- #define **NBUP_ERR_BAD_SOCK** -1

  File descriptor socket is invlaid.
- #define **NBUP_ERR_TIMEOUT** -2

  Timeout.
- #define **NBUP_ERR_TOO_LARGE** -3

  Application update record is too large.
- #define **NBUP_ERR_WRONG_PLAT** -4

  Device platform name does not match.
- #define **NBUP_ERR_BAD_PROG** -5

  Programming failed, invalid checksum.
- #define **NBUP_ERR_BAD_AUTH** -6

  Application update record authentication failed.

21.69.1 Detailed Description

21.70 NetBurner MQTT Client implementation

Functions

- void UserMain (void *pd)

  This example will provide a menu through the serial debug port that enables you to set/clear static IP settings, and start/stop the DHCP Client service.

21.70.1 Detailed Description

The NetBurner MQTT Client implementation

21.70.2 Function Documentation

21.70.2.1 UserMain()

```
void UserMain (
            void * pd )
```

This example will provide a menu through the serial debug port that enables you to set/clear static IP settings, and start/stop the DHCP Client service.
UserMain
Main entry point for the example
UserMain
Main entry point of the example
UserMain
Main entry point to the example
UserMain
Main entry point to the program
UserMain
The main task
UserMain, the entry point of the example
UserMain
Main entry point of program
UserMain
Main entry point of example
UserMain
Main entry point for example
This example will provide a menu through the serial debug port that enables you to set/clear static IP settings, and start/stop the DHCP Client service.
UserMain Task
UserMain Task This is the first task to be executed and will create the UDP Reader Task.
UserMain This is the first task to be executed and will create the UDP Reader Task.
This example will provide a menu through the serial debug port that enables you to set/clear static IP settings, and start/stop the DHCP Client service.
Main entry point for the example
UserMain
Main entry point for the program
UserMain
Main entry point of program.

21.71 Network Interfaces

Network Interface Functions.

Modules

- IPADDR4 Functions

  Network Interface functions with IPADDR4 return types.
- Low Level Processing (NetDoRx)

  Callback functions to add custom Ethenet handlers and pass Ethernet frames in the bottom of the TCP/IP stack.

Classes

- class InterfaceBlock

Network interface configuration block class for interface control and configuration.

Functions

- int Removeinterface (int interface)

  Remove a network interface from the system.
- void EnableMulticast (MACADR macAddress, int interface=0)

  Enable Multicast on an existing interface.
- void DisableMulticast (MACADR macAddress, int interface=0)

  Disable Multicast on an existing interface.
- InterfaceBlock * GetInterfaceBlock (int interface=0)

  Get an InterfaceBlock control and configuration object.
- int32_t GetFirstInterface (void)

  Returns the Interface Number of the first registered network interface.
- int32_t GetNextInterface (int lastInterface)

  Returns the Interface Number of the next registered network interface.
- int32_t GetInterfaceNumber (InterfaceBlock *pifb)

  Returns the Interface Number of the specified network interface InterfaceBlock.
- int32_t GetInterfaceForMyAddress4 (IPADDR4 ip)

  Returns the Interface Number of the specified network interface IPv4 address.
- bool GetInterfaceLink (int ifn)

  Returns the network interface link status.
- MACADR InterfaceMAC (int interface)

  Returns the MAC address of the specified network interface.
- bool InterfaceLinkActive (int interface)

  Returns the link status of the specified network interface.
- int InterfaceLinkSpeed (int interface)

  Returns the 10/100 link speed of the specified network interface.
- bool InterfaceLinkDuplex (int interface)

  Returns the full or half link duplex of the specified network interface.
- const char * InterfaceName (int interface)

  Returns the interface name of the specified network interface.

21.71.1 Detailed Description

Network Interface Functions.
Callback functions to add custom Ethenet handlers and pass Ethernet.
Network Interfaces are configured and controlled C++ InterfaceBlock objects. The InterfaceBlock contains variables and member functions for things such as:

- The Configuration Server interface
- IP settings for both IPv4 and IPv6 types
- Configuration to static or DHCP, and DHCP status (both v4 and v6)
- Multihome
- Routing
- AutoIP

Note

While the recommended method for getting and setting network interface parameters is to obtain a pointer to an InterfaceBlock, there are also non-member functions of the object that can be used for situations that are best served by a single function call, such as just getting a single IP address of an interface. When using those functions pay special attention to the return value type. While IPADDR should be used because it supports both IPADDR4 and IPADDR6 types automatically, some of these helper functions may specifically return an IPADDR4 type.

21.71.2 Function Documentation

21.71.2.1 DisableMulticast()

```
void DisableMulticast (
            MACADR macAddress,
            int interface = 0 )
```

Disable Multicast on an existing interface.
Calls registered interface multicast routine.

Parameters

macAddress	Multicast MAC address.
interface	Interface number. If not specified or 0, operates on the first registerd interface.

See also

EnableMulticast(MACADR macAddress, int interface), InterfaceBlock::EnableMulticast()

21.71.2.2 EnableMulticast()

```
void EnableMulticast (
            MACADR macAddress,
            int interface = 0 )
```

Enable Multicast on an existing interface.
Calls registered interface multicast routine.

Parameters

macAddress	Multicast MAC address.
interface	Interface number. If not specified, default is is 0.

See also

DisableMulticast(), InterfaceBlock::EnableMulticast()

21.71.2.3 GetFirstInterface()

```
int32_t GetFirstInterface (
            void  )
```

Returns the Interface Number of the first registered network interface.
If called on a device with wired Ethernet, this will be the first Ethernet interface typical interface number 1.

Returns

Interface number of the first network interface, or 0 if none exist.

21.71.2.4 GetInterfaceBlock()

```
InterfaceBlock * GetInterfaceBlock (
            int interface = 0 )
```

Get an InterfaceBlock control and configuration object.

Parameters

interface	Interface number. If not specified or 0, operates on the first registerd interface.

Returns

Pointer to an InterfaceBlock on success, Null if requested interface is invalid.

21.71.2.5 GetInterfaceForMyAddress4()

```
int32_t GetInterfaceForMyAddress4 (
            IPADDR4 ip )
```

Returns the Interface Number of the specified network interface IPv4 address.

Parameters

ip	IPv4 network interface address.

Returns

Interface number of the specified IPv4 address

21.71.2.6 GetInterfaceLink()

```
bool GetInterfaceLink (
            int ifn )
```

Returns the network interface link status.

Parameters

ifn	Network interface number

Returns

True if link, otherwise false

See also

InterfaceBlock::LinkActive(), InterfaceLinkActive()

21.71.2.7 GetInterfaceNumber()

```
int32_t GetInterfaceNumber (
            InterfaceBlock * pifb )
```

Returns the Interface Number of the specified network interface InterfaceBlock.

Parameters

pifb	Pointer to the InterfaceBlock

Returns

Interface number of the InterfaceBlock.

See also

InterfaceBlock::GetInterfaceNumber()

21.71.2.8 GetNextInterface()

```
int32_t GetNextInterface (
            int lastInterface )
```

Returns the Interface Number of the next registered network interface.

Parameters

lastInterface	Specifies the interface number starting point to begin search.

Returns

Interface number of the next network interface, or 0 if none exist.

21.71.2.9 InterfaceLinkActive()

```
bool InterfaceLinkActive (
            int interface )
```

Returns the link status of the specified network interface.

Parameters

interface	Interface number

Returns

True if link is active, otherwise false

See also

InterfaceBlock::LinkActive(), GetInterfaceLink()

21.71.2.10 InterfaceLinkDuplex()

```
bool InterfaceLinkDuplex (
            int interface )
```

Returns the full or half link duplex of the specified network interface.

Parameters

interface	Interface number

Returns

True if full duplex, false if half duplex

See also

InterfaceBlock::LinkDuplex()

21.71.2.11 InterfaceLinkSpeed()

```
int InterfaceLinkSpeed (
            int interface )
```

Returns the 10/100 link speed of the specified network interface.

Parameters

interface	Interface number

Returns

10 or 100 representing the Mbps speed

See also

InterfaceBlock::LinkSpeed()

21.71.2.12 InterfaceMAC()

```
MACADR InterfaceMAC (
            int interface )
```

Returns the MAC address of the specified network interface.

Parameters

interface	Interface number

Returns

MAC address of type MACADR

21.71.2.13 InterfaceName()

```
const char * InterfaceName (
            int interface )
```

Returns the interface name of the specified network interface.

Parameters

interface	Interface number

Returns

Pointer to a character string containing the name of the network interface

See also

InterfaceBlock::GetInterfaceName()

21.71.2.14 Removeinterface()

```
int Removeinterface (
            int interface )
```

Remove a network interface from the system.

Parameters

interface	Interface number

Returns

1 if successful, 0 if invalid interface number

See also

InterfaceBlock::RegisterInterface()

21.72 Ocotp

Classes

- struct _ocotp_timing

  OCOTP timing structure. Note that, these value are used for calcalating the read/write timings. And the values should statisfy below rules:

Variables

- uint32_t _ocotp_timing::wait
- uint32_t _ocotp_timing::relax
- uint32_t _ocotp_timing::strobe_prog
- uint32_t _ocotp_timing::strobe_read

Driver version

- enum _ocotp_status { }

  Error codes for the OCOTP driver.
- typedef enum _ocotp_status **status_t**

  Error codes for the OCOTP driver.
- typedef struct _ocotp_timing ocotp_timing_t

  OCOTP timing structure. Note that, these value are used for calcalating the read/write timings. And the values should statisfy below rules:
- void OCOTP_Init (OCOTP_Type ∗base, uint32_t srcClock_Hz)

  Initializes OCOTP controller.
- void OCOTP_Deinit (OCOTP_Type ∗base)

  De-initializes OCOTP controller.
- void OCOTP_ReloadShadowRegister (OCOTP_Type ∗base)

  Reload the shadow register. This function will help reload the shadow register without reseting the OCOTP module. Please make sure the OCOTP has been initialized before calling this API.
- uint32_t OCOTP_ReadFuseShadowRegister (OCOTP_Type ∗base, uint32_t address)

  Read the fuse shadow register with the fuse addess.
- status_t OCOTP_WriteFuseShadowRegister (OCOTP_Type ∗base, uint32_t address, uint32_t data)

  Write the fuse shadow register with the fuse addess and data. Please make sure the wrtie address is not locked while calling this API.
- #define **FSL_OCOTP_DRIVER_VERSION** (MAKE_VERSION(2, 0, 0))

  OCOTP driver version 2.0.0.
- #define **MAKE_STATUS**(x, y) y
- #define **assert**(x)

21.72.1 Detailed Description

21.72.2 Typedef Documentation

21.72.2.1 ocotp_timing_t

`typedef struct _ocotp_timing ocotp_timing_t`

OCOTP timing structure. Note that, these value are used for calcalating the read/write timings. And the values should statisfy below rules:
Tsp_rd=(WAIT+1)/ipg_clk_freq should be >= 150ns; Tsp_pgm=(RELAX+1)/ipg_clk_freq should be >= 100ns; Trd = ((STROBE_READ+1)- 2∗(RELAX_READ+1)) /ipg_clk_freq, The Trd is required to be larger than 40 ns. Tpgm = ((STROBE_PROG+1)- 2∗(RELAX_PROG+1)) /ipg_clk_freq; The Tpgm should be configured within the range of 9000 ns < Tpgm < 11000 ns;

21.72.3 Enumeration Type Documentation

21.72.3.1 _ocotp_status

`enum _ocotp_status`

Error codes for the OCOTP driver.

Enumerator

kStatus_OCOTP_AccessError	eFuse and shadow register access error.
kStatus_OCOTP_CrcFail	CRC check failed.
kStatus_OCOTP_AccessError	eFuse and shadow register access error.
kStatus_OCOTP_CrcFail	CRC check failed.

21.72.4 Function Documentation

21.72.4.1 OCOTP_Deinit()

```
void OCOTP_Deinit (
            OCOTP_Type * base )
```

De-initializes OCOTP controller.

Return values

kStatus_Success	upon successful execution, error status otherwise.

21.72.4.2 OCOTP_Init()

```
void OCOTP_Init (
            OCOTP_Type * base,
            uint32_t srcClock_Hz )
```

Initializes OCOTP controller.

Parameters

base	OCOTP peripheral base address.

Parameters

srcClock_Hz	source clock frequency in unit of Hz.

21.72.4.3 OCOTP_ReadFuseShadowRegister()

```
uint32_t OCOTP_ReadFuseShadowRegister (
            OCOTP_Type * base,
            uint32_t address )
```

Read the fuse shadow register with the fuse addess.

Parameters

base	OCOTP peripheral base address.
address	the fuse address to be read from.

21.72.4.4 OCOTP_ReloadShadowRegister()

```
void OCOTP_ReloadShadowRegister (
            OCOTP_Type * base )
```

Reload the shadow register. This function will help reload the shadow register without reseting the OCOTP module. Please make sure the OCOTP has been initialized before calling this API.

Parameters

base	OCOTP peripheral base addess.

21.72.4.5 OCOTP_WriteFuseShadowRegister()

```
status_t OCOTP_WriteFuseShadowRegister (
            OCOTP_Type * base,
            uint32_t address,
            uint32_t data )
```

Write the fuse shadow register with the fuse addess and data. Please make sure the wrtie address is not locked while calling this API.

Parameters

base	OCOTP peripheral base address.
address	the fuse address to be written.
data	the value will be writen to fuse address.

Return values

write	status, kStatus_Success for success and kStatus_Fail for failed.

21.72.5 Variable Documentation

21.72.5.1 relax

`uint32_t _ocotp_timing::relax`
Relax time value to fill in the TIMING register.

21.72.5.2 strobe_prog

`uint32_t _ocotp_timing::strobe_prog`
Storbe program time value to fill in the TIMING register.

21.72.5.3 strobe_read

`uint32_t _ocotp_timing::strobe_read`
Storbe read time value to fill in the TIMING register.

21.72.5.4 wait

`uint32_t _ocotp_timing::wait`
Wait time value to fill in the TIMING register.

21.73 POP3 - Post Office Protocol

Modules

- POP3 Return Codes

Functions

- int POPGetResultCode (int fd, uint32_t timeout)

  Returns the result code of the previous POP3 operation.
- int POP3_InitializeSession (IPADDR server_address, uint16_t port, PCSTR UserName, PCSTR Password, uint32_t timeout)

  Create a connection to the POP3 server and log in.
- int POP3_CloseSession (int session)

  Close a POP3 session.
- int POP3_StatCmd (int session, uint32_t *num_messages, uint32_t *total_bytes, uint32_t timeout)

  Returns the status of the mailstore on the POP3 server.
- int POP3_ListCmd (int session, uint32_t message_number, uint32_t *total_bytes, uint32_t timeout)

  Get the size of a message on the server.
- int POP3_DeleteCmd (int session, uint32_t message_number, uint32_t timeout)

  Delete a pending message on the server.
- int POP3_RetrieveMessage (int session, uint32_t message_number, char *buffer, char **subject_ptr, char **body_ptr, int max_bufferlen, uint32_t timeout)

  Retrieve a message from the server.
- PCSTR GetPOPErrorString (int err)

  Returns the error text for an error code.

21.73.1 Detailed Description

21.73.2 Function Documentation

21.73.2.1 GetPOPErrorString()

```
PCSTR GetPOPErrorString (
            int err )
```

Returns the error text for an error code.

Parameters

err	Error number to decode

Returns

Pointer to a string containing the error code text

21.73.2.2 POP3_CloseSession()

```
int POP3_CloseSession (
            int session )
```

Close a POP3 session.

Parameters

session	The session number returned by POP3_InitializeSession()

Returns

POP3 Return Codes

21.73.2.3 POP3_DeleteCmd()

```
int POP3_DeleteCmd (
            int session,
            uint32_t message_number,
            uint32_t timeout )
```

Delete a pending message on the server.
Note: the server will not actually delete the message until the session is closed.

Parameters

session	The session number returned by POP3_InitializeSession().
message_number	The message number to delete. The message number can be obtained by POP3_StatCmd().
timeout	Timeout in system Time Ticks.

Returns

Return code POP3 Return Codes

21.73.2.4 POP3_InitializeSession()

```
int POP3_InitializeSession (
            IPADDR server_address,
            uint16_t port,
            PCSTR UserName,
            PCSTR Password,
            uint32_t timeout )
```

Create a connection to the POP3 server and log in.

Parameters

server_address	Server IP address
port	Server port number
UserName	Account username
Password	Account password
timeout	Timeout in system Time Ticks

Returns

The POP3 session number if greater than 0, otherwise error code POP3 Return Codes

See also

SSL_POP3_InitializeSession

21.73.2.5 POP3_ListCmd()

```
int POP3_ListCmd (
            int session,
            uint32_t message_number,
            uint32_t * total_bytes,
            uint32_t timeout )
```

Get the size of a message on the server.

Parameters

session	The session number returned by POP3_InitializeSession().
message_number	The message number to query. The message number can be obtained by POP3_StatCmd().
total_bytes	Pointer to the variable in which to store the total number of bytes of the pending message.
timeout	Timeout in system Time Ticks.

Returns

Return code POP3 Return Codes

21.73.2.6 POP3_RetrieveMessage()

```
int POP3_RetrieveMessage (
            int session,
            uint32_t message_number,
            char * buffer,
            char ** subject_ptr,
            char ** body_ptr,
            int max_bufferlen,
            uint32_t timeout )
```

Retrieve a message from the server.

The message is retrieved as a block of bytes including the header, subject, etc. The headers will appear first. Note that the message is left on the server and will not be deleted until the POP3_DeleteCmd() function is called for the message number.

Parameters

session	The session number returned by POP3_InitializeSession().

Parameters

message_number	The message number to retrieve. The message number can be obtained by POP3_StatCmd().
buffer	Pointer to the buffer to store the entire message, including headers
subject_ptr	If not NULL, a pointer to the start of the subject in the message buffer
body_ptr	If not NULL, a pointer to the start of the body in the message buffer
max_bufferlen	Maximum length of the buffer
timeout	Timeout in system Time Ticks.

Returns

Return code POP3 Return Codes

21.73.2.7 POP3_StatCmd()

```
int POP3_StatCmd (
            int session,
            uint32_t * num_messages,
            uint32_t * total_bytes,
            uint32_t timeout )
```

Returns the status of the mailstore on the POP3 server.

Parameters

session	The session number returned by POP3_InitializeSession()
num_messages	Pointer to the variable in which to store the number of pending messages
total_bytes	Pointer to the variable in which to store the total number of bytes of the pending messages
timeout	Timeout in system Time Ticks

Returns

Return code POP3 Return Codes

21.73.2.8 POPGetResultCode()

```
int POPGetResultCode (
            int fd,
            uint32_t timeout )
```

Returns the result code of the previous POP3 operation.

Parameters

fd	TCP socket file descriptor for the active session
timeout	Timeout in system Time Ticks

Returns

POP3 Return Codes

21.74 POP3 Return Codes

Macros

- #define **POP_OK** (0)

  No errors occurred.
- #define **POP_TIMEOUT** (-1)

  Time out.
- #define **POP_PASSWORDERROR** (-2)

  Password error.
- #define **POP_CONNECTFAIL** (-3)

  Connection failed.
- #define **POP_COMMANDFAIL** (-4)

  Command failed.
- #define **POP_BADSESSION** (-5)

  Session error.
- #define **POP_NETWORKERROR** (-6)

  Network error.
- #define **POP_BUFFER_FULL** (-7)

  Receive buffer full.

21.74.1 Detailed Description

21.75 PPP - Point to Point Protocol

Modules

- PPP Connection State
- PPP Return Codes

21.75.1 Detailed Description

Like Ethernet, Point to Point Protocol (PPP) is a data link layer which transfers data between nodes on a network. The main difference is that PPP is used between two end points, whereas Ethernet is a multidrop configuration.
PPP can be used over many types of physical networks, including: fiber, phone line, serial cable and cellular telephone. Once you have created a PPP connection to or from your NetBurner device you can use it with all the standard network applications such as the web server, tcp, udp, ftp, email, etc.
Internet Service Providers (ISP) can implement PPP with many different options that vary from ISP to ISP. The PPP example program included in the development kit examples directory specifies the most common ISP options and can be used as a code example to specify any specific options you may need in your application.

21.76 PPP Connection State

Enumerations

- enum enum_PPPState {
  eClosed , eInitializingModem , eDialing , eWait4Ring ,
  eAnswering , eWaitForTrain , eLCPNegotiate , ePAPAuthenticate ,
  eCHAPAuthenticate , eNCPNegotiate , eOpen , eClosing }

  PPP States.

21.76.1 Detailed Description

21.76.2 Enumeration Type Documentation

21.76.2.1 enum_PPPState

enum enum_PPPState
PPP States.

Enumerator

eClosed	Connection closed.
eInitializingModem	Initializing modem.
eDialing	Dialing.
eWait4Ring	Waiting for ring indicator.
eAnswering	Answering incoming connection.
eWaitForTrain	Waiting for train.
eLCPNegotiate	LCP negotiation.
ePAPAuthenticate	PAP authentication.
eCHAPAuthenticate	CHAP authentication.
eNCPNegotiate	NCP negotiation.
eOpen	Connection open.
eClosing	Connection closing.

21.77 PPP Return Codes

Macros

- #define **ERR_PPP_SUCCESS** (0)
  Success.
- #define **ERR_PPP_ALREADY_OPEN** (-1)
  A session is already open.
- #define **ERR_PPP_NO_DIALTONE** (-2)
  No dial tone.
- #define **ERR_PPP_NO_ANSWER** (-3)
  The remote client did not answer.
- #define **ERR_PPP_BUSY** (-4)
  The remote client is sending a busy signal.
- #define **ERR_PPP_FAIL** (-5)
  The attempted action has failed.
- #define **ERR_PPP_PASSFAIL** (-6)
  Pass/Fail.
- #define **ERR_PPP_LOSTCARRIER** (-7)
  Lost connection carrier signal.
- #define **ERR_PPP_NO_MODEM** (-8)
  No modem detected.
- #define **ERR_PPP_LCP_FAILED** (-9)
  LCP negotiation has failed.
- #define **ERR_PPP_CHAPFAIL** (-10)
  CHAP negotiation has failed.

21.77.1 Detailed Description

21.78 QSPI state

Macros

- #define **QSPI_OK** (0)

  QSPI is busy.
- #define **QSPI_BUSY** (1)

  QSPI is ready.

21.78.1 Detailed Description

21.79 QuadSpiModuleNumber

QuadSPI Peripheral Module.

Macros

- #define **DEFAULT_QUADSPI_MODULE** 0

  Default QUADSPI module.
- #define **QUADSPI_MODULE_COUNT** 1

  Number of modules: 0, 1.

21.79.1 Detailed Description

QuadSPI Peripheral Module.

21.80 SAM Control Area Network (MCAN) Low Level Driver

Module Setting

- enum mcan_timeout_mode { MCAN_TIMEOUT_CONTINUES = MCAN_TOCC_TOS_CONTINUOUS , MCAN_TIMEOUT_TX_EVEN_FIFO = MCAN_TOCC_TOS_TX_EV_TIMEOUT , MCAN_TIMEOUT_RX_FIFO_0 = MCAN_TOCC_TOS_RX0_EV_TIMEOUT , MCAN_TIMEOUT_RX_FIFO_1 = MCAN_TOCC_TOS_RX1↩ _EV_TIMEOUT }

  Can time out modes.
- enum mcan_nonmatching_frames_action { MCAN_NONMATCHING_FRAMES_FIFO_0 , MCAN_NONMATCHING_FRAMES_ , MCAN_NONMATCHING_FRAMES_REJECT }

  Can non-matching frames action.
- enum mcan_interrupt_source {
  MCAN_RX_FIFO_0_NEW_MESSAGE = MCAN_IE_RF0NE , MCAN_RX_FIFO_0_WATERMARK = MCAN_IE_RF0WE , MCAN_RX_FIFO_0_FULL = MCAN_IE_RF0FE , MCAN_RX_FIFO_0_LOST_MESSAGE = MCAN_IE_RF0LE ,
  MCAN_RX_FIFO_1_NEW_MESSAGE = MCAN_IE_RF1NE , MCAN_RX_FIFO_1_WATERMARK = MCAN_IE_RF1WE , MCAN_RX_FIFO_1_FULL = MCAN_IE_RF1FE , MCAN_RX_FIFO_1_MESSAGE_LOST = MCAN_IE_RF1LE ,
  MCAN_RX_HIGH_PRIORITY_MESSAGE = MCAN_IE_HPME , MCAN_TIMESTAMP_COMPLETE = MCAN_IE_TCE , MCAN_TX_CANCELLATION_FINISH = MCAN_IE_TCFE , MCAN_TX_FIFO_EMPTY = MCAN_IE_TFEE ,
  MCAN_TX_EVENT_FIFO_NEW_ENTRY = MCAN_IE_TEFNE , MCAN_TX_EVENT_FIFO_WATERMARK = MCAN_IE_TEFWE , MCAN_TX_EVENT_FIFO_FULL = MCAN_IE_TEFFE , MCAN_TX_EVENT_FIFO_ELEMENT_LOST = MCAN_IE_TEFLE ,
  MCAN_TIMESTAMP_WRAPAROUND = MCAN_IE_TSWE , MCAN_MESSAGE_RAM_ACCESS_FAILURE = MCAN_IE_MRAFE , MCAN_TIMEOUT_OCCURRED = MCAN_IE_TOOE , MCAN_RX_BUFFER_NEW_MESSAGE = MCAN_IE_DRXE ,

MCAN_ERROR_LOGGING_OVERFLOW = MCAN_IE_ELOE , MCAN_ERROR_PASSIVE = MCAN_IE_EPE , MCAN_WARNING_STATUS = MCAN_IE_EWE , MCAN_BUS_OFF = MCAN_IE_BOE , MCAN_WATCHDOG = MCAN_IE_WDIE , MCAN_CRC_ERROR = MCAN_IE_CRCEE , MCAN_BIT_ERROR = MCAN_IE_BEE , MCAN_ACKNOWLEDGE_ERROR = MCAN_IE_ACKEE , MCAN_FORMAT_ERROR = MCAN_IE_FOEE , MCAN_STUFF_ERROR = MCAN_IE_STEE }

Can module interrupt source.

21.80.1 Detailed Description

This driver for Atmel® | SMART SAM devices provides an low level interface for the configuration and management of the device's Control Area Network functionality.

Note

Since "The Control Area Network (CAN) performs communication according to ISO 11898-1 (Bosch CAN specification 2.0 part A,B) and to Bosch CAN FD specification V1.0", the driver is focus on the MAC layer and try to offer the APIs which can be used by upper application layer.

For storage of Rx/Tx messages and for storage of the filter configuration, a message RAM is needed to the CAN module. In this driver, the message RAM is static allocated, the related setting is defined and can be changed in the module configuration file "conf_mcan.h".
The following peripherals are used by this module:

- CAN (Control Area Network)

The following devices can use this module:

- SAMV71

The outline of this documentation is as follows:

- Prerequisites
- Module Overview
- Special Considerations
- Extra Information
- Examples
- API Overview

21.80.2 Prerequisites

There are no prerequisites for this module.

21.80.3 Module Overview

This driver provides an interface for the Control Area Network Controller functions on the device.

21.80.4 Special Considerations

There are no special considerations for this module.

21.80.5 Extra Information

For extra information see asfdoc_sam_mcan_extra. This includes:

- asfdoc_sam_mcan_extra_acronyms
- asfdoc_sam_mcan_extra_dependencies
- asfdoc_sam_mcan_extra_errata
- asfdoc_sam_mcan_extra_history

21.80.6 Examples

For a list of examples related to this driver, see asfdoc_sam_mcan_exqsg.

21.80.7 API Overview

21.80.8 Enumeration Type Documentation

21.80.8.1 mcan_interrupt_source

enum mcan_interrupt_source
Can module interrupt source.
Enum for the interrupt source.

Enumerator

MCAN_RX_FIFO_0_NEW_MESSAGE	Rx FIFO 0 New Message Interrupt Enable.
MCAN_RX_FIFO_0_WATERMARK	Rx FIFO 0 Watermark Reached Interrupt Enable.
MCAN_RX_FIFO_0_FULL	Rx FIFO 0 Full Interrupt Enable.
MCAN_RX_FIFO_0_LOST_MESSAGE	Rx FIFO 0 Message Lost Interrupt Enable.
MCAN_RX_FIFO_1_NEW_MESSAGE	Rx FIFO 1 New Message Interrupt Enable.
MCAN_RX_FIFO_1_WATERMARK	Rx FIFO 1 Watermark Reached Interrupt Enable.
MCAN_RX_FIFO_1_FULL	Rx FIFO 1 Full Interrupt Enable.
MCAN_RX_FIFO_1_MESSAGE_LOST	Rx FIFO 1 Message Lost Interrupt Enable.
MCAN_RX_HIGH_PRIORITY_MESSAGE	High Priority Message Interrupt Enable.
MCAN_TIMESTAMP_COMPLETE	Transmission Completed Interrupt Enable.
MCAN_TX_CANCELLATION_FINISH	Transmission Cancellation Finished Interrupt Enable.
MCAN_TX_FIFO_EMPTY	Tx FIFO Empty Interrupt Enable.
MCAN_TX_EVENT_FIFO_NEW_ENTRY	Tx Event FIFO New Entry Interrupt Enable.
MCAN_TX_EVENT_FIFO_WATERMARK	Tx Event FIFO Watermark Reached Interrupt Enable.
MCAN_TX_EVENT_FIFO_FULL	Tx Event FIFO Full Interrupt Enable.
MCAN_TX_EVENT_FIFO_ELEMENT_LOST	Tx Event FIFO Element Lost Interrupt Enable.
MCAN_TIMESTAMP_WRAPAROUND	Timestamp Wraparound Interrupt Enable.
MCAN_MESSAGE_RAM_ACCESS_FAILURE	Message RAM Access Failure Interrupt Enable.
MCAN_TIMEOUT_OCCURRED	Timeout Occurred Interrupt Enable.
MCAN_RX_BUFFER_NEW_MESSAGE	Message stored to Dedicated Rx Buffer Interrupt Enable.
MCAN_ERROR_LOGGING_OVERFLOW	Error Logging Overflow Interrupt Enable.
MCAN_ERROR_PASSIVE	Error Passive Interrupt Enable.
MCAN_WARNING_STATUS	Warning Status Interrupt Enable.
MCAN_BUS_OFF	Bus_Off Status Interrupt Enable.
MCAN_WATCHDOG	Watchdog Interrupt Enable.
MCAN_CRC_ERROR	CRC Error Interrupt Enable
MCAN_BIT_ERROR	Bit Error Interrupt Enable.
MCAN_ACKNOWLEDGE_ERROR	Acknowledge Error Interrupt Enable .
MCAN_FORMAT_ERROR	Format Error Interrupt Enable
MCAN_STUFF_ERROR	Stuff Error Interrupt Enable

21.80.8.2 mcan_nonmatching_frames_action

enum mcan_nonmatching_frames_action
Can non-matching frames action.

Enumerator

MCAN_NONMATCHING_FRAMES_FIFO_0	Accept in Rx FIFO 0.
MCAN_NONMATCHING_FRAMES_FIFO_1	Accept in Rx FIFO 1.
MCAN_NONMATCHING_FRAMES_REJECT	Reject.

21.80.8.3 mcan_timeout_mode

enum mcan_timeout_mode
Can time out modes.

Enumerator

MCAN_TIMEOUT_CONTINUES	Continuous operation.
MCAN_TIMEOUT_TX_EVEN_FIFO	Timeout controlled by TX Event FIFO.
MCAN_TIMEOUT_RX_FIFO_0	Timeout controlled by Rx FIFO 0.
MCAN_TIMEOUT_RX_FIFO_1	Timeout controlled by Rx FIFO 1.

21.81 SAME70 (DSPI)

Modules

- DspiModuleNumber
  DSPI Peripheral Module.
- DspiState
  DSPI Bus State .

Classes

- class SPIModule
  SPI Peripheral Module Class.

Enumerations

- enum csReturnType {
  DEASSERT_NEVER = 0 , DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 , DEASSERT_NEVER = 0 ,
  DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 }
  Chip select deassertion modes. Used to determine when the driver should deassert chip selects during SPI transfer.
- enum spiChipSelect {
  CHIP_SELECT_0 = 0xFE , CHIP_SELECT_1 = 0xFD , CHIP_SELECT_2 = 0xFB , CHIP_SELECT_3 = 0xF7 ,
  CHIP_SELECT_DISABLED = 0xFF , CHIP_SELECT_0 = 0 , CHIP_SELECT_1 = 1 , CHIP_SELECT_2 = 2 , CHIP_SELECT_3 = 3 , CHIP_SELECT_DISABLED = 0xFF }
  Chip select number.
- enum spiChipSelectPolarity { CS_ASSERT_LOW = 0x00 , CS_ASSERT_HIGH = 0xFF , CS_ASSERT_LOW = 0 , CS_ASSERT_HIGH = 1 }
  Chip select polarity.

Functions

- uint8_t DSPIInit (uint8_t SPIModule=DEFAULT_DSPI_MODULE, uint32_t Baudrate=2000000, uint8_t QueueBitSize=8, uint8_t CS=0x00, uint8_t CSPol=0x0F, uint8_t ClkPolarity=0, uint8_t ClkPhase=1, BOOL DoutHiz=TRUE, uint8_t QCD=0, uint8_t DTL=0)

  Initialize a DSPI module.
- uint8_t DSPIStart (uint8_t SPIModule, puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL, uint8_t enableDMA=TRUE, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Start a DSPI transfer.
- BOOL DSPIdone (uint8_t SPIModule=DEFAULT_DSPI_MODULE)

  Check SPI status.
- uint8_t QSPIInit (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x0F, uint8_t chipSelectPolarity=1, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Compatibility function for previous drivers. Initialize SPI module.
- uint8_t QSPIStart (puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL)

  Compatibility function for previous drivers. Start a SPI transfer.
- BOOL QSPIdone ()

  Compatibility function for previous drivers. Check SPI status.

21.81.1 Detailed Description

Supported Platforms:

- MODM7AE70
- SBE70LC

The NetBurner ARM SAME70 DSPI Group. Note that for the ARM SAME70 there are 3 types of SPI peripherals:

- SPI using the SPI peripheral: Provides the most configuration options. Used in single data channel mode: SCLK, MISO, MOSI, Chip Selects.
- SPI using the QSPI peripheral: Single data channel as with SPI, but fewer configuration options.
- SPI using the USART peripheral: Single data channel as with SPI, but fewer configuration options.

21.81.2 Enumeration Type Documentation

21.81.2.1 csReturnType

`enum csReturnType`

Chip select deassertion modes. Used to determine when the driver should deassert chip selects during SPI transfer.

Enumerator

DEASSERT_NEVER	The chip select used for the transaction should remain asserted, even after the transaction is complete.
DEASSERT_AFTER_LAST	The chip select should remain asserted for the full duration of the transaction, and only be deasserted after the final transfer.
DEASSERT_EVERY_TRANSFER	The chip select should be deasserted between every transfer within the transaction.
DEASSERT_NEVER	The chip select used for the transaction should remain asserted, even after the transaction is complete.

Enumerator

DEASSERT_AFTER_LAST	The chip select should remain asserted for the full duration of the transaction, and only be deasserted after the final transfer.
DEASSERT_EVERY_TRANSFER	The chip select should be deasserted between every transfer within the transaction.Deassert chip select After every transfer.

21.81.2.2 spiChipSelect

`enum spiChipSelect`
Chip select number.
If supported by the SPI peripheral, these values can be used to configure the chip select to be used during a SPI transfer.

Enumerator

CHIP_SELECT_0	Configure bit mask for SPI chip select 0.
CHIP_SELECT_1	Configure bit mask for SPI chip select 1.
CHIP_SELECT_2	Configure bit mask for SPI chip select 2.
CHIP_SELECT_3	Configure bit mask for SPI chip select 3.
CHIP_SELECT_DISABLED	Configure bit mask to disable chip select.
CHIP_SELECT_0	Configure the SPI peripheral to use chip select 0.
CHIP_SELECT_1	Configure the SPI peripheral to use chip select 1.
CHIP_SELECT_2	Configure the SPI peripheral to use chip select 2.
CHIP_SELECT_3	Configure the SPI peripheral to use chip select 3.
CHIP_SELECT_DISABLED	Configure the SPI peripheral to disable chip select.

21.81.2.3 spiChipSelectPolarity

`enum spiChipSelectPolarity`
Chip select polarity.
This enum exists for API compatibility between platforms. The SAME70's SPI peripheral does not support modifying chip select polarity. The SPI peripheral only supports asserting chip select to logic level LOW during a transaction. If you need to assert the chip select to logic level HIGH during a transaction, your alternative is to use a GPIO for chip select.

Enumerator

CS_ASSERT_LOW	Bit mask configures all active chip selects as active low, inactive high.
CS_ASSERT_HIGH	Bit mask configures all active chip selects as inactive low, active high.
CS_ASSERT_LOW	Assert all chip selects to logic level LOW during a SPI transaction.
CS_ASSERT_HIGH	Assert all chip selects to logic level HIGH during a SPI transaction.

21.81.3 Function Documentation

21.81.3.1 DSPIdone()

```
BOOL DSPIdone (
            uint8_t SPIModule = DEFAULT_DSPI_MODULE )
```

Check SPI status.

Returns

true if DSPI is finished, false if active

21.81.3.2 DSPIInit()

```
uint8_t DSPIInit (
            uint8_t SPIModule = DEFAULT_DSPI_MODULE,
            uint32_t Baudrate = 2000000,
            uint8_t QueueBitSize = 8,
            uint8_t CS = 0x00,
            uint8_t CSPol = 0x0F,
            uint8_t ClkPolarity = 0,
            uint8_t ClkPhase = 1,
            BOOL DoutHiz = TRUE,
            uint8_t QCD = 0,
            uint8_t DTL = 0 )
```

Initialize a DSPI module.
Notes:

- The maximum baud rate is CPU_CLOCK / 3
- Will initialize to the highest available baud rate that does not exceed the maximum
- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for greater than 8 bits per transfer then the data must be uint16_t aligned
- If configured for greater than 16 bits per transfer then the data must be uint32_t aligned
- Chip select is based on chipSelectPolarity
- 0 data captured leading edge of DSPI_CLK, changed following edge.
- 1 data changed leading edge of DSPI_CLK, captured following edge.
- 0 default is as close to 1/2 DSPI_CLK without going under, keeping with the interface to SPI
- 0 default is 17/(system clock / 2), in keeping with interface to QSPI

Parameters

SPIModule	SPI module number, 0 - 1
Baudrate	Maximum baud rate requested
QueueBitSize	Number of bits per transfer: 8, 16 or 32
CS	SPI chip selects to use for transfer
CSPol	0 = inactive logic level low, 1 = high
ClkPolarity	0 = inactive logic level low, 1 = high
ClkPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
DoutHiz	Data output high impedance between transfers
QCD	Delay from chip select to valid clock (default is 0)
DTL	Chip select mode dspiChipSelectMode

Returns

Current stat of DSPI bus dspiState

21.81.3.3 DSPIStart()

```
uint8_t DSPIStart (
            uint8_t SPIModule,
            puint8_t transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            OS_SEM * finishedSem = NULL,
            uint8_t enableDMA = TRUE,
            int csReturnToInactive = DEASSERT_AFTER_LAST )
```

Start a DSPI transfer.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.
- If DSPI_Finished points to a semaphore, then the DSPI will POST to it when the transfer is complete.
- The semaphore is optional, but it can increase efficiency.

Parameters

SPIModule	DSPI module to use, 0 - 1
transmitBufferPtr	Pointer to the buffer containing the data to transmit
receiveBufferPtr	Pointer to the buffer to store the received data
byteCount	Number of bytes to transmit
finishedSem	Optional semaphore to post to when finished
enableDMA	Enable DMA transfers
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

21.81.3.4 QSPIdone()

```
BOOL QSPIdone (
            void  )  [inline]
```

Compatibility function for previous drivers. Check SPI status.
Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.
Note that the 'Q' stands for Queued SPI

Returns

true if DSPI is finished, false if active

21.81.3.5 QSPIInit()

```
uint8_t QSPIInit (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x0F,
            uint8_t chipSelectPolarity = 1,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )  [inline]
```

Compatibility function for previous drivers. Initialize SPI module.
Initialize Queued Serial Peripheral Interface (QSPI)
Note that the 'Q' stands for Queued SPI
Will use the default SPI module, 0.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.
- If DSPI_Finished points to a semaphore, then the DSPI will POST to it when the transfer is complete.
- The semaphore is optional, but it can increase efficiency.

Parameters

baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8, 16 or 32
peripheralChipSelects	SPI chip selects to use for transfer
chipSelectPolarity	0 = inactive logic level low, 1 = high
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	Delay from chip select to valid clock (default is 0)
delayAfterTransfer	Chip select mode spiChipSelect

Returns

The current state of the SPI bus dspiState

21.81.3.6 QSPIStart()

```
uint8_t QSPIStart (
            puint8_t transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            OS_SEM * finishedSem = NULL )  [inline]
```

Compatibility function for previous drivers. Start a SPI transfer.
Start QSPI Data Transfer.
Note that the 'Q' stands for Queued SPI

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.
- If DSPI_Finished points to a semaphore, then the DSPI will POST to it when the transfer is complete.
- The semaphore is optional, but it can increase efficiency.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
receiveBufferPtr	Pointer to the buffer to store the received data
byteCount	Number of bytes to transmit
finishedSem	Optional semaphore to post to when finished

Returns

The current state of the SPI bus dspiState

21.82 SAME70 (QSPI)

Modules

- QuadSpiModuleNumber

  QuadSPI Peripheral Module.

Classes

- class SPI_QSPI

  The Single-Bit SPI mode QSPI Peripheral Class.

21.82.1 Detailed Description

Supported Platforms:

- MODM7AE70
- SBE70LC

The NetBurner ARM SAME70 Quad SPI (QSPI) API. Note that there are 3 types of SPI peripherals:

- SPI using the SPI peripheral: Provides the most configuration options. Used in single data channel mode: SCLK, MISO, MOSI, Chip Selects.
- SPI using the QuadSPI peripheral: Single data channel as with SPI, but fewer configuration options.
- SPI using the USART peripheral: Single data channel as with SPI, but fewer configuration options.

21.83 SAME70 (USART)

Modules

- UsartModuleNumber

  USART Peripheral Module.

Classes

- class SPI_USART

  USART in SPI mode Peripheral Module Class.

21.83.1 Detailed Description

Supported Platforms:

- MODM7AE70
- SBE70LC

Note that for the ARM SAME70 there are 3 types of SPI peripherals:

- SPI using the SPI peripheral: Provides the most configuration options. Used in single data channel mode: SCLK, MISO, MOSI, Chip Selects.
- SPI using the QSPI peripheral: Single data channel as with SPI, but fewer configuration options.
- SPI using the USART peripheral: Single data channel as with SPI, but fewer configuration options.

21.84 SAME70 GPIO (MODM7AE70)

Classes

- class PinIO

  GPIO Pin Class.
- class _PinVector

  GPIO Pin Vector Base Class.
- class PinVector< n >

  GPIO Pin Vector Class PinVector is a template instantiation of the _PinVector class, allowing for minimal storage requirements for potentially large vectors, without heavy code duplication due to template copies.

Variables

- volatile Pio & PinIO::pio
- uint32_t PinIO::mask

21.84.1 Detailed Description

The Netburner ARM SAME70 GPIO Pins class driver.

21.84.2 Variable Documentation

21.84.2.1 mask

`PinIO::mask`
Bit mask setting which pins in the given Pio hardware module the PinIO object will manage.

21.84.2.2 pio

`PinIO::pio`
Reference to the Pio hardware module that this PinIO will manage. Combines with the mask to manage a single pin or multiple within the module.

21.85 SAME70 I2C

Classes

- class WireIntf

  Wire Interface Class for I2C.
- class I2C

  I2C Peripheral Class.
- class I2CDevice

  I2C Device Class (recommended)

21.85.1 Detailed Description

Supported Platforms:

- MODM7AE70
- SBE70LC

  The I2C peripheral interfaces can be controlled three ways:

  – The I2C Device class (recommended): I2CDevice
  – The I2C Class, which offers additional low level control I2C
  – The Wire Interface Class, which offers a Two Wire Interface API WireIntf

21.86 SMTP - Send Email

Modules

- MIME Content Types
- SMTP Error Codes

Functions

- int SendMail (IPADDR smtp_server, PCSTR userid, PCSTR from_addr, PCSTR to_addr, PCSTR subject, PCSTR textbody)

  Send an email message. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message.
- int SendMailAuth (IPADDR smtp_server, PCSTR userid, PCSTR pass, PCSTR from_addr, PCSTR to_addr, PCSTR subject, PCSTR textbody)

  Send an email message with plain text authentication. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message.
- int SendMailEx (IPADDR smtp_server, PCSTR userid, PCSTR from_addr_rev_path, PCSTR from_addr_↩ memo_hdr, PCSTR to_addr, PCSTR subject, PCSTR textbody)

  Send an email message function, extended version.
- int IsMailError ()

  Returns the error status of the last send mail transaction.
- void PrintNBError (int fd=0)

  If an error occurred, prints the error information received from the SMTP server.
- void PrintServerLog (int fd=0)

  Prints the server log of the last send mail transaction.
- int SendMailAuthStartMIME (IPADDR smtp_server, PCSTR userid, PCSTR pass, PCSTR from_addr, PCSTR to_addr, PCSTR subject, int &fd)

  Start a Multi-purpose Internet Mail Extension (MIME)session.
- int SendMailAuthAddMIME (int fd, int ContentType, const char ∗pContent, const char ∗FileName)

  Add a MIME part or attachment to an open MIME Session.

- int SendMailAuthEndMIME (int fd, PCSTR userid)

  Send a MIME email message and close the SMTP session.

- enum CONTENT_TYPE_ENUM {
  CONTENT_TYPE_PLAIN_TEXT , CONTENT_TYPE_PLAIN_TEXT_ATTACH , CONTENT_TYPE_BINARY_ATTACH
  , CONTENT_TYPE_HTML_DECOMP ,
  CONTENT_TYPE_END }

  SMTP MIME Conetnet Types.

21.86.1 Detailed Description

Send emails using the SMTP specification. The Send Mail functions provide a variety of sending options for simple sending an email, or sending with authentication. The IsMailError function is used to check for errors after an email is sent.

21.86.2 Enumeration Type Documentation

21.86.2.1 CONTENT_TYPE_ENUM

```
enum CONTENT_TYPE_ENUM
```

SMTP MIME Conetnet Types.

Enumerator

CONTENT_TYPE_PLAIN_TEXT	Plain text.
CONTENT_TYPE_PLAIN_TEXT_ATTACH	Plain text attachment.
CONTENT_TYPE_BINARY_ATTACH	Binary attachment.
CONTENT_TYPE_HTML_DECOMP	HTML.
CONTENT_TYPE_END	Additional content types can be added above this line.

21.86.3 Function Documentation

21.86.3.1 IsMailError()

```
int IsMailError ( )
```

Returns the error status of the last send mail transaction.

Returns

0 if no error occurred, SMTP Error Codes.

See also

PrintNBError(), PrintServerLog()

21.86.3.2 PrintNBError()

```
void PrintNBError (
            int fd = 0 )
```

If an error occurred, prints the error information received from the SMTP server.
If an error occurs when sending an email, the system will record the error information received from the SMTP server. The information is only valid for the last transaction.

Parameters

fd	If no parameter is specified information will be sent to stdout. Otherwise it will be sent to the specified file descriptor.

See also

IsMailError(), PrintServerLog()

21.86.3.3 PrintServerLog()

```
void PrintServerLog (
            int fd = 0 )
```

Prints the server log of the last send mail transaction.
The results of each send mail transaction are logged to a buffer. This function will send the results of the last transaction to stdout, or the specified file descriptor. The information is only valid for the last transaction.

Parameters

fd	If no parameter is specified information will be sent to stdout. Otherwise it will be sent to the specified file descriptor.

See also

IsMailError(), PrintNBError()

21.86.3.4 SendMail()

```
int SendMail (
            IPADDR smtp_server,
            PCSTR userid,
            PCSTR from_addr,
            PCSTR to_addr,
            PCSTR subject,
            PCSTR textbody )
```

Send an email message. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message.

Parameters

smtp_server	IP address of the destination SMTP server.
userid	ASCII string representing the RFC 931 identification.
from_addr	"From" email address
to_addr	Destination email address
subject	Email subject
textbody	Body of email message

Returns

0 on failure, 1 on success.

See also

SendMailAuth()

21.86.3.5 SendMailAuth()

```
int SendMailAuth (
            IPADDR smtp_server,
            PCSTR userid,
            PCSTR pass,
            PCSTR from_addr,
            PCSTR to_addr,
            PCSTR subject,
            PCSTR textbody )
```

Send an email message with plain text authentication. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message.
Send an email message with SMTP plain text authentication per RFC 931. The function will open a TCP connection to the specified SMTP server, authenticate the connection with the specified username and password, create a message based on the parameters, and send the message.

Parameters

smtp_server	IP address of the destination SMTP server.
userid	ASCII string representing the RFC 931 identification.
pass	ASCII string to provide for AUTH identification.
from_addr	"From" email address
to_addr	Destination email address
subject	Email subject
textbody	Body of email message

Returns

0 on failure, 1 on success.

See also

SendMail()

21.86.3.6 SendMailAuthAddMIME()

```
int SendMailAuthAddMIME (
            int fd,
            int ContentType,
            const char * pContent,
            const char * FileName )
```

Add a MIME part or attachment to an open MIME Session.
Must only be called for an open session started with SendMailAuthStartMIME(). It can be called as many times as needed to add each MIME part or attachment to the message.

Parameters

fd	The file descriptor of the open MIME session.
ContentType	The MIME content type, such as: CONTENT_TYPE_PLAIN_TEXT, CONTENT_TYPE_PLAIN_TEXT_ATTACH, CONTENT_TYPE_BINARY_ATTACH.
∗*pContent*	Pointer to the content to add.
∗*FileName*	File name for the attachment.

Returns

0 on failure, 1 on success.

See also

SendMailAuthStartMIME(), SendMailAuthEndMIME()

21.86.3.7 SendMailAuthEndMIME()

```
int SendMailAuthEndMIME (
            int fd,
            PCSTR userid )
```

Send a MIME email message and close the SMTP session.
Must only be called for an open session started with SendMailAuthStartMIME().

Parameters

fd	The file descriptor of the open MIME session.
∗userid	The user ID that was used by SendMailAuthStartMIME() to open the session.

Returns

0 on failure, 1 on success.

See also

SendMailAuthStartMIME(), SendMailAuthAddMIME()

21.86.3.8 SendMailAuthStartMIME()

```
int SendMailAuthStartMIME (
            IPADDR smtp_server,
            PCSTR userid,
            PCSTR pass,
            PCSTR from_addr,
            PCSTR to_addr,
            PCSTR subject,
            int & fd )
```

Start a Multi-purpose Internet Mail Extension (MIME)session.
Start a MIME session with plain text password authentication. This function will open a TCP connection to the SMTP server, authenticate, then return while leaving the TCP connection open. The Session file descriptor is returned in the "int &fd" reference variable. This function must be called before any other SendMail MIME function.
Sending an E-Mail message with Multipurpose Internet Mail Extensions (MIME) is a multi step process:

- Call SendMailAuthStartMIME( ) to begin a Session with the SMTP server, connection will remain open.
- Call SendMailAuthAddMIME( ) for each MIME attachment you want to add.
- Call SendMailAuthEndMIME( ) to send the email message and close the Session.

The following MIME types are supported by default. The binary attachment should cover most types such as jpg, gif, etc. The types are defined in `\nburn\nbrtos\include\mailto.h`, and the functions are implemented in `\nburn\system\mailto.cpp`.
MIME Content Types defined by MIME Content Types.

Parameters

smtp_server	IP address of the destination SMTP server.
userid	ASCII string representing the RFC 931 identification.
pass	Password.
from_addr	Reverse path per RFC 821, reverse source route.
to_addr	Destination email address.
subject	Email subject.
fd	A reference (or pointer) to an integer value of the calling function. The value of fd is modified by this function and will contain the file descriptor for the open SMTP Session. This value must always be checked to be greater than 0 for a valid Session. The fd is passed to other MIME functions to identify the open Session.

Returns

0 on failure, 1 on success.

See also

SendMailAuthAddMIME(), SendMailAuthEndMIME()

21.86.3.9 SendMailEx()

```
int SendMailEx (
            IPADDR smtp_server,
            PCSTR userid,
            PCSTR from_addr_rev_path,
            PCSTR from_addr_memo_hdr,
            PCSTR to_addr,
            PCSTR subject,
            PCSTR textbody )
```

Send an email message function, extended version.

Send an email message with extended parameters. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message. This function is identical to SendMail( ), with the addition of the from_addr_rev_path and from_addr_memo_hdr parameters.

Parameters

smtp_server	IP address of the destination SMTP server.
userid	ASCII string representing the RFC 931 identification.
from_addr_rev_path	Reverse path per RFC 821, reverse source route.
from_addr_memo_hdr	From address memo header per RFC 821. Contains items such as date, subject, to:, CC: and from:.
to_addr	Destination email address
subject	Email subject
textbody	Body of email message

Returns

0 on failure, 1 on success.

See also

SendMail()

21.87 SMTP Error Codes

Macros

- #define **STATUS_OK** (0)

  OK, no errors.
- #define **CONNECT_TO_SMTP_SERVER_FAILED** (-1)

  Could not connect to SMTP server.
- #define **INITIAL_SERVER_REPLY_FAILED** (-2)

  Initial server reply failed.
- #define **HELO_SERVER_REPLY_FAILED** (-3)

  Server HELO reply failed.
- #define **MAIL_FROM_SERVER_REPLY_FAILED** (-4)

  Mail From server reply failed.
- #define **RCPT_TO_SERVER_REPLY_FAILED** (-5)

  Receipt To server reply failed.
- #define **DATA_SERVER_REPLY_FAILED** (-6)

  Data server reply failed.
- #define **DATA_END_SERVER_REPLY_FAILED** (-7)

  Date end server reply failed.
- #define **AUTH_LOGIN_SERVER_REPLY_FAILED** (-8)

  AUTH login server reply failed.
- #define **USER_ID_SERVER_REPLY_FAILED** (-9)

  User ID server reply failed.
- #define **PASSWORD_SERVER_REPLY_FAILED** (-10)

  Password server reply failed.
- #define **CONNECT931_SMTP_SERVER_FAILED** (-11)

  SMTP connection failed.

21.87.1 Detailed Description

21.88 SOCKS

Modules

- SOCKS Error Codes

Macros

- #define **SOCKS_MAX_UNAME_SIZE** 255

  Max character length for usernames. Defined in RFC 1929.
- #define **SOCKS_MAX_PASSWD_SIZE** 255

  Max character length for passwords. Defined in RFC 1929.

Enumerations

- enum SocksAuthType : unsigned char { eSocksAuthTypeGssApi = 0x01 , eSocksAuthTypeUnPw = 0x02 , eSocksAuthTypeNoAuth = 0x04 }

  SOCKS Autherization Types.
- enum SocksClientCmd : unsigned char { eSocksClientCmdConnect = 1 , eSocksClientCmdBind = 2 , eSocksClientCmdUdpAssoc = 3 }

  SOCKS Client Commands.
- enum SocksAdrType : unsigned char { eSocksAdrTypeNone = 0x00 , eSocksAdrTypeIpv4 = 0x01 , eSocksAdrTypeDomain = 0x03 , eSocksAdrTypeIpv6 = 0x04 }

  SOCKS Address Types.

Functions

- bool AuthWithGssApi ()

  A weak function that should be overriden by the developer in order to support GSSAPI authorization.
- void SetSocksProxySettings (SocksProxy *socksProxy)

  Set the system level SOCKS proxy settings object to the one that is passed in. If this object is set and is marked as enabled, calls to CoreConnect() (made from connect(), DoGet(), etc), will all initially connect to the proxy server using the SOCKS5 protocol.
- SocksProxy * GetSocksProxySettings ()

  Get a pointer to the currnetly set Socks proxy settings object.

21.88.1 Detailed Description

The NetBurner SOCKS Library

21.88.2 Enumeration Type Documentation

21.88.2.1 SocksAdrType

`enum SocksAdrType : unsigned char`
SOCKS Address Types.

Enumerator

eSocksAdrTypeNone	None.
eSocksAdrTypeIpv4	IPv4.
eSocksAdrTypeDomain	Domain.
eSocksAdrTypeIpv6	IPv6.

21.88.2.2 SocksAuthType

`enum SocksAuthType : unsigned char`
SOCKS Autherization Types.

Enumerator

eSocksAuthTypeGssApi	GSS API.
eSocksAuthTypeUnPw	User name and password.
eSocksAuthTypeNoAuth	Not as defined in RFC so that it could be propery tested against when being set by user.

21.88.2.3 SocksClientCmd

`enum SocksClientCmd : unsigned char`
SOCKS Client Commands.

Enumerator

eSocksClientCmdConnect	Connect.
eSocksClientCmdBind	Not currently supported, see RFC 1928.
eSocksClientCmdUdpAssoc	Not currently supported, see RFC 1928.

21.88.3 Function Documentation

21.88.3.1 AuthWithGssApi()

```
bool AuthWithGssApi ( )
```

A weak function that should be overriden by the developer in order to support GSSAPI authorization.

Return values

true	The GSSAPI authorization succeeded.
false	The GSSAPI authorization failed.

21.88.3.2 GetSocksProxySettings()

```
SocksProxy * GetSocksProxySettings ( )
```

Get a pointer to the currnetly set Socks proxy settings object.

Returns

SocksProxy∗

21.88.3.3 SetSocksProxySettings()

```
void SetSocksProxySettings (
            SocksProxy * socksProxy )
```

Set the system level SOCKS proxy settings object to the one that is passed in. If this object is set and is marked as enabled, calls to CoreConnect() (made from connect(), DoGet(), etc), will all initially connect to the proxy server using the SOCKS5 protocol.

Parameters

socksProxy	The proxy object that the system should reference.

21.89 SOCKS Error Codes

Macros

- #define **SOCKS_SUCCESS** (0)

  The connection was successful.
- #define **SOCKS_ERR_TIMEOUT** (-500)

  The connection timed out.
- #define **SOCKS_BAD_ADR_TYPE** (-501)

  Address type specified does not match the address provided.
- #define **SOCKS_CONN_ABORTED** (-502)

  The other side aborted the connection.
- #define **SOCKS_BAD_AUTH_TYPE** (-503)

  The server doesn't support the requested authorization types.
- #define **SOCKS_BAD_UN_PW** (-504)

  The server did not approve the username and password provided.
- #define **SOCKS_UN_TOO_LONG** (-505)

The specified username is too long.

- #define **SOCKS_PW_TOO_LONG** (-506)

  The specified password is too long.

- #define **SOCKS_BAD_REPLY** (-507)

  The server reply was not successful.

- #define **SOCKS_BAD_DOMAIN** (-508)

  Unable to resolve domain name passed.

- #define **SOCKS_CMD_NOT_SUPPORTED** (-509)

  The requested command aren't supported.

- #define **SOCKS_BAD_PARAM** (-510)

  The passed in parameters are not valid.

21.89.1 Detailed Description

Error codes that can be returned by SOCKS related functions.

21.90 SPI

Modules

- MCF5441x (DSPI)
- MCF5441x (QSPI)
- SAME70 (DSPI)
- SAME70 (QSPI)
- SAME70 (USART)

21.90.1 Detailed Description

Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals over a short distance, such as port expanders, FPGAs, CPLDs, shift registers, sensors, and flash SD cards. Unlike an asynchronous UART that does not use a separate clock, the SPI operates in a synchronous mode using a single clock for both sender and receiver that determines the data rate as well as when each bit of data is valid. This API uses the "native" SPI mode with the following signals:

- Clock (CLK). Synchronous clock signal.
- Master out, Slave in (MOSI). Data signal from master to slave device.
- Master in, Slave out (MISO). Data signal from slave to master.
- Chip select (CS). Chip selects can be part of the SPI peripheral, or can be general purpose I/O.

An advantage of SPI is that the receiving hardware can be a simple shift register. This is a much simpler and less expensive than a UART implementation.

21.91 SSH

Modules

- SSH Error Codes

Typedefs

- typedef int(∗ sshUserAuthenticateFn) (const char ∗usernamePtr, const char ∗passwordPtr)

  [DEPRECATED] User provided SSH username and password authenticate routine for a server. Please consider sshUserAuthenticateWithTypeFn.
- typedef int(∗ sshUserAuthenticateWithTypeFn) (const char ∗usernamePtr, const char ∗authValPtr, AuthType authType)

  User provided SSH user authenticate routine for a server.
- typedef int(∗ sshGetUserPwFn) (const NBString &usernamePtr, NBString &passwordPtr)

  User provided SSH user password authentication routine for clients.
- typedef int(∗ sshGetUserKeyFn) (const NBString &usernamePtr, NBString &publicKey, NBString &privateKey, NBString &keyType)

  User provided SSH user key authenticate routine for clients.
- typedef int(∗ sshUserGetKeyFn) (int keyRequested, const unsigned char ∗∗keyBufferPtr, int ∗keyLengthPtr)

  The user defined callback to get the server key used during the initial SSH negotiation.

Functions

- void SshSetUserAuthenticate (sshUserAuthenticateFn sshUserAuthenticateFnPtr)

  [DEPRECATED] Sets the user defined server authentication function. Please consider sshUserAuthenticateWith↩ TypeFn.
- sshUserAuthenticateFn SshGetUserAuthenticate (void)

  [DEPRECATED] Gets the user defined server authentication function. Please consider SshGetUserAuthenticate↩ WithType.
- void SshSetUserAuthenticateWithType (sshUserAuthenticateWithTypeFn sshUserAuthenticateFnPtr)

  Sets the user defined server authentication function.
- sshUserAuthenticateWithTypeFn SshGetUserAuthenticateWithType (void)

  Gets the user defined server authentication function..
- void SshClientSetGetUserPaswordFn (sshGetUserPwFn sshGetUserPwFnPtr)

  Sets the user defined client authentication function for getting user passwords during SSH authentication.
- sshGetUserPwFn SshClientGetUserPaswordFn (void)

  Gets the user defined client authentication function for getting a user password during authentication.
- void SshClientSetGetUserKeyFn (sshGetUserKeyFn sshGetUserKeyFnPtr)

  Sets the user defined client authentication function for getting user keys during SSH authentication.
- sshGetUserKeyFn SshClientGetUserKeyFn (void)

  Gets the user defined client authentication function for getting a user key during authentication.
- void SshSetUserGetKey (sshUserGetKeyFn sshUserGetKeyFnPtr)

  Sets the user defined callback method to provide the server key.
- sshUserGetKeyFn SshGetUserGetKey (void)

  Gets the user defined callback method to provide the server key.
- bool SshValidateKey (const char ∗candidateKey, int candidateKeySize, int ∗keyTypePtr, int key↩ Format=WOLFSSH_FORMAT_ASN1)

  Takes a key and returns if it's valid or not.
- bool SshWritePublicKey (int publicKeyFd, unsigned char ∗candidateKey, int candidateKeySize)

  Write public key to file descriptor. Takes both PEM and ANS1 formats.
- int NbSshInit ()

  Initializes the underlying SSH framework. This will start a background task used to handle negotiations and SSH traffic. This is automatically called by SshAccept and SshConnect, and doesn't need to be called directly except in special circumstances.
- int SshAccept (int listenFd, IPADDR ∗clientAddress, uint16_t ∗securePort, uint16_t timeout)

  Accepts and negotiates SSH session. Automatically calls NbSshInit() if required.
- int SshConnect (IPADDR clientAddress, uint16_t securePort, uint16_t localPort, uint16_t timeout, const char ∗username)

Issues a connect request to negotiates an SSH session. Automatically calls NbSshInit() if required.

- SshSocket * SshNegotiateSession (int fd)

  Negotiates an SSH server session on an open file descriptor.
- SshSocket * SshNegotiateSessionClient (int secureFd, const char *username)

  Negotiates an SSH client session on an open file descriptor.
- void SshPrintStatistics (int secureFd)

  Negotiates an SSH client session on an open file descriptor.
- int SshGetKeySize ()

  Determins and returns SSH's installed key size.
- int SshSetBannerText (const char *banner)

  Sets the banner text displayed by the SSH server on connection.
- int SshSetSockOption (int fd, int option)

  Set SSH TCP socket options.
- int SshClrSockOption (int fd, int option)

  Clear SSH TCP socket options.
- int SshGetSockOption (int fd)

  Returns the options for the specified SSH TCP socket.

21.91.1 Detailed Description

The NetBurner SSH Library

21.91.2 Typedef Documentation

21.91.2.1 sshGetUserKeyFn

```
typedef int(* sshGetUserKeyFn) (const NBString &usernamePtr, NBString &publicKey, NBString
&privateKey, NBString &keyType)
```

User provided SSH user key authenticate routine for clients.

Parameters

	usernamePtr	Username in plain text
out	*publicKey*	Used to return the corresponding public key for the user
out	*privateKey*	Used to return the corresponding pviate key for the user
out	*keyType*	Used to return the corresponding key type (ECC or RSA) for the user

Return values

>0	Pasword available for the user
<=0	Authentication failed

21.91.2.2 sshGetUserPwFn

```
typedef int(* sshGetUserPwFn) (const NBString &usernamePtr, NBString &passwordPtr)
```

User provided SSH user password authentication routine for clients.

Parameters

	usernamePtr	Username in plain text

Parameters

out	*passwordPtr*	Used to return the corresponding password for the user

Return values

>0	Pasword available for the user
<=0	Authentication failed

21.91.2.3 sshUserAuthenticateFn

```
typedef int(* sshUserAuthenticateFn) (const char *usernamePtr, const char *passwordPtr)
```

[DEPRECATED] User provided SSH username and password authenticate routine for a server. Please consider sshUserAuthenticateWithTypeFn.

Parameters

usernamePtr	Username in plain text
passwordPtr	Password in plain text

Return values

1	Authentication passed
!1	Authentication failed

21.91.2.4 sshUserAuthenticateWithTypeFn

```
typedef int(* sshUserAuthenticateWithTypeFn) (const char *usernamePtr, const char *authValPtr,
AuthType authType)
```

User provided SSH user authenticate routine for a server.

Parameters

usernamePtr	Username in plain text
authValPtr	The value that is being passed in for authentication
authType	Either a password or a key

Return values

1	Authentication passed
!1	Authentication failed

21.91.2.5 sshUserGetKeyFn

```
typedef int(* sshUserGetKeyFn) (int keyRequested, const unsigned char **keyBufferPtr, int *key↩
LengthPtr)
```

The user defined callback to get the server key used during the initial SSH negotiation.

Parameters

keyRequested	Type key requested (ECC or RSA)
keyBufferPtr	Buffer containing the key (ASN1 or PEM format supported)
keyLengthPtr	Size of the key in 8 bit bytes

Return values

0	Key and length are valid
-1	Key requested is not available

See also

SshSetUserGetKey()

SshGetUserGetKey()

21.91.3 Function Documentation

21.91.3.1 NbSshInit()

```
int NbSshInit ( )
```

Initializes the underlying SSH framework. This will start a background task used to handle negotiations and SSH traffic. This is automatically called by SshAccept and SshConnect, and doesn't need to be called directly except in special circumstances.

Return values

SSH_SUCCESS	if SSH system has been correctly initialized.
Other	SSH error code if initialization was unsuccessful.

See also

SshAccept()

SshConnect()

21.91.3.2 SshAccept()

```
int SshAccept (
            int listenFd,
            IPADDR * clientAddress,
            uint16_t * securePort,
            uint16_t timeout )
```

Accepts and negotiates SSH session. Automatically calls NbSshInit() if required.

Parameters

listenFd	File descriptor of listening socket
clientAddress	Address of client
securePort	Secure port of negotiated socket
timeout	Ticks to wait for connection, 0 is infinite

Return values

>0	The secure file descriptor if successful.
Other	A TCP or SSH error code, depending on the error..

See also

NbSshInit()

SshConnect()

21.91.3.3 SshClientGetUserKeyFn()

```
sshGetUserKeyFn SshClientGetUserKeyFn (
            void  )
```

Gets the user defined client authentication function for getting a user key during authentication.

Return values

sshGetUserPwFnPtr	Sets the user defined server authentication function.

See also

SshClientSetGetUserPaswordFn()

SshClientGetUserPaswordFn()

SshClientSetGetUserKeyFn()

21.91.3.4 SshClientGetUserPaswordFn()

```
sshGetUserPwFn SshClientGetUserPaswordFn (
            void  )
```

Gets the user defined client authentication function for getting a user password during authentication.

Return values

A	pointer to the fuction that is currently set for passing in a user's password during authentication.

See also

SshClientSetGetUserPaswordFn()

SshClientSetGetUserKeyFn()

SshClientGetUserKeyFn()

21.91.3.5 SshClientSetGetUserKeyFn()

```
void SshClientSetGetUserKeyFn (
            sshGetUserKeyFn sshGetUserKeyFnPtr )
```

Sets the user defined client authentication function for getting user keys during SSH authentication.

Parameters

sshGetUserKeyFnPtr	Sets the user defined server authentication function.

See also

SshClientSetGetUserPaswordFn()

SshClientGetUserPaswordFn()

SshClientGetUserKeyFn()

21.91.3.6 SshClientSetGetUserPaswordFn()

```
void SshClientSetGetUserPaswordFn (
            sshGetUserPwFn sshGetUserPwFnPtr )
```

Sets the user defined client authentication function for getting user passwords during SSH authentication.

Parameters

sshGetUserPwFnPtr	Sets the user defined client authentication function.

See also

SshClientGetUserPaswordFn()

SshClientSetGetUserKeyFn()

SshClientGetUserKeyFn()

21.91.3.7 SshClrSockOption()

```
int SshClrSockOption (
            int fd,
            int option )
```

Clear SSH TCP socket options.

Parameters

fd	Socket file descriptor.
option	Socket option to clear: TCP Socket Options.

Returns

A bitmask of the options for the specified socket

See also

SshSetSockOption()

SshGetSockOption()

21.91.3.8 SshConnect()

```
int SshConnect (
            IPADDR clientAddress,
            uint16_t securePort,
            uint16_t localPort,
            uint16_t timeout,
            const char * username )
```

Issues a connect request to negotiates an SSH session. Automatically calls NbSshInit() if required.

Parameters

clientAddress	Address of client
securePort	Secure port of negotiated socket
localPort	Optional parameter to specify a Local port number. Recommend this always be set to a value of 0 so a random local port number is used.
timeout	Ticks to wait for connection, 0 is infinite
username	Username to use in negotiation process of secure connection

Return values

>0	The secure file descriptor if successful.
0	Request timed out.
<0	A TCP or SSH error code, depending on the error..

See also

SshAccept()

NbSshInit()

21.91.3.9 SshGetKeySize()

```
int SshGetKeySize ( )
```

Determins and returns SSH's installed key size.

Returns

The installed key's size

21.91.3.10 SshGetSockOption()

```
int SshGetSockOption (
            int fd )
```

Returns the options for the specified SSH TCP socket.

Parameters

fd	Socket file descriptor.

Returns

A bitmask of the options for the specified socket

See also

SshSetSockOption()

SshClrSockOption()

21.91.3.11 SshGetUserAuthenticate()

```
sshUserAuthenticateFn SshGetUserAuthenticate (
            void  )
```

[DEPRECATED] Gets the user defined server authentication function. Please consider SshGetUserAuthenticate↩ WithType.

Return values

Returns	a pointer to the function fuction that is currently set for server user authentication.

See also

SshGetUserAuthenticateWithType

21.91.3.12 SshGetUserAuthenticateWithType()

```
sshUserAuthenticateWithTypeFn SshGetUserAuthenticateWithType (
            void  )
```

Gets the user defined server authentication function..

Return values

A	pointer to the fuction that is currently set for server user authentication.

See also

SshGetUserAuthenticateWithType()

21.91.3.13 SshGetUserGetKey()

```
sshUserGetKeyFn SshGetUserGetKey (
            void  )
```

Gets the user defined callback method to provide the server key.

Return values

sshUserGetKeyFn	The user defined callback

See also

sshUserGetKeyFn()

SshSetUserGetKey()

21.91.3.14 SshNegotiateSession()

```
SshSocket * SshNegotiateSession (
            int fd )
```

Negotiates an SSH server session on an open file descriptor.

Parameters

fd	File descriptor to use in the SSN negotiation, usually obtained with a call to accept()

Returns

A pointer to the SSH socket created on a successfull connection, or a nullptr if not successful.

21.91.3.15 SshNegotiateSessionClient()

```
SshSocket * SshNegotiateSessionClient (
            int secureFd,
            const char * username )
```

Negotiates an SSH client session on an open file descriptor.

Parameters

secureFd	File descriptor to use in the SSN negotiation
username	Username to use in negotiation process of secure connection

Returns

A pointer to the SSH socket created on a successfull connection, or a nullptr if not successful.

21.91.3.16 SshPrintStatistics()

```
void SshPrintStatistics (
            int secureFd )
```

Negotiates an SSH client session on an open file descriptor.

Parameters

secureFd	Secure file descriptor

21.91.3.17 SshSetBannerText()

```
int SshSetBannerText (
            const char * banner )
```

Sets the banner text displayed by the SSH server on connection.

Parameters

banner	The text to display to any connected clients

Return values

SSH_SUCCESS	If successful
<0	Error code on failure

21.91.3.18 SshSetSockOption()

```
int SshSetSockOption (
            int fd,
            int option )
```

Set SSH TCP socket options.

Parameters

fd	Socket file descriptor.
option	Socket option to set: TCP Socket Options.

Returns

A bitmask of the options for the specified socket

See also

SshClrSockOption()

SshGetSockOption()

21.91.3.19 SshSetUserAuthenticate()

```
void SshSetUserAuthenticate (
            sshUserAuthenticateFn sshUserAuthenticateFnPtr )
```

[DEPRECATED] Sets the user defined server authentication function. Please consider sshUserAuthenticateWith↩ TypeFn.

Parameters

sshUserAuthenticateFnPtr	Fuction to set

See also

SshSetUserAuthenticateWithType()

21.91.3.20 SshSetUserAuthenticateWithType()

```
void SshSetUserAuthenticateWithType (
            sshUserAuthenticateWithTypeFn sshUserAuthenticateFnPtr )
```

Sets the user defined server authentication function.

Parameters

sshUserAuthenticateFnPtr	Fuction to set

See also

SshSetUserAuthenticateWithType()

21.91.3.21 SshSetUserGetKey()

```
void SshSetUserGetKey (
            sshUserGetKeyFn sshUserGetKeyFnPtr )
```

Sets the user defined callback method to provide the server key.

Parameters

sshUserGetKeyFnPtr	The user defined callback

See also

sshUserGetKeyFn()

SshGetUserGetKey()

21.91.3.22 SshValidateKey()

```
bool SshValidateKey (
            const char * candidateKey,
            int candidateKeySize,
            int * keyTypePtr,
            int keyFormat = WOLFSSH_FORMAT_ASN1 )
```

Takes a key and returns if it's valid or not.

Parameters

	candidateKey	A pointer to a buffer containing the key
	candidateKeySize	The size of the key in bytes
out	*keyTypePtr*	Is set to what kind of key is passed it (ECC and RSA are valid)
	keyFormat	The key format (ASN1 or PEM)

Return values

true	The key is valid
false	The key is not valid

21.91.3.23 SshWritePublicKey()

```
bool SshWritePublicKey (
            int publicKeyFd,
            unsigned char * candidateKey,
            int candidateKeySize )
```

Write public key to file descriptor. Takes both PEM and ANS1 formats.

Parameters

publicKeyFd	Open target file descriptor
candidateKey	key in buffer
candidateKeySize	Size of key in bytes

Return values

true	A valid key was written to the file descriptor
false	An error occurred

21.92 SSH Error Codes

Macros

- #define **SSH_SUCCESS** (0)

  The function completed successfully.
- #define **SSH_ERROR_FAILED_SESSION_FAILED** (-300)

  Not currently used, kept for backward compatibility.
- #define **SSH_ERROR_FAILED_NEGOTIATION** (-301)

  The connection failed the SSH negotiation.
- #define **SSH_ERROR_FAILED_INITIALIZATION** (-302)

  The SSH system failed to initialize.
- #define **SSH_ERROR_FAILED_CONTEXT_INIT** (-303)

  Unable to instantiate SSH client/server context.
- #define **SSH_ERROR_BAD_KEY** (-304)

  The key provided was invalid.
- #define **SSH_ERROR_BAD_ARGUMENT** (-305)

  A bad argument was provided to the function.
- #define SSH_FAILED_KEY_CHECK (-306)

21.92.1 Detailed Description

Error codes that can be returned by the NetBurner SSH functions.

21.92.2 Macro Definition Documentation

21.92.2.1 SSH_FAILED_KEY_CHECK

```
#define SSH_FAILED_KEY_CHECK  (-306)
```

There were no keys designated to use for the SSH library, the onboard generated key is not valid, and auto-cert generation is not enabled or the cert generation function failed.

21.93 STD File System Seek Codes

Macros

- #define **FS_SEEK_SET** 0

  Beginning of file.
- #define **FS_SEEK_CUR** 1

  Current position of the file pointer.
- #define **FS_SEEK_END** 2

  End of file.

21.93.1 Detailed Description

The codes used when calling `f_seek()`.

21.94 Save Config Record Errors

Macros

- #define **SAVE_CONF_ERR_SUCCESS** 0

  Success.
- #define **SAVE_CONF_ERR_INVALID_CONFIG_NUM** -1

  Invalid configuration number.
- #define **SAVE_CONF_ERR_NOT_CONFIGURED** -2

  Not configured.
- #define **SAVE_CONF_ERR_UNKNOWN** 1

  Unknown error.

21.94.1 Detailed Description

21.95 Scan Request Errors

Macros

- #define **SCAN_ERR_SUCCESS** 0

  Success.
- #define **SCAN_ERR_MSG_LENGTH** -1

  Invalid message length.
- #define **SCAN_ERR_IN_PROGRESS** -2

  Scan in progress.
- #define **SCAN_ERR_SSID_LEN_LONG** -3

  SSID length too long.
- #define **SCAN_ERR_INVALID_TABLE** -4

  Invalid table.
- #define **SCAN_ERR_INVALID_OPTION** -5

  Invalid option.
- #define **SCAN_ERR_TOO_MANY_CHANNELS** -6

  Too many channels.
- #define **SCAN_ERR_UNKNOWN** 1

  Unknown error.

21.95.1 Detailed Description

21.96 Security Options

Macros

- #define **SEC_VALUE_OPEN** (0x00)

  Option list security value: Open network.
- #define **SEC_VALUE_WEP** (0x01)

  Option list security value: WEP.
- #define **SEC_VALUE_WPA** (0x02)

  Option list security value: WPA.
- #define **SEC_VALUE_WPA2** (0x03)

  Option list security value: WPA2.
- #define **SEC_VALUE_WPS** (0x04)

  Option list security value: WPS.
- #define **SEC_VALUE_ANY** (0xFE)

Option list security value: Any, used to connect as a client regardless of how security is configured for AP.

- #define **SEC_VALUE_UNKNOWN** (0xFF)

  Option list security value: Unknown security.

21.96.1 Detailed Description

21.97 Serial Interfaces

Modules

- Serial Port Error Codes

Macros

- #define **ADDR_ESCAPE_CHAR** (0xFF)

  Address escape character.
- #define SimpleOpenSerial(p, b) OpenSerial(p, b, 1, 8, eParityNone)

  Simple open a serial port.

Enumerations

- enum parity_mode {
  eParityNone , eParityOdd , eParityEven , eParityMulti ,
  eParityMultiOdd , eParityMultiEven }

  Serial Parity Modes.

Functions

- int OpenSerial (int portnum, unsigned int baudrate, int stop_bits, int data_bits, parity_mode parity)

  Open a serial port.
- int OpenDefaultSerial ()

  Opens the Default serial port as defined by the Boot Config settings.
- void SerialExpandRxBuffer (int fd, int nb)

  Expand the received serial buffer.
- int SerialClose (int portnum)

  Close a serial port.
- void SerialEnableTxFlow (int port, int enab)

  Enable transmit software flow control on the specified UART.
- void SerialEnableRxFlow (int port, int enab)
- void SerialEnableHwTxFlow (int port, int enab)
- void SerialEnableHwRxFlow (int port, int enab)
- void Serial485HalfDupMode (int port, int enab)
- void SendBreak (int port, uint32_t time)
- int serwriteaddress (int fd, const char c)
- int GetUartErrorReg (int fd)
- void SetRTS (int port, bool val)
- BOOL SerialSendComplete (int fd)

21.97.1 Detailed Description

The NetBurner Serial

21.97.2 Macro Definition Documentation

21.97.2.1 SimpleOpenSerial

```
#define SimpleOpenSerial(
            p,
            b ) OpenSerial(p, b, 1, 8, eParityNone)
```

Simple open a serial port.
Select the UART number and baud rate, default parameters will be 1 stop bit, no parity, 8 data bits.

Parameters

p	The UART to open. First port is 0. The maximum number is platform dependent
b	The speed of the serial port in bits per second

See also

OpenSerial()

21.97.3 Enumeration Type Documentation

21.97.3.1 parity_mode

```
enum parity_mode
```

Serial Parity Modes.

Enumerator

eParityNone	No parity.
eParityOdd	Odd parity.
eParityEven	Even parity.
eParityMulti	Multi parity.
eParityMultiOdd	Multi mode, Odd parity.
eParityMultiEven	Multi mode, Even parity.

21.97.4 Function Documentation

21.97.4.1 GetUartErrorReg()

```
int GetUartErrorReg (
            int fd )
```

Gets the UART error register. This is only applicable on select platforms. Any errors are added to this register in a logical OR operation. Calling this function to read the error status will clear the register.
Bit Description
3 Received break 2 Framing error 1 Parity error 0 Overrun error
fd - The file descriptor associated with the UART serial port whose error register is to be retrieved.
return - The value of the error register if successful (positive int - the bitwise values are represented by the UART↩
ERR∗ definitions found near the top of this file); otherwise, one of the error codes is returned: SERIAL_ERR_↩
NOSUCH_PORT (-1) SERIAL_ERR_PORT_NOTOPEN (-2)

21.97.4.2 OpenDefaultSerial()

```
int OpenDefaultSerial ( )
```

Opens the Default serial port as defined by the Boot Config settings.

This function opens the Default serial port as defined by the Boot Config settings. It is called automatically by the system as part of the system initialization process. Unlike OpenSerial and SimpleOpenSerial, OpenDefaultSerial will configure the Tx and Rx pins to the Boot Config settings.

Returns

The file descriptor of the default serial port (> 0), or Serial Port Error Codes

See also

OpenSerial()

21.97.4.3 OpenSerial()

```
int OpenSerial (
            int portnum,
            unsigned int baudrate,
            int stop_bits,
            int data_bits,
            parity_mode parity )
```

Open a serial port.

This function opens a serial port. Note that functions to open a serial port, such as OpenSerial() and SimpleOpenSerial(), must be called before any functions or pin assignments that configure the operation of the serial port. You must also configure all pins that have the same UART capability. For example, if UART 1 TX can be configured to come out on more than one pin (as a second or third alternate function), you must set those pins to some function other than UART 1 TX. This is because the functions that open a serial port will configure all serial port functions to default operation and default pins. For example, hardware flow control will be off, and the default pin assignments will be used on those platform that have multiple outputs.

Parameters

portnum	The UART to open. First port is 0. The maximum number is platform dependent
baudrate	The speed of the serial port in bits per second
stop_bits	The number of stop bits, 1 or 2
data_bits	The number of data bits sent per character or frame: 5, 6, 7 or 8
parity	The type of parity checking to use: eParityNone, eParityEven, eParityOdd or eParityMulti

Returns

The file descriptor of the serial port (> 0), or Serial Port Error Codes

See also

SimpleOpenSerial()

21.97.4.4 SendBreak()

```
void SendBreak (
            int port,
            uint32_t time )
```

Sets a break in the UART transmission for a given period of time. The break starts when character transmission completes. The break is delayed until any character in the transmitter shift register is sent. Any character in the transmitter holding register is sent after the break.

Note: This feature may not be supported on all UART modules. Please refer to the processor's datasheet for details on supported features.

port - The UART whose transmitter will be forced low (start break). time - Specifies the amount of time in ticks that the break will hold; when time expires, the break will be stopped; 20 ticks equal 1 second by default.
return - Nothing to return.

21.97.4.5 Serial485HalfDupMode()

```
void Serial485HalfDupMode (
            int port,
            int enab )
```

Enables or disables RS-485 half-duplex mode. Full-duplex mode is automatically enabled when half-duplex mode is disabled. This must be explicitly called before RS-485 functionality can be used. Jumpers may also be needed depending on the hardware device and/or development board used.
port - The UART port to use; the UART that can only be used for RS-485 depends on the platform (e.g., MOD52xx mounted on the MOD-DEV-100 development board use only UART 0 for RS-485, CB34-EX use only UART 1 for RS-485). enab - '0' disables half-duplex (enables full-duplex); '1' enables half- duplex (disables full-duplex)
return - Nothing to return.

21.97.4.6 SerialClose()

```
int SerialClose (
            int portnum )
```

Close a serial port.

Parameters

portnum	The UART to close. First port is 0. The maximum number is platform dependent

Returns

0 if successful, otherwise Serial Port Error Codes

See also

OpenSerial()

21.97.4.7 SerialEnableHwRxFlow()

```
void SerialEnableHwRxFlow (
            int port,
            int enab )
```

Enables or disables request-to-send hardware flow control on receive (RxRTS). When enabled, receive throttles the transmitter on the other end. Jumpers may also be needed to enable the RxRTS line depending on the device and/or development board used.
(NB Device) Rx <–<–< Tx (Other Device) (NB Device) RxRTS >-->--> TxCTS (Other Device)
port - The UART whose flow control will be toggled. enab - '0' disables flow control; '1' (or any non-zero number) will enable it.
return - Nothing to return.

21.97.4.8 SerialEnableHwTxFlow()

```
void SerialEnableHwTxFlow (
            int port,
            int enab )
```

Enables or disables clear-to-send hardware flow control on transmit (TxCTS). When enabled, transmit is throttled by the receiver on the other end. Jumpers may also be needed to enable the TxCTS line depending on the device and/or development board used.
(NB Device) Tx >-->--> Rx (Other Device) (NB Device) TxCTS <–<–< RxRTS (Other Device)

Note: Calling this function to disable Tx hardware flow control after enabling RS-485 full-duplex mode configures for RS-422 mode (Tx stays actively driven).
port - The UART whose flow control will be toggled. enab - '0' disables flow control; '1' (or any non-zero number) will enable it.
return - Nothing to return.

21.97.4.9 SerialEnableRxFlow()

```
void SerialEnableRxFlow (
            int port,
            int enab )
```

Enables or disables software flow control (XON/XOFF) on receive. When enabled, the NetBurner device will send special XON and XOFF characters to another device in order to control the flow of incoming data.
port - The UART whose flow control will be toggled. enab - '0' disables flow control; '1' (or any non-zero number) will enable it.
return - Nothing to return.

21.97.4.10 SerialEnableTxFlow()

```
void SerialEnableTxFlow (
            int port,
            int enab )
```

Enable transmit software flow control on the specified UART.
Enables or disables software flow control (XON/XOFF) on transmit. When enabled, the NetBurner device will recognize the special XON and XOFF characters being sent from another device in order to throttle the output.

Parameters

port	UART number
enab	0 disables flow control, 1 enables flow control

21.97.4.11 SerialExpandRxBuffer()

```
void SerialExpandRxBuffer (
            int fd,
            int nb )
```

Expand the received serial buffer.
This function expands the number of buffer descriptors the specific serial port is allowed to use. Each buffer descriptor counts for 1500 bytes fo setting this to 10 would make the serial Rx buffer 15000 bytes.

21.97.4.12 SerialSendComplete()

```
BOOL SerialSendComplete (
            int fd )
```

Determines if data waiting in transmitter holding/shift registers are already sent out.
return - True if finished, false if data still sending

21.97.4.13 serwriteaddress()

```
int serwriteaddress (
            int fd,
            const char c )
```

Sends an address character via the UART port number associated with the given file descriptor. This function can only be utilized if the UART is initialized in multidrop mode.
fd - The file descriptor associated with the UART serial port that will be used. c - The address character to be sent.

return - '1' if successful; otherwise, one of the error codes is returned (note that SERIAL_ERR_PORT_NOTOPEN is also returned if UART is not initialized to be in multidrop mode): SERIAL_ERR_NOSUCH_PORT (-1) SERIAL↩ _ERR_PORT_NOTOPEN (-2)

21.97.4.14 SetRTS()

```
void SetRTS (
            int port,
            bool val )
```

Sets or clears the Request To Send signal for the specified serial port.
port - The UART port whose RTS signal will be set or cleared. (0 or 1) val - The Boolean value to configure the signal; TRUE sets it, while FALSE clears it.
return - Nothing to return.

21.98 Serial Port Error Codes

Macros

- #define **SERIAL_ERR_NOSUCH_PORT** (-1)

  Port number does not exist.
- #define **SERIAL_ERR_PORT_NOTOPEN** (-2)

  Port is not open.
- #define **SERIAL_ERR_PORT_ALREADYOPEN** (-3)

  Port is already open.
- #define **SERIAL_ERR_PARAM_ERROR** (-4)

  Parameter error.

21.98.1 Detailed Description

21.99 Shutdown Notifications

Modules

- ShutdownReasons

  Shutdown Reasons.

Functions

- bool NBApproveShutdown (int reason)

  Approve action that will result in a reboot.
- void NBFaultNotify ()

  Emergency notification of a fault shutdown.

21.99.1 Detailed Description

The NBApproveShutdown callback function can be used by an application to put the system in a safe state before an application update, configuration update, or reboot event occurs. For example, closing active TCP sockets, ensuring Flash and/or file system write operations are complete, and putting critical peripherals in a safe state. The NBApproveShutdown() function as a weak reference to a system function that always returns true by default. If an application creates its own function using the same signature, that function will be used instead.
A reason for the reboot request is passed to the function. The system will automatically call NBapproveShutdown() for the following reasons:

```
#define SHUTDOWN_CODEUPDATE         (1)   // A code update is requested
#define SHUTDOWN_CONFIGURE_REBOOT   (2)   // Configuration values have been modified with a requested reboot
```

An application can choose to ignore the parameters or add its own. For custom reasons, the application should call NBApproveShutdown() with the appropriate reason:

```
// Custom reboot reason. System reasons start at 1, so pick something much larger
```

```
#define SHUTDOWN_CUSTOM_REBOOT 100

if( NBApproveShutdown(SHUTDOWN_CUSTOM_REBOOT))
{
  OSTimeDly(TICKS_PER_SECOND * 5);
  ForceReboot();
}
```

21.99.2 Function Documentation

21.99.2.1 NBApproveShutdown()

```
bool NBApproveShutdown (
            int reason )
```

Approve action that will result in a reboot.

Note

Called for application updates or configuration changes that include the reboot option

This function will be called when the configuration values have changed with the option to reboot, or for an application update (always requires a reboot).
If the process fails at some time after this notification is approved, the system may not execute a reboot. This function is satisfied as a weak reference in the NetBurner library. In order to use it in your application, create a function of the same name (callback function).

Parameters

reason	The reason for the shutdown request (ShutdownReasons)

Return values

true	Ok to proceed with the modification and reboot
false	Abort the update

21.99.2.2 NBFaultNotify()

```
void NBFaultNotify ( )
```

Emergency notification of a fault shutdown.

Note

Called after a critical error, such as a trap or memory fault before a reboot

If the system has a critical fault of some kind it will call this function before rebooting. Since this is a sever critical fault, the function is being called in an exception context with a very small stack. It should execute whatever is necessary and return immediately with a minimal number of variables.
It is extremely limited in the actions it can perform. It cannot execute:

- Any RTOS functions
- printf family functions
- Writes to network sockets
- TCP or UDP functions
- Writes to serial ports
- RTOS delay functions

- Writes to Flash memory
- File system operations

Since this is a critical fault condition and the system is unstable, writing directly to I/O registers has the most chance of success.

21.100 ShutdownReasons

Shutdown Reasons.

Macros

- #define **SHUTDOWN_CODEUPDATE** (1)

  A code update is requested.
- #define **SHUTDOWN_CONFIGURE_REBOOT** (2)

  Configuration values have been modified with a requested reboot.

21.100.1 Detailed Description

Shutdown Reasons.

21.101 Signed Application Update

Modules

- NBUpdate Function Return Values

Functions

- void RegisterAppSigningPublicKey (const char *pKey)

  Enable APP Signing by registering a RSA PEM or DER format public key.
- int ProgramApplication (uint32_t where, uint8_t *pAppImage)

  Program an appliction image into Flash memory.
- int UpdateFromStream (int fd, AppUpdateRecord *&pu, uint32_t timeout)

  Program/update an appliction image from a data stream.

21.101.1 Detailed Description

21.101.2 Function Documentation

21.101.2.1 ProgramApplication()

```
int ProgramApplication (
            uint32_t where,
            uint8_t * pAppImage )
```

Program an appliction image into Flash memory.

Parameters

where	Location in Flash memory
pAppImage	Pointer to the application image to program

Return values

0	Success
-1	Application image pointer invalid
-2	Invalid platform header

21.101.2.2 RegisterAppSigningPublicKey()

```
void RegisterAppSigningPublicKey (
            const char * pKey )
```

Enable APP Signing by registering a RSA PEM or DER format public key.
This function should be called BEFORE the init() function if you want to guarantee security. It is ok to register a temporary text blob early, then overwrite it once the key via other means.

Parameters

pKey	Pointer to the public key to register

21.101.2.3 UpdateFromStream()

```
int UpdateFromStream (
            int fd,
            AppUpdateRecord *& pu,
            uint32_t timeout )
```

Program/update an appliction image from a data stream.

Parameters

fd	File descriptor of data stream
pu	Application record
timeout	Timeout value in system time ticks (eg TICKS_PER*SECOND * 10)

Return values

0	Success
-1	Application image pointer invalid
-2	Invalid platform header

21.102 Stopwatch Timer

Classes

- class StopWatch

  Stopwatch for timing events.

21.102.1 Detailed Description

The StopWatch class can be used to create a StopWatch object that can time events. Please refer to the Stopwatch example in `\nburn\examples`
Example

```
StopWatch myStopwatch;

myStopwatch.Start();
for ( volatile int i = 0; i < numLoop; i++ )
{
   count++;
}
myStopwatch.Stop();
```

See also

Interval Timer

High Resolution Delay Timer

OSTimeDly()

21.103 Stream Update

Modules

- Stream Update Return Values

Functions

- int ReadBinaryApplicationCodeFromStream (int fd)

  Read a binary application image from an input stream and program flash memory.
- int ReadS19ApplicationCodeFromStream (int fd)

  Read an ASCII application image in s-record .s19 format from an input stream and program flash memory.

21.103.1 Detailed Description

Functions to read an application form a source with a file descriptor. Useful for programming/updating an application image from sources such as flash memory cards, posts from other devices, and files received from FTP.

21.103.2 Function Documentation

21.103.2.1 ReadBinaryApplicationCodeFromStream()

```
int ReadBinaryApplicationCodeFromStream (
            int fd )
```

Read a binary application image from an input stream and program flash memory.
Read a binary application image from an input stream and program the application image in flash memory. Flash memory will not be modified unless the entire application is received without error.
A reboot is required to run the new application. Before any reboot, the running application should free, close and clean up any items necessary to put the device in a safe state. For example, it would be better to close active TCP or FTP connections rather than just reset so that any connected clients can close normally. If you wish to reboot from a running application you can use the
`void ForceReboot()`
function ForceReboot().

Parameters

fd	The File Descriptor to read the input data stream from

Returns

Stream Update Return Values

21.103.2.2 ReadS19ApplicationCodeFromStream()

```
int ReadS19ApplicationCodeFromStream (
            int fd )
```

Read an ASCII application image in s-record .s19 format from an input stream and program flash memory.
Read a ASCII application image in s-record .s19 format from an input stream and program the application image in flash memory. Flash memory will not be modified unless the entire application is received without error.
A reboot is required to run the new application. Before any reboot, the running application should free, close and clean up any items necessary to put the device in a safe state. For example, it would be better to close active TCP or FTP connections rather than just reset so that any connected clients can close normally. If you wish to reboot from a running application you can use the
`void ForceReboot()`
function ForceReboot().

Parameters

fd	The File Descriptor to read the input data stream from

Returns

Stream Update Return Values

21.104 Stream Update Return Values

Macros

- #define **STREAM_UP_FAIL** (0)

  Action failed.
- #define **STREAM_UP_OK** (1)

  Action succeeded.

21.104.1 Detailed Description

21.105 System Functions

NetBurner General System Functions.

Functions

- int SaveUserParameters (void *pCopyFrom, int len)

  Save data to the on-chip flash memory User Parameter area.
- void * GetUserParameters (void)

  Returns a void pointer to the user parameter area.
- const char * GetReleaseTag ()

  Returns the NNDK release tag information.

21.105.1 Detailed Description

NetBurner General System Functions.

21.105.2 Function Documentation

21.105.2.1 GetReleaseTag()

```
const char * GetReleaseTag ( )
```

Returns the NNDK release tag information.

Returns

Pointer to a string representing the development tools release tag

21.105.2.2 GetUserParameters()

```
void * GetUserParameters (
            void  )
```

Returns a void pointer to the user parameter area.
On most platforms the total amount of space is 8k bytes. Typically used to retrieve a structure.
This function is here for reverse compatibility with releases prior to NetBurner 3.0. Alternatives are to use the 3.0 configuration tree, or the EFFS STD file system.

Returns

Void pointer to the User Parameter area in flash memory

See also

SaveUserParameters()

21.105.2.3 SaveUserParameters()

```
int SaveUserParameters (
            void * pCopyFrom,
            int len )
```

Save data to the on-chip flash memory User Parameter area.
On most platforms the total amount of space is 8k bytes. Typically used to save a structure.
This function is here for reverse compatibility with releases prior to NetBurner 3.0. Alternatives are to use the 3.0 configuration tree, or the EFFS STD file system.

Parameters

pCopyFrom	Pointer to the data to copy and save
len	Length of data

Returns

The number of items copied, or <= 0 on failure

See also

GetUserParameters()

21.106 TCP

Modules

- TCP Notify
- TCP Socket Options
- TCP Socket State
- TCP Socket Status

Functions

- int accept (int listening_socket, IPADDR *address, uint16_t *port, uint16_t timeout)

  Accept an incoming connection from a listening socket.
- int connect (const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout)

  Make an outgoing TCP connection to a remote host.
- int connectvia (const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, const IPADDR &localIp↩ Address)

  Make an outgoing TCP connection to a remote host using the specified local interface IP address.
- int connectvia (const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, int ifnum)

  Make an outgoing TCP connection to a remote host using the specified local interface number.
- int connectwlocal (const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort, uint32_t timeout, const IPADDR &localIpAddress=IPADDR::NullIP(), int intf=-1)

  Make an outgoing TCP connection to a remote host using the specified local interface number or IP address.
- int listen (const IPADDR &addr, uint16_t port, uint8_t maxpend=5)

  Listen for incoming connections on the specified network interface IP address.
- int listenvia (const IPADDR &addr, uint16_t port, int ifn, uint8_t maxpend=5)

  Listen for incoming connections on the specified network interface IP address.
- int listenvia (const IPADDR &addr, uint16_t port, const IPADDR &localIpAddress, uint8_t maxpend=5)

  Listen for incoming connections on the specified network interface IP address.
- int NoBlockConnect (const IPADDR &ipAddress, uint16_t remotePort)

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
- int NoBlockConnectVia (const IPADDR &ipAddress, uint16_t remotePort, const IPADDR &interfaceIp↩ Address=IPADDR::NullIP())

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
- int NoBlockConnectVia (const IPADDR &ipAddress, uint16_t remotePort, int ifnum)

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
- int NoBlockConnectwlocal (const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort, IPADDR interfaceIpAddress=IPADDR::NullIP(), int ifn=-1)

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
- IPADDR GetSocketRemoteAddr (int fd)

  Returns the IP address of the remote host associated with the specified file descriptor.
- IPADDR GetSocketLocalAddr (int fd)

  Returns the IP address of the local interface associated with the connection.
- uint16_t GetSocketRemotePort (int fd)

  Returns the port number of the remote host associated with the connection.
- uint16_t GetSocketLocalPort (int fd)

  Returns the local port number associated with the connection.
- uint32_t TcpGetLastRxTime (int fd)

  Returns the value of system Time Ticks when the last packet was received. Used for the TCP Keep Alive feature.
- void TcpSendKeepAlive (int fd)

  Send a TCP keep alive packet to a remote host.

- uint32_t TcpGetLastRxInterval (int fd)

  Returns the number of system Time Ticks since the last packet was received. This is the difference between the current system time and lastRxTime of the socket.
- int GetTcpRtxCount (int fd)

  Returns the number of re-transmits that have occurred on the specified connection.
- uint8_t SetOutOfOrderBuffers (int fd, uint8_t max)

  Set the maximum number of out-of-order TCP buffers for the specified TCP socket.
- int setsockoption (int fd, int option)

  Set TCP socket options.
- int clrsockoption (int fd, int option)

  Clear TCP socket options.
- int getsockoption (int fd)

  Returns the options for the specified TCP socket.
- int SetSocketUnackBuffers (int fd, uint8_t val)

  Set the maximum number of outbound TCP buffers in the transmit un-acknowledged list for the specified TCP socket.
- int SetSocketRxBuffers (int fd, int n)

  Set the number of TCP receive buffers for the specified TCP socket.
- int SetSocketTxBuffers (int fd, int n)

  Set the number of TCP transmit buffers for the specified TCP socket.
- int abortsocket (int fd)

  Execute an abort on the specified TCP socket.
- int SockReadWithTimeout (int fd, char *buf, int nbytes, uint32_t timeout)

  Attempt to read from a TCP socket until the timeout expires.
- char SocketPeek (int fd)

  Returns the next char that would be read, 0 if no data.
- int TcpGetSocketInterface (int fd)

  Return the network interface associated with a TCP socket.
- uint8_t TcpGetSocketState (int fd)

  Return the current state of a TCP socket.
- uint16_t TcpGetRxBufferSpaceUsed (int fd)

  Returns the number of bytes used in a socket's RX buffer.
- uint16_t TcpGetTxBufferAvailSpace (int fd)

  Returns the number of bytes available in a socket's TX buffer.
- uint16_t TcpGetTxDataWaiting (int fd)

  Returns the number of bytes waiting to be sent in a socket's TX Buffer.
- BOOL TcpAllDataAcked (int socket)

  Check the data acknowledged state of a socket.
- BOOL WaitForSocketFlush (int fd, uint32_t ticks)

  Wait for a socket flush operation to complete. A socket is flushed if all sent data has been acknowledged.

21.106.1 Detailed Description

21.106.2 Function Documentation

21.106.2.1 abortsocket()

```
int abortsocket (
            int fd )
```

Execute an abort on the specified TCP socket.

Parameters

fd	Socket file descriptor

Returns

TCP Socket Status

21.106.2.2 accept()

```
int accept (
            int listening_socket,
            IPADDR * address,
            uint16_t * port,
            uint16_t timeout )
```

Accept an incoming connection from a listening socket.
A socket must be listening in order to accept a TCP connection, and a connection request must be accepted before it can be used. Each time a connection request is accepted, a slot is freed in the listen pending queue for that socket.

Parameters

listening_socket	The listening socket from which to accept the connection.
∗*address*	Pointer to an IPADDR object that can store the source IP address of the connection. Can be NULL.
∗*port*	Pointer to a variable to store the source port number of the connection. Can be NULL.
timeout	Number of system Time Ticks to wait. A value of 0 waits forever. For example, TICKS_PER_SECOND ∗ 5 waits five seconds.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

listen(), close()

21.106.2.3 clrsockoption()

```
int clrsockoption (
            int fd,
            int option )
```

Clear TCP socket options.

Parameters

fd	Socket file descriptor.
option	Socket option to clear: TCP Socket Options.

Returns

A bitmask of the options for the specified socket.

See also

setsockoption(), getsockoption(), SetSocketUnackBuffers(), SetSocketRxBuffers(), SetSocketTxBuffers()

21.106.2.4 connect()

```
int connect (
            const IPADDR & ipAddress,
            uint16_t remotePort,
            uint32_t timeout )  [inline]
```

Make an outgoing TCP connection to a remote host.

Parameters

ipAddress	Remote IP address.
remotePort	Remote port number.
timeout	Timeout in system time ticks (use TICKS_PER_SECOND macro).

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

connectvia(), connectwlocal()

21.106.2.5 connectvia() [1/2]

```
int connectvia (
            const IPADDR & ipAddress,
            uint16_t remotePort,
            uint32_t timeout,
            const IPADDR & localIpAddress )  [inline]
```

Make an outgoing TCP connection to a remote host using the specified local interface IP address.
The connect() function automatically uses the default network interface. The connectvia() function enables the application to specify a particular local network interface.

Parameters

ipAddress	Remote IP address .
remotePort	Remote port number.
timeout	Timeout in system time ticks (use TICKS_PER_SECOND macro).
localIpAddress	IP address of the local interface. If a value of IPADDR::NULL() is used, it will attempt to use the local interface associated with the remote IP address.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

connect(), connectwlocal()

21.106.2.6 connectvia() [2/2]

```
int connectvia (
            const IPADDR & ipAddress,
            uint16_t remotePort,
            uint32_t timeout,
            int ifnum )  [inline]
```

Make an outgoing TCP connection to a remote host using the specified local interface number.
The connect() function automatically uses the default network interface. The connectvia() function enables the application to specify a particular local network interface.

Parameters

ipAddress	Remote IP address.
remotePort	Remote port number.
timeout	Timeout in system time ticks (use TICKS_PER_SECOND macro).
ifnum	The local interface number. The interface number will override all routing based on the source and destination IP address.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

connect(), connectwlocal()

21.106.2.7 connectwlocal()

```
int connectwlocal (
            const IPADDR & ipAddress,
            uint16_t localPort,
            uint16_t remotePort,
            uint32_t timeout,
            const IPADDR & localIpAddress = IPADDR::NullIP(),
            int intf = -1 )  [inline]
```

Make an outgoing TCP connection to a remote host using the specified local interface number or IP address.

Warning

Although this function that allows a local port number parameter for extreme edge cases, local port numbers should be avoided. Local port numbers create connectivity failures when a remote host has a half open socket condition.

Parameters

ipAddress	Remote IP address.
localPort	Optional parameter to specify a Local port number. Recommend this always be set to a value of 0 so a random local port number is used. Warning! Specifying a local port number will create connectivity failures if the remote host has a half open socket condition.
remotePort	Remote port number.
timeout	Timeout in system time ticks (use TICKS_PER_SECOND macro).
localIpAddress	Optional IP address of local interface to use for connection. If not used, such as when specifying the intf parameter, set to IPADDR::NullIP().
intf	Optional local interface number to use for connection. Do not specify if the interfaceIpAddress parameter is used.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

connect(), connectvia()

21.106.2.8 GetSocketLocalAddr()

```
IPADDR GetSocketLocalAddr (
            int fd )  [inline]
```

Returns the IP address of the local interface associated with the connection.

Parameters

fd	The socket file descriptor.

Returns

The IP address of the remote host.

See also

GetSocketRemoteAddr()

21.106.2.9 GetSocketLocalPort()

```
uint16_t GetSocketLocalPort (
            int fd )
```

Returns the local port number associated with the connection.

Parameters

fd	The socket file descriptor.

Returns

The local port number for the connection.

See also

GetSocketRemotePort()

21.106.2.10 GetSocketRemoteAddr()

```
IPADDR GetSocketRemoteAddr (
            int fd )  [inline]
```

Returns the IP address of the remote host associated with the specified file descriptor.

Parameters

fd	The socket file descriptor.

Returns

The IP address of the remote host.

See also

GetSocketLocalAddr()

21.106.2.11 GetSocketRemotePort()

```
uint16_t GetSocketRemotePort (
            int fd )
```

Returns the port number of the remote host associated with the connection.

Parameters

fd	The socket file descriptor.

Returns

The remote host port number.

See also

GetSocketLocalPort()

21.106.2.12 getsockoption()

```
int getsockoption (
            int fd )
```

Returns the options for the specified TCP socket.

Parameters

fd	Socket file descriptor.

Returns

A bitmask of the options for the specified socket.

See also

setsockoption(), clrsockoption(), SetSocketUnackBuffers(), SetSocketRxBuffers(), SetSocketTxBuffers()

21.106.2.13 GetTcpRtxCount()

```
int GetTcpRtxCount (
            int fd )
```

Returns the number of re-transmits that have occurred on the specified connection.

Parameters

fd	The socket file descriptor.

Returns

The number of system Time Ticks since the last packet was received on the connection.

21.106.2.14 listen()

```
int listen (
             const IPADDR & addr,
             uint16_t port,
             uint8_t maxpend = 5 )  [inline]
```

Listen for incoming connections on the specified network interface IP address.
Begin listening for incoming connections. A connection must be accepted with the accept() function before they can be used. Each time a connection request is accepted is frees up another slot in the pending queue.

Parameters

addr	The address from which to accept connections. INADDR_ANY will accept all connections.
port	The network port number to listen on.
maxpend	The maximum number of pending connection requests allowed for the listen socket.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

```
int32_t fdListen = listen(INADDR_ANY, 1234);
```

See also

accept(), listenvia(), close()

21.106.2.15 listenvia() [1/2]

```
int listenvia (
             const IPADDR & addr,
             uint16_t port,
             const IPADDR & localIpAddress,
             uint8_t maxpend = 5 )  [inline]
```

Listen for incoming connections on the specified network interface IP address.
Begin listening for incoming connections. A connection must be accepted with the accept() function before they can be used. Each time a connection request is accepted is frees up another slot in the pending queue.

Parameters

addr	The address from which to accept connections. INADDR_ANY will accept all connections.
port	The network port number to listen on.
maxpend	The maximum number of pending connection requests allowed for the listen socket.
localIpAddress	The IP address of the network interface to listen on. If two interfaces have the same address, the lower numbered interface will be used.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

accept(), listen(), close()

21.106.2.16 listenvia() [2/2]

```
int listenvia (
            const IPADDR & addr,
            uint16_t port,
            int ifn,
            uint8_t maxpend = 5 )  [inline]
```

Listen for incoming connections on the specified network interface IP address.
Begin listening for incoming connections. A connection must be accepted with the accept() function before they can be used. Each time a connection request is accepted is frees up another slot in the pending queue.

Parameters

addr	The address from which to accept connections. INADDR_ANY will accept all connections.
port	The network port number to listen on.
maxpend	The maximum number of pending connection requests allowed for the listen socket.
ifn	Interface Number of the network Interface to be used.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

accept(), listen(), close()

21.106.2.17 NoBlockConnect()

```
int NoBlockConnect (
            const IPADDR & ipAddress,
            uint16_t remotePort )  [inline]
```

Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.

Warning

To free system resources a close() must be called, even if the current state is TCP_STATE_CLOSING. Otherwise a resource leak will occur.

Parameters

ipAddress	Remote IP address.
remotePort	Remote port number.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

NoBlockConnectVia(), TcpGetSocketState(), writeavail(), connect(), connectvia(), connectwlocal()

21.106.2.18 NoBlockConnectVia() [1/2]

```
int NoBlockConnectVia (
            const IPADDR & ipAddress,
            uint16_t remotePort,
            const IPADDR & interfaceIpAddress = IPADDR::NullIP() )  [inline]
```

Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
This function is the similar to NoBlockConnect(), with the additional capability of specifying the local interface IP address to use the for connection.

Warning

To free system resources a close() must be called, even if the current state is TCP_STATE_CLOSING. Otherwise a resource leak will occur.

Parameters

ipAddress	Remote IP address
remotePort	Remote port of socket
interfaceIpAddress	Optional parameter to specify which local network interface to use for the connection. The default is IPADDR::NullIP(), which will use the default local interface.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

NoBlockConnect(), TcpGetSocketState(), writeavail(), connect(), connectvia(), connectwlocal()

21.106.2.19 NoBlockConnectVia() [2/2]

```
int NoBlockConnectVia (
            const IPADDR & ipAddress,
            uint16_t remotePort,
            int ifnum )  [inline]
```

Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
This function is the similar to NoBlockConnect(), with the additional capability of specifying the local interface number to use the for connection.

Warning

To free system resources a close() must be called, even if the current state is TCP_STATE_CLOSING. Otherwise a resource leak will occur.

Parameters

ipAddress	Remote IP address.
remotePort	Remote port number.
ifnum	Optional parameter to specify which local network interface to use for the connection by Interface Number. Will override any routing based on destination.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

NoBlockConnect(), TcpGetSocketState(), writeavail(), connect(), connectvia(), connectwlocal()

21.106.2.20 NoBlockConnectwlocal()

```
int NoBlockConnectwlocal (
            const IPADDR & ipAddress,
            uint16_t localPort,
            uint16_t remotePort,
            IPADDR interfaceIpAddress = IPADDR::NullIP(),
            int ifn = -1 )  [inline]
```

Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
This function is the similar to NoBlockConnect(), with the additional capability of specifying the local interface number or IP address to use the for connection. If not specified, it will the default local interface.

Warning

To free system resources a close() must be called, even if the current state is TCP_STATE_CLOSING. Otherwise a resource leak will occur.

Although this function that allows a local port number parameter for extreme edge cases, local port numbers should be avoided. Local port numbers create connectivity failures when a remote host has a half open socket condition.

Parameters

ipAddress	IP address of the remote host.
localPort	Optional parameter to specify a Local port number. Recommend this always be set to a value of 0 so a random local port number is used. **Warning! Specifying a local port number will create connectivity failures if the remote host has a half open socket condition.**
remotePort	Remote port number.
interfaceIpAddress	Optional IP address of local interface to use for connection. If not used, such as when specifying the ifn parameter, set to IPADDR::NullIP().
ifn	Optional parameter to specify which local network interface to use for the connection by Interface Number. Will override any routing based on destination or interface IP address.

Returns

A value greater than 0 on success, representing the socket file descriptor. A value less than 0 on error, TCP Socket Status.

See also

TcpGetSocketState(), NoBlockConnect(), NoBlockConnectVia()

21.106.2.21 SetOutOfOrderBuffers()

```
uint8_t SetOutOfOrderBuffers (
            int fd,
            uint8_t max )  [inline]
```

Set the maximum number of out-of-order TCP buffers for the specified TCP socket.
There are three types of buffers assigned to a TCP socket: receive, transmit and out of order. TCP packets may be received in sequence, or out of order. If out of order, they must be either stored for use later, or retransmitted by the sending host. The default value is 5 buffer of approximately 1500 bytes each. The maximum number is 43. A higher number of buffers can increase bulk data throughput, but will increase TCP latency if there is a problem with the connection, such as a reset.

Parameters

fd	Socket file descriptor
max	Maximum number of buffers

Returns

TCP Socket Status

See also

setsockoption(), getsockoption(), clrsockoption(), SetSocketRxBuffers(), SetSocketTxBuffers()

21.106.2.22 SetSocketRxBuffers()

```
int SetSocketRxBuffers (
            int fd,
            int n )
```

Set the number of TCP receive buffers for the specified TCP socket.
Sets the number of TCP buffers that can be accumulated on the receive socket receive buffer. Each buffer is 1548 bytes. The amount of free space in the receive buffer sets the TCP window size, controlling how much data the other side of the TCP connection is allowed to send.
When data is read from a socket, it frees up space in a buffer, allowing the other side of the TCP connection to send more data. Since the maximum window size is 65536, raising this value beyond 43 buffers has no benefit. The system default value is defined in constants.h with a value of 5 to conserve network buffers. Unless you are transferring large amounts of bulk data into your NetBurner device, the default provides more than enough space.

Parameters

fd	Socket file descriptor
n	Number of buffers

Returns

TCP Socket Status

See also

setsockoption(), getsockoption(), clrsockoption(), SetSocketUnackBuffers(), SetSocketTxBuffers()

21.106.2.23 SetSocketTxBuffers()

```
int SetSocketTxBuffers (
            int fd,
            int n )
```

Set the number of TCP transmit buffers for the specified TCP socket.
Sets the number of buffers that can be put in the transmit queue. Each buffer is 1548 bytes. The transmit queue accumulates data up to this limit regardless of the state of acknowledged buffers, or window size, from the other side of the TCP connection. This value sets the maximum size for a TCP write operation. Note: you must always check the return value of any write function to determine how many bytes were written.

Parameters

fd	Socket file descriptor
n	Number of buffers

Returns

TCP Socket Status

See also

setsockoption(), getsockoption(), clrsockoption(), SetSocketRxBuffers(), SetSocketUnackBuffers(), write(), writeall()

21.106.2.24 SetSocketUnackBuffers()

```
int SetSocketUnackBuffers (
            int fd,
            uint8_t val )  [inline]
```

Set the maximum number of outbound TCP buffers in the transmit un-acknowledged list for the specified TCP socket.
Sets the number of outbound TCP buffers allowed as un-acknowledged on a TCP socket. When transmitting, data moves from the TCP transmit queue to the list of un-acknowledged TCP packets once it is sent. When the other side of the TCP connection acknowledges the packet, it is removed from the un-acknowledged list. TCP can only send as many un-acknowledged buffers as the other side's TCP window size will allow.
Increasng this value will increase bulk data throughput on a fast network. Increasing the value could decreate performance on a lossy network without selective acknowledgement. Note: most PC's have selective acknowledgement enabled, many NAT routers do not.

Parameters

fd	Socket file descriptor
val	Number of buffers

Returns

TCP Socket Status

See also

setsockoption(), getsockoption(), clrsockoption(), SetSocketRxBuffers(), SetSocketTxBuffers()

21.106.2.25 setsockoption()

```
int setsockoption (
            int fd,
            int option )
```

Set TCP socket options.

Parameters

fd	Socket file descriptor.
option	Socket option to set: TCP Socket Options.

Returns

A bitmask of the options for the specified socket.

See also

clrsockoption(), getsockoption(), SetSocketUnackBuffers(), SetSocketRxBuffers(), SetSocketTxBuffers()

21.106.2.26 SocketPeek()

```
char SocketPeek (
            int fd )
```

Returns the next char that would be read, 0 if no data.

Parameters

fd	Socket file descriptor

Returns

The next char that would be read.

21.106.2.27 SockReadWithTimeout()

```
int SockReadWithTimeout (
            int fd,
            char * buf,
            int nbytes,
            uint32_t timeout )
```

Attempt to read from a TCP socket until the timeout expires.
The number of bytes parameter is the maximum to read. The function will return when at least 1 byte is available, so the return value must always be checked.

Note

Works with TCP sockets; does not work with TLS sockets.

Parameters

fd	Socket file descriptor
buf	Point to buffer to store received data
nbytes	Maximum number of bytes to read
timeout	Time out in ticks per second. Recommended use is the `TICKS_PER_SECOND` macro. For example: TICKS_PER_SECOND ∗ 5 for a 5 second delay.

Returns

Number of bytes read, or error TCP Socket Status

See also

ReadWithTimeout()

21.106.2.28 TcpAllDataAcked()

```
BOOL TcpAllDataAcked (
            int socket )
```

Check the data acknowledged state of a socket.

Parameters

socket	Socket file descriptor.

Returns

True if all the sent data has been acknowledged by the other side of TCP connection.

See also

WaitForSocketFlush

21.106.2.29 TcpGetLastRxInterval()

```
uint32_t TcpGetLastRxInterval (
            int fd )
```

Returns the number of system Time Ticks since the last packet was received. This is the difference between the current system time and lastRxTime of the socket.

Parameters

fd	The socket file descriptor.

Returns

The number of system Time Ticks since the last packet was received on the connection.

See also

TcpGetLastRxTime()

21.106.2.30 TcpGetLastRxTime()

```
uint32_t TcpGetLastRxTime (
            int fd )
```

Returns the value of system Time Ticks when the last packet was received. Used for the TCP Keep Alive feature. Each TCP connection has a corresponding Socket_struct. One of the elements of this struct is a variable called lastRxTime, which stores the current system time tick every time a packet is received. This function enables you to find out when the last packet was received by returning LastRxTime.
This function is used in conjunction with TcpSendKeepAlive() to implement the TCP keepalive feature.

- If lastRxTime is the same before and after a keepalive packet is sent, the client has not responded to the keepalive packet and it can be assumed that connection is lost.
- Make sure to allow enough time for the client to respond to the keep alive packet before checking the time.
- Do not call TcpGetLastRxTime() more often than once every second.

Parameters

fd	The socket file descriptor.

Returns

The value of system Time Ticks when the last packet was received on the connection.

See also

TcpSendKeepAlive()

21.106.2.31 TcpGetRxBufferSpaceUsed()

```
uint16_t TcpGetRxBufferSpaceUsed (
            int fd )
```

Returns the number of bytes used in a socket's RX buffer.

Parameters

fd	Socket file descriptor.

Returns

The number of bytes used

See also

TcpGetTxBufferAvailSpace
TcpGetTxDataWaiting

21.106.2.32 TcpGetSocketInterface()

```
int TcpGetSocketInterface (
            int fd )
```

Return the network interface associated with a TCP socket.

Parameters

fd	Socket file descriptor

Returns

The network interface number (undefined for listening sockets)

21.106.2.33 TcpGetSocketState()

```
uint8_t TcpGetSocketState (
            int fd )
```

Return the current state of a TCP socket.

Parameters

fd	Socket file descriptor.

Returns

The current state of the socket: TCP Socket State

21.106.2.34 TcpGetTxBufferAvailSpace()

```
uint16_t TcpGetTxBufferAvailSpace (
            int fd )
```

Returns the number of bytes available in a socket's TX buffer.

Parameters

fd	Socket file descriptor.

Returns

The number of bytes available

See also

TcpGetRxBufferSpaceUsed

21.106.2.35 TcpGetTxDataWaiting()

```
uint16_t TcpGetTxDataWaiting (
            int fd )
```

Returns the number of bytes waiting to be sent in a socket's TX Buffer.

Parameters

fd	Socket file descriptor.

Returns

The number of bytes used

See also

TcpGetTxBufferAvailSpace

TcpGetRxBufferSpaceUsed

21.106.2.36 TcpSendKeepAlive()

```
void TcpSendKeepAlive (
            int fd )
```

Send a TCP keep alive packet to a remote host.
Sends an empty packet with a decremented sequence number to a remote host. The sequence number will be one less than the last time a packet was sent. The remote host will respond with a TCP acknowledgment (ACK). This function is used in conjunction with TcpGetLastRxTime() to implement the TCP keepalive feature.

Parameters

fd	The socket file descriptor

See also

TcpGetLastRxTime()

21.106.2.37 WaitForSocketFlush()

```
BOOL WaitForSocketFlush (
            int fd,
            uint32_t ticks )
```

Wait for a socket flush operation to complete. A socket is flushed if all sent data has been acknowledged.

Parameters

fd	Socket file descriptor.
ticks	The maximum number of system time ticks to wait.

Return values

true	All the sent data has been acknowledged by the other side of TCP connection.
false	A timeout occurred.

See also

TcpAllDataAcked

21.107 TCP Notify

Typedefs

- typedef void() tcp_notify_handler(int tcpFd)

  TCP notification callback type.

Functions

- void RegisterTCPReadNotify (int tcpFd, tcp_notify_handler *notifyHandler)

  Register a TCP socket and callback function for read notifications.
- void RegisterTCPWriteNotify (int tcpFd, tcp_notify_handler *notifyHandler)

  Register a TCP socket and callback function for write notifications.

21.107.1 Detailed Description

Advanced function for notification of read or write activivty on a TCP socket. Callback functions can operate on a TCP socket as they would normally. This functionality provides a method of notification when read or write activities occur:

- A read notification calls the callback function when data is received.
- A write notification calls the callback function when the TCP write buffer can accept more data.

21.107.2 Typedef Documentation

21.107.2.1 tcp_notify_handler

```
typedef void() tcp_notify_handler(int tcpFd)
```

TCP notification callback type.
An application creates a callback function of this type to process TCP read and/or write notifications.

Parameters

tcpFd	TCP socket file descriptor of the socket that generated the notification

See also

RegisterTCPReadNotify(), RegisterTCPWriteNotify()

21.107.3 Function Documentation

21.107.3.1 RegisterTCPReadNotify()

```
void RegisterTCPReadNotify (
            int tcpFd,
            tcp_notify_handler * notifyHandler )
```

Register a TCP socket and callback function for read notifications.
The callback function will be called whenever data is read from the TCP socket

Parameters

tcpFd	Existing TCP socket file descriptor to monitor
notifyHandler	Pointer to callback function to process the notification

See also

RegisterTCPWriteNotify()

21.107.3.2 RegisterTCPWriteNotify()

```
void RegisterTCPWriteNotify (
            int tcpFd,
            tcp_notify_handler * notifyHandler )
```

Register a TCP socket and callback function for write notifications.
The callback function will be called whenever data can be written to the TCP socket buffer

Parameters

tcpFd	Existing TCP socket file descriptor to monitor
notifyHandler	Pointer to callback function to process the notification

See also

RegisterTCPReadNotify()

21.108 TCP Socket Options

Macros

- #define **SO_DEBUG** 1

  Reserved.
- #define **SO_NONAGLE** 2

  Disable the Nagle algorithm.
- #define **SO_NOPUSH** 4

  Disable TCP PUSH feature.

21.108.1 Detailed Description

Socket options for use with setsocketoptions().

21.109 TCP Socket State

Macros

- #define **TCP_STATE_CLOSED** (0)

  Socket Closed.
- #define **TCP_STATE_LISTEN** (1)

  Listen Socket.
- #define **TCP_STATE_SYN_SENT** (2)

  Packet with synchronization flag sent.
- #define **TCP_STATE_SYN_RCVD** (3)

  Packet with synchronisation flag received.
- #define **TCP_STATE_WAIT_FOR_ACCEPT** (4)

  Socket waiting for accept.
- #define **TCP_STATE_ESTABLISHED** (5)

  TCP session established.
- #define **TCP_STATE_CLOSE_WAIT** (6)

FIN received from client and socet is in the process of closing.

- #define **TCP_STATE_LAST_ACK** (7)

  Server is in the process of sending its own FIN signal.

- #define **TCP_STATE_FIN_WAIT_1** (8)

  Connection is active but not currently being used.

- #define **TCP_STATE_FIN_WAIT_2** (9)

  Client has received first ACK of the first FIN signal from server.

- #define **TCP_STATE_CLOSING** (10)

  Session is closing, but data may still be processed.

- #define **TCP_STATE_TIME_WAIT** (11)

  Client recognizes connection still open, but not currently being used.

21.109.1 Detailed Description

The current state of a TCP socket.

21.110 TCP Socket Status

Macros

- #define **TCP_ERR_NORMAL** (0)

  No errors.

- #define **TCP_ERR_TIMEOUT** (-1)

  Socket timed out.

- #define **TCP_ERR_NOCON** (-2)

  No connection exists.

- #define **TCP_ERR_CLOSING** (-3)

  Socket is in the process of closing.

- #define **TCP_ERR_NOSUCH_SOCKET** (-4)

  No such socket exists.

- #define **TCP_ERR_NONE_AVAIL** (-5)

  No new sockets are available.

- #define **TCP_ERR_CON_RESET** (-6)

  Connection has been reset.

- #define **TCP_ERR_CON_ABORT** (-7)

  Connection has been aborted.

21.110.1 Detailed Description

The current status of a TCP socket. When calling API functions that return a file descriptor, such as listen(), accept(), connect(), etc., a negative value represents an error, while a value greater than 0 represents the file descriptor number.

21.111 TFTP

Modules

- TFTP Return Codes

Functions

- int GetTFTP (PCSTR fileName, PCSTR mode, puint8_t buffer, int &len, uint32_t timeout, IPADDR4 server, uint16_t port=IANA_TFTP_PORT)

  Get a file from a TFTP server.
- int SendTFTP (PCSTR fileName, PCSTR mode, puint8_t buffer, int len, uint32_t timeout, uint32_t packet↩ Timeout, IPADDR4 server, uint16_t port=IANA_TFTP_PORT)

  Send a file to a TFTP server.

21.111.1 Detailed Description

The NetBurner TFTP Group

21.111.2 Function Documentation

21.111.2.1 GetTFTP()

```
int GetTFTP (
              PCSTR fileName,
              PCSTR mode,
              puint8_t buffer,
              int & len,
              uint32_t timeout,
              IPADDR4 server,
              uint16_t port = IANA_TFTP_PORT )
```

Get a file from a TFTP server.

Parameters

fileName	Name of file to retrieve
mode	Transfer mode. 'b' = binary, 't' = text
buffer	Data destination buffer
len	Maximum length when function is called, bytes read upon function return.
timeout	Total amount of time for file transfer, in system TimeTicks
server	IPv4 address of TFTP server
port	Optional TFTP server port number. If not specified the default TFTP port number 69 will be used.

Returns

TFTP Return Codes

See also

SendTFTP

21.111.2.2 SendTFTP()

```
int SendTFTP (
              PCSTR fileName,
              PCSTR mode,
              puint8_t buffer,
              int len,
              uint32_t timeout,
              uint32_t packetTimeout,
```

```
            IPADDR4 server,
            uint16_t port = IANA_TFTP_PORT )
```

Send a file to a TFTP server.

Parameters

fileName	Name of file to retrieve
mode	Transfer mode. 'b' = binary, 't' = text
buffer	Data destination buffer
len	Length of file to transfer, in bytes
timeout	Total amount of time for file transfer, in system TimeTicks
packetTimeout	Timeout for any single UDP packet to successfully transfer and be acknowledged, in system Time Ticks
server	IPv4 address of TFTP server
port	Optional TFTP server port number. If not specified the default TFTP port number 69 will be used.

Returns

TFTP Return Codes

See also

SendTFTP

21.112 TFTP Return Codes

Macros

- #define **TFTP_OK** (0)

  Success.
- #define **TFTP_TIMEOUT** (1)

  TFTP Timeout.
- #define **TFTP_ERROR** (2)

  TFTP Error.

21.112.1 Detailed Description

21.113 Time

Functions

- time_t set_time (time_t time_to_set, uint32_t fraction=0)

  Set the system time to the value passed in the time_t parameter.
- time_t timegm (struct tm ∗bts)

  Returns the value of type time_t that represents the GMT time described by the tm structure pointed by bts.
- void tzsetchar (const char ∗tzenv)

  Set the system local time.

21.113.1 Detailed Description

The NBTime library provides methods to set and read the system time. The system time will reset whenever power is lost or the system is reset. On power-up, system time can be set:

- With a NTP server (most common)

- From a real time clock (RTC)j
- Manually

Examples are provided in the example section for each of these methods.
Time is always stored as GMT. If you are using an internal or external real-time clock, it should also be set to GMT. Example application function calls are shown below. Refer to the examples for your specific NetBurner platform for RTC functions. Functions with an "_r" are reentrant.
List of common standard Time library functions include:

- time_t time() Returns current system GMT time.
- gmtime_r(time_t &timeT, tm &stmUTC) Converts given time since epoch (a time_t value pointed to by timeT) to calendar time, expressed in Coordinated Universal Time (UTC) in the struct tm format. The result is stored in the struct tm pointed to by stmUTC.
- localtime_r(time_t &tv, tm &tm_struct) Converts given time since epoch (time_t) to calendar time, expressed in local time, in the struct tm format. The result is stored in the tm struct.
- strftime() Creates a formatted time string and stores it in a buffer.
- double difftime(time_t t1, time_t t2) Returns the difference between two time_t values.

21.113.2 Function Documentation

21.113.2.1 set_time()

```
time_t set_time (
             time_t time_to_set,
             uint32_t fraction = 0 )
```

Set the system time to the value passed in the time_t parameter.

Parameters

time_to_set	time_t value to set the system time.
fraction	the fraction (in 1/(2^32) of the second)

Returns

The system time time_t value

21.113.2.2 timegm()

```
time_t timegm (
             struct tm * bts )
```

Returns the value of type time_t that represents the GMT time described by the tm structure pointed by bts.
Performs the reverse conversion gmtime(). This is not to be confused with the mktime() function, which performs the reverse conversion of the localtime() function.

Parameters

bts	Pointer to a tm structure

Returns

time_t value of tm

21.113.2.3 tzsetchar()

```
void tzsetchar (
            const char * tzenv )
```

Set the system local time.
This function initializes the TZ environment variable to contain the local time zone offset information from the standard time (UTC) and, if applicable, daylight saving time (DST). This does not overwrite the system time to the local time. The TZ environment variable is used by the ctime(), localtime(), mktime(), and strftime() functions of the standard time.h library to calculate the local time.
Once the TZ variable is set, the time library functions use it to determine when DST is in effect, when type conversions are made, or what time zone string is provided in the case of the strftime() function.
The system provides time zones around the world that can be referenced in the TZRecords variable in timezones.h. The time zone strings are located in timezones.cpp. Please refer to the timezones and RTC example programs for methods on utilizing these time zone settings.

Parameters

tzenv	The time zone string. For example: tzsetchar("EST5EDT4,M3.2.0/02:00:00,M11.1.0/02:00:00"); Eastern

21.114 Timers

Modules

- High Resolution Delay Timer
- Interval Timer
- Stopwatch Timer

21.114.1 Detailed Description

Hardware based high resolution timers:

- StopWatch to time events
- Interval Timer for periodic semaphore posts or interrupts
- HiResDelay (DelayObject) providing microsecond delay functions

21.115 UDP Error Codes

Macros

- #define **UDP_ERR_NOSUCH_SOCKET** (-1)

  Socket does not exist.
- #define **UDP_ERR_NOTOPEN_TO_WRITE** (-2)

  Socket not open for write.
- #define **UDP_ERR_NOTOPEN_TO_READ** (-3)

  Socket not open for read.

21.115.1 Detailed Description

21.116 UDP Packet Object

The NetBurner UDP protocol API is implemented in two ways:

Modules

- UDP Error Codes

Classes

- class UDPPacket

  UDP Packet Class.

Functions

- bool RegisterUDPFifo (uint16_t listenPort, OS_FIFO *pFifo)

  Register a FIFO to receive incoming UDP packets.
- bool RegisterUDPFifoVia (uint16_t listenPort, OS_FIFO *pFifo, int interface)

  Register a FIFO to receive incoming UDP packets on the specified network interface.
- uint16_t RegisterEphemeralFifo (OS_FIFO *pfifo, int ifn=-1)

  Register a UDP FIFO with a random port number to receive incoming UDP packets.
- bool RegisterUDPFifoWithNotify (uint16_t listenPort, OS_FIFO *pFifo, udp_data_notify *pNotifyFunction)

  Register a FIFO to receive incoming UDP packets and a callback function to receive a notification when a packet is received.
- bool RegisterUDPFifoWithNotifyVia (uint16_t listenPort, OS_FIFO *pFifo, udp_data_notify *pNotifyFunction, int interface)

  Register a FIFO to receive incoming UDP packets and a callback function to receive a notification when a packet is received. Same as RegisterUDPFifoWithNotify() with the additional parameter to specify the local network interface.
- void UnregisterUDPFifo (uint16_t listenPort, bool drain=false)

  Unregister a UDP FIFO.

21.116.1 Detailed Description

The NetBurner UDP protocol API is implemented in two ways:

1. As a C++ class, which makes memory management easier UDPPacket.
2. Wrapper functions that implement a standard UDP sockets interface. For example, sendto() and recvfrom().

21.116.2 Function Documentation

21.116.2.1 RegisterEphemeralFifo()

```
uint16_t RegisterEphemeralFifo (
            OS_FIFO * pfifo,
            int ifn = -1 )
```

Register a UDP FIFO with a random port number to receive incoming UDP packets.
A OS_FIFO is used to receive incoming UDP Packets. This function registers an OS_FIFO to listen on a random unused UDP port number, which can be useful when sending a UDP packet to a remote host that will respond to the received packet source port number.

Parameters

pfifo	Pointer to the OS_FIFO to register
ifn	Optional local interface number to listen on

Returns

The registered port number of success, 0 on failure.

See also

RegisterUDPFifoWithNotify(), UnregisterUDPFifo()

21.116.2.2 RegisterUDPFifo()

```
bool RegisterUDPFifo (
            uint16_t listenPort,
            OS_FIFO * pFifo )
```

Register a FIFO to receive incoming UDP packets.
A OS_FIFO is used to receive incoming UDP Packets. This function registers an OS_FIFO to listen to a specific UDP port number.

Parameters

listenPort	Port to listen on for incoming packets
pFifo	Pointer to the OS_FIFO to register

Returns

true if successful

See also

RegisterUDPFifoWithNotify(), UnregisterUDPFifo(), RegisterEphemeralFifo()

21.116.2.3 RegisterUDPFifoVia()

```
bool RegisterUDPFifoVia (
            uint16_t listenPort,
            OS_FIFO * pFifo,
            int interface )
```

Register a FIFO to receive incoming UDP packets on the specified network interface.
A OS_FIFO is used to receive incoming UDP Packets. This function registers an OS_FIFO to listen to a specific UDP port number and local network interface.

Parameters

listenPort	Port to listen on for incoming packets
pFifo	Pointer to the OS_FIFO to register
interface	Local network interface number

Returns

true if successful

See also

RegisterUDPFifo(), RegisterUDPFifoWithNotify(), UnregisterUDPFifo(), RegisterEphemeralFifo()

21.116.2.4 RegisterUDPFifoWithNotify()

```
bool RegisterUDPFifoWithNotify (
            uint16_t listenPort,
            OS_FIFO * pFifo,
            udp_data_notify * pNotifyFunction )
```

Register a FIFO to receive incoming UDP packets and a callback function to receive a notification when a packet is received.
A OS_FIFO is used to receive incoming UDP Packets. This function registers an OS_FIFO to listen to a specific UDP port number and a notification callback function.

Parameters

listenPort	Port to listen on for incoming packets
pFifo	Pointer to the OS_FIFO to register
pNotifyFunction	Pointer to callback function to receive notification

Returns

true if successful

See also

RegisterUDPFifo(), UnregisterUDPFifo()

21.116.2.5 RegisterUDPFifoWithNotifyVia()

```
bool RegisterUDPFifoWithNotifyVia (
            uint16_t listenPort,
            OS_FIFO * pFifo,
            udp_data_notify * pNotifyFunction,
            int interface )
```

Register a FIFO to receive incoming UDP packets and a callback function to receive a notification when a packet is received. Same as RegisterUDPFifoWithNotify() with the additional parameter to specify the local network interface.
A OS_FIFO is used to receive incoming UDP Packets. This function registers an OS_FIFO to listen to a specific UDP port number and a notification callback function.

Parameters

listenPort	Port to listen on for incoming packets
pFifo	Pointer to the OS_FIFO to register
pNotifyFunction	Pointer to callback function to receive notification
interface	Local network interface number

Returns

true if successful

See also

RegisterUDPFifo(), UnregisterUDPFifo()

21.116.2.6 UnregisterUDPFifo()

```
void UnregisterUDPFifo (
```

```
            uint16_t listenPort,
            bool drain = false )
```

Unregister a UDP FIFO.

Parameters

listenPort	Port to listen on for incoming packets.
drain	Optional, default if false. If set to true, will free all queued pool pointers.

See also

RegisterUDPFifo(), RegisterUDPFifoWithNotify()

21.117 UDP Socket

The NetBurner UDP protocol Socket API.

Functions

- int CreateRxUdpSocket (uint16_t listening_port)

  Open a UDP socket for receiving incoming UDP packets.
- int CreateTxUdpSocket (const IPADDR &send_to_addr, uint16_t remote_port, uint16_t local_port)

  Open a UDP socket for transmitting UDP packets.
- int CreateRxTxUdpSocket (const IPADDR &send_to_addr, uint16_t send_to_remote_port, uint16_t local_port)

  Open a UDP socket that can transmit and receive UDP packets.
- int sendto (int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t remote_port)

  Send a UDP packet.
- int sendtovia (int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t remote_port, int intfnum)

  Send a UDP packet on the specified interface.
- int recvfrom (int sock, puint8_t buffer, int len, IPADDR *pAddr, uint16_t *pLocal_port, uint16_t *pRemote_port)

  Receive a UDP packet.
- int SendFragmentedUdpPacket (const IPADDR &to, uint16_t source_port, uint16_t dest_port, puint8_t data, int length)

  Send a large fragmented UDP packet.
- int CreateRxUdpSocketVia (uint16_t listening_port, int interfacenum)

  Open a UDP socket for receiving incoming UDP packets on the specified network interface, by interface number.
- int CreateRxUdpSocketVia (uint16_t listening_port, const IPADDR interfaceIp)

  Open a UDP socket for receiving incoming UDP packets on the network interface with the specified IP address. If more than one interface has the same IP address, the lower interface number is used.
- int CreateRxTxUdpSocketVia (const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t local_port, int interfacenum)

  Open a UDP socket that can transmit and receive UDP packets on the local network interface specified by an interface number.
- int CreateRxTxUdpSocketVia (const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t local_port, IPADDR interfaceIp)

  Open a UDP socket that can transmit and receive UDP packets on the local network interface specified by an IP address.
- int CreateTxUdpSocketVia (const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port, int interfacenum)

  Open a UDP socket for transmitting UDP packets on the local network interface specified by the interface number.
- int CreateTxUdpSocketVia (const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port, IPADDR interfaceIp)

  Open a UDP socket for transmitting UDP packets on the local network interface specified by the interface IP address.

21.117.1 Detailed Description

The NetBurner UDP protocol Socket API.

1. As a C++ class, which makes memory management easier UDPPacket.
2. Wrapper functions that implement a standard UDP sockets interface.For example, sendto() and recvfrom().

21.117.2 Function Documentation

21.117.2.1 CreateRxTxUdpSocket()

```
int CreateRxTxUdpSocket (
            const IPADDR & send_to_addr,
            uint16_t send_to_remote_port,
            uint16_t local_port )  [inline]
```

Open a UDP socket that can transmit and receive UDP packets.

Parameters

send_to_addr	Destination IP address
send_to_remote_port	Destination port number
local_port	Local port number. A value of 0 will select a random port number (recommended)

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocket(), CreateTxUdpSocket()

21.117.2.2 CreateRxTxUdpSocketVia() [1/2]

```
int CreateRxTxUdpSocketVia (
            const IPADDR send_to_addr,
            uint16_t send_to_remote_port,
            uint16_t local_port,
            int interfacenum )  [inline]
```

Open a UDP socket that can transmit and receive UDP packets on the local network interface specified by an interface number.

Parameters

send_to_addr	Destination IP address
send_to_remote_port	Destination port number
local_port	Local port number. A value of 0 will select a random port number (recommended)
interfacenum	Local network interface number

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocketVia(), CreateTxUdpSocketVia()

21.117.2.3 CreateRxTxUdpSocketVia() [2/2]

```
int CreateRxTxUdpSocketVia (
            const IPADDR send_to_addr,
            uint16_t send_to_remote_port,
            uint16_t local_port,
            IPADDR interfaceIp )  [inline]
```

Open a UDP socket that can transmit and receive UDP packets on the local network interface specified by an IP address.

Parameters

send_to_addr	Destination IP address
send_to_remote_port	Destination port number
local_port	Local port number. A value of 0 will select a random port number (recommended)
interfaceIp	IP address of local network interface to use

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocketVia(), CreateTxUdpSocketVia()

21.117.2.4 CreateRxUdpSocket()

```
int CreateRxUdpSocket (
            uint16_t listening_port )
```

Open a UDP socket for receiving incoming UDP packets.
Returns a file descriptor that can be used by read functions such as: read() and select() to process multiple file descriptors at one time.

Parameters

listening_port	Port to listen on for incoming packets

Returns

The file descriptor for the UDP socket

See also

CreateTxUdpSocket(), CreateRxTxUdpSocket()

21.117.2.5 CreateRxUdpSocketVia() [1/2]

```
int CreateRxUdpSocketVia (
            uint16_t listening_port,
            const IPADDR interfaceIp )
```

Open a UDP socket for receiving incoming UDP packets on the network interface with the specified IP address. If more than one interface has the same IP address, the lower interface number is used.
Returns a file descriptor that can be used by read functions such as read() and select().

Parameters

listening_port	Port to listen on for incoming packets
interfaceIp	IP address of the network interface

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocketVia(), CreateTxUdpSocketVia()

21.117.2.6 CreateRxUdpSocketVia() [2/2]

```
int CreateRxUdpSocketVia (
            uint16_t listening_port,
            int interfacenum )
```

Open a UDP socket for receiving incoming UDP packets on the specified network interface, by interface number.
Returns a file descriptor that can be used by read functions such as read() and select().

Parameters

listening_port	Port to listen on for incoming packets
interfacenum	Network interface number

Returns

The file descriptor for the UDP socket

See also

CreateTxUdpSocketVia(), CreateRxTxUdpSocketVia()

21.117.2.7 CreateTxUdpSocket()

```
int CreateTxUdpSocket (
            const IPADDR & send_to_addr,
            uint16_t remote_port,
            uint16_t local_port )  [inline]
```

Open a UDP socket for transmitting UDP packets.
Returns a file descriptor that can be used by transmit functions such as sendto().

Parameters

send_to_addr	Destination IP address
remote_port	Destination port number
local_port	Local port number. A value of 0 will select a random port number (recommended)

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocket(), CreateRxTxUdpSocket()

21.117.2.8 CreateTxUdpSocketVia() [1/2]

```
int CreateTxUdpSocketVia (
            const IPADDR send_to_addr,
            uint16_t remote_port,
            uint16_t local_port,
            int interfacenum )  [inline]
```

Open a UDP socket for transmitting UDP packets on the local network interface specified by the interface number. Returns a file descriptor that can be used by transmit functions such as sendto().

Parameters

send_to_addr	Destination IP address
remote_port	Destination port number
local_port	Local port number. A value of 0 will select a random port number (recommended)
interfacenum	Local network interface number

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocketVia(), CreateRxTxUdpSocketVia()

21.117.2.9 CreateTxUdpSocketVia() [2/2]

```
int CreateTxUdpSocketVia (
            const IPADDR send_to_addr,
            uint16_t remote_port,
            uint16_t local_port,
            IPADDR interfaceIp )  [inline]
```

Open a UDP socket for transmitting UDP packets on the local network interface specified by the interface IP address. Returns a file descriptor that can be used by transmit functions such as sendto().

Parameters

send_to_addr	Destination IP address
remote_port	Destination port number
local_port	Local port number. A value of 0 will select a random port number (recommended)
interfaceIp	IP address of the local network interface

Returns

The file descriptor for the UDP socket

See also

CreateRxUdpSocketVia(), CreateRxTxUdpSocketVia()

21.117.2.10 recvfrom()

```
int recvfrom (
             int sock,
             puint8_t buffer,
             int len,
             IPADDR * pAddr,
             uint16_t * pLocal_port,
             uint16_t * pRemote_port )  [inline]
```

Receive a UDP packet.

Parameters

sock	File descriptor from a previous call to CreateRxUdpSocket() or CreateRxTxSocket()
buffer	Pointer to BYTE array buffer to store received UDP packet data
len	Maximum number of bytes to receive
pAddr	Pointer to an IPADDR variable to store the sender's IP address of the packet
pLocal_port	Pointer to WORD variable to store the local port number of the packet
pRemote_port	Pointer to WORD variable to store the sender's port number of the packet

Returns

The number of bytes received, or a negative number if an error occurred UDP Error Codes

See also

sendto(), sendtovia()

21.117.2.11 SendFragmentedUdpPacket()

```
int SendFragmentedUdpPacket (
             const IPADDR & to,
             uint16_t source_port,
             uint16_t dest_port,
             puint8_t data,
             int length )  [inline]
```

Send a large fragmented UDP packet.

Parameters

to	Destination IP address
source_port	Source port number, a value of 0 selects a random source port number (recommended)
dest_port	Destination port number
data	Pointer to data to send
length	Number of bytes to send

Returns

The number of bytes sent, or 0 if an error occurred.

21.117.2.12 sendto()

```
int sendto (
            int sock,
            puint8_t what_to_send,
            int len_to_send,
            const IPADDR & to_addr,
            uint16_t remote_port )  [inline]
```

Send a UDP packet.

Parameters

sock	Socket to send packet to
what_to_send	Pointer to data to be sent
len_to_send	Number of data bytes to send
to_addr	Destination IP address
remote_port	Destination port number

Returns

The number of bytes sent, or a negative number if an error occurred UDP Error Codes

See also

sendtovia(), recvfrom()

21.117.2.13 sendtovia()

```
int sendtovia (
            int sock,
            puint8_t what_to_send,
            int len_to_send,
            const IPADDR & to_addr,
            uint16_t remote_port,
            int intfnum )  [inline]
```

Send a UDP packet on the specified interface.

Parameters

sock	Socket to send packet to
what_to_send	Pointer to data to be sent
len_to_send	Number of data bytes to send
to_addr	Destination IP address
remote_port	Destination port number
intfnum	Interface number

Returns

The number of bytes sent, or a negative number if an error occurred UDP Error Codes

See also

sendto(), recvfrom()

21.118 UsartModuleNumber

USART Peripheral Module.

Macros

- #define **DEFAULT_USART_SPI_MODULE** 0

  Default QUADSPI module.
- #define **USART_SPI_MODULE_COUNT** 2

  Number of modules: 0, 1.

21.118.1 Detailed Description

USART Peripheral Module.

21.119 User Authorization Manager

Modules

- Authorization Responses
- Authorization Types

Classes

- struct UserAuthRecord

  A stored record of a user's authorization credentials. The value is hashed when saved so it can't be read directly. User's can currently only have one authorization record per user name.
- class UserAuthManager

  The user authorization manager class allows application developers the ability to manage user authorization records. The can be loaded and saved to any storage space, including the config system or UserParams. Authorization values are hashed before being saved. Validation compares both the hash as well as the authorization type. Adding, updating, and removing records will automatically call the user devined save functions. For usage, please see the example found in `examples/SSH/sshServerUserAuth`.

Typedefs

- typedef int(∗ SaveAuthRecordsFn) (const UserAuthRecord ∗authRec)

  User provided function for saving user authorization records. This allows the users to dictate where their authorization records are stored.
- typedef int(∗ LoadAuthRecordsFn) (UserAuthRecord ∗authRec)

  User provided function for loading user authorization records. This allows the users to dictate where their authorization records are stored.

21.119.1 Detailed Description

The NetBurner User Authorization Manager

21.119.2 Typedef Documentation

21.119.2.1 LoadAuthRecordsFn

```
typedef int(* LoadAuthRecordsFn) (UserAuthRecord *authRec)
```

User provided function for loading user authorization records. This allows the users to dictate where their authorization records are stored.

Parameters

authRec	A pointer to an array of UserAuthRecords where the data will be loaded.

Return values

0	If the authorization records were loaded correctly.
!0	If there was an error loading the authorization records. Error codes are left to application developers to define.

21.119.2.2 SaveAuthRecordsFn

```
typedef int(* SaveAuthRecordsFn) (const UserAuthRecord *authRec)
```

User provided function for saving user authorization records. This allows the users to dictate where their authorization records are stored.

Parameters

authRec	A pointer to an array of UserAuthRecords that should be saved.

Return values

0	If the authorization records were saved correctly.
!0	If there was an error saving the authorization records. Error codes are left to application developers to define.

21.120 Web Client

Modules

- Web Client Error Codes

Classes

- class ParsedURI

  Parsed Uniform Resource Identifier Class (URI)

Functions

- void SetHttpDiag (bool b)

  Enable/disable Web Client HTTP diagnostics to the console port.
- int DoMultipartStartPost (ParsedURI &TheUri, const char *separator, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND, uint32_t contentLength=0)

  Start a multipart HTTP post using a pre-parsed URI object.
- int DoMultipartStartPost (const char *pUrl, const char *separator, uint16_t TIMEOUT_WAIT=10 *TICKS_↩ PER_SECOND, uint32_t contentLength=0)

  Start a multipart HTTP post using a pointer to a URL.

- void DoMultipartItem (int tcpfd, const char *Disposition, const char *separator, const unsigned char *data, int len)

  Send a multipart item.
- void DoMultipartBoundary (int tcpfd, const char *Disposition, const char *separator)

  Send a multipart boundary.
- bool DoMultipartFinished (int tcpfd, const char *separator, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Send a multipart item.
- bool DoUrlEncodedFormPost (ParsedURI &TheUri, char *headers, char *form_data, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT)

  Post a JSON file using a HTTP Form Post and a a ParsedURI object.
- bool DoUrlEncodedFormPost (const char *pUrl, char *headers, char *form_data, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT)

  Post a JSON file using a HTTP POST and a URL string and pointer to JSON data.
- bool DoJsonPost (const char *pUrl, const char *Json_Data_To_Post, buffer_object &result_buffer, const char *AdditionalHeaders=NULL, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using a HTTP POST and a URL string and pointer to JSON data.
- bool DoJsonPost (ParsedURI &TheUri, const char *Json_Data_To_Post, buffer_object &result_buffer, const char *AdditionalHeaders=NULL, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using a HTTP POST and a ParsedURI object.
- bool DoJsonPost (const char *pUrl, ParsedJsonDataSet &jsonout, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using a HTTP POST and a URL string.
- bool DoJsonPost (ParsedURI &TheUri, ParsedJsonDataSet &jsonout, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using HTTP and a ParsedURI object.
- bool DoJsonPostHttpFile (const char *pUrl, const char *FragmentName, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using HTTP and a URL string.
- bool DoJsonPostHttpFile (ParsedURI &TheUri, const char *FragmentName, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using HTTP and a ParsedURI object.
- bool DoGet (ParsedURI &TheUri, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)
- bool DoGetEx (ParsedURI &TheUri, const char *headers, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)
- bool DoGet (const char *pUrl, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- bool DoGetEx (const char *pUrl, const char *headers, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- int DoGet (ParsedURI &TheUri, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a reference to a parsed Uniform Resource Identifier (URI)
- int DoGetEx (ParsedURI &TheUri, const char *headers, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a reference to a parsed Uniform Resource Identifier (URI)
- int DoGet (const char *pUrl, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- int DoGetEx (const char *pUrl, const char *headers, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

Execute a HTTP/HTTPS GET request using a pointer to a URL string.

- int DoGetUpdate (ParsedURI &TheUri, uint16_t TIMEOUT_WAIT=10 ∗TICKS_PER_SECOND)

  Execute a firmware update from the specified URI.
- int DoGetUpdate (const char ∗pUrl, uint16_t TIMEOUT_WAIT=10 ∗TICKS_PER_SECOND)

  Execute a firmware update from the specified URI.
- int PopulateAuthHeader (const char ∗user, const char ∗password, char ∗buffer, int maxlen)

  Fill in a username and password into a buffer for use as an extra header.
- bool StartWebClient (int prio, const char ∗url1, const char ∗url2=NULL, bool bDoNtp=false)

  Start the web client using a URL string.
- bool StartWebClient (int prio, const NBString &url1, const NBString &url2, bool bDoNtp=false)

  Start the web client using a NBString.
- bool StartWebClient (int prio, const NBString &url1, bool bDoNtp=false)

  Start the web client using a NBString.

21.120.1 Detailed Description

The Web Client Group

21.120.2 Function Documentation

21.120.2.1 DoGet() [1/4]

```
bool DoGet (
            const char * pUrl,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a HTTP/HTTPS GET request using a pointer to a URL string.
This function will parse the URL string to connect to the host.

Parameters

pUrl	Pointer to a URL string. A DNS lookup will be done if necessary.
&result_buffer	Reference to the memory buffer in which to store the result.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.2 DoGet() [2/4]

```
int DoGet (
            const char * pUrl,
            unsigned char * result,
            int maxl,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a HTTP/HTTPS GET request using a pointer to a URL string.
This function will parse the URL string to connect to the host.

Parameters

pUrl	Pointer to a URL string. A DNS lookup will be done if necessary.

Parameters

result	Pointer to the buffer in which to store the result.
maxl	Maximum length of the result buffer.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

21.120.2.3 DoGet() [3/4]

```
bool DoGet (
            ParsedURI & TheUri,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Parameters

&TheUri	Reference to the pre-parsed URI.
&result_buffer	Reference to the location to store the result.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.4 DoGet() [4/4]

```
int DoGet (
            ParsedURI & TheUri,
            unsigned char * result,
            int maxl,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a HTTP/HTTPS GET request using a reference to a parsed Uniform Resource Identifier (URI)
This function uses a reference to a parsed URI, providing faster execution and avoiding parsing a URL, DNS, etc.

Parameters

TheUri	Reference to the ParsedURI object.
result	Pointer to the buffer in which to store the result.
maxl	Maximum length of the result buffer.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

21.120.2.5 DoGetEx() [1/4]

```
bool DoGetEx (
            const char * pUrl,
```

```
            const char * headers,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a HTTP/HTTPS GET request using a pointer to a URL string.
This function will parse the URL string to connect to the host.

Parameters

pUrl	Pointer to a URL string. A DNS lookup will be done if necessary.
headers	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
&result_buffer	Reference to the memory buffer in which to store the result.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.6 DoGetEx() [2/4]

```
int DoGetEx (
            const char * pUrl,
            const char * headers,
            unsigned char * result,
            int maxl,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a HTTP/HTTPS GET request using a pointer to a URL string.
This function will parse the URL string to connect to the host.

Parameters

pUrl	Pointer to a URL string. A DNS lookup will be done if necessary.
headers	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
result	Pointer to the buffer in which to store the result.
maxl	Maximum length of the result buffer.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

21.120.2.7 DoGetEx() [3/4]

```
bool DoGetEx (
            ParsedURI & TheUri,
            const char * headers,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Parameters

&TheUri	Reference to the pre-parsed URI.

Parameters

headers	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
&result_buffer	Reference to the location to store the result.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.8 DoGetEx() [4/4]

```
int DoGetEx (
            ParsedURI & TheUri,
            const char * headers,
            unsigned char * result,
            int maxl,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a HTTP/HTTPS GET request using a reference to a parsed Uniform Resource Identifier (URI)
This function uses a reference to a parsed URI, providing faster execution and avoiding parsing a URL, DNS, etc.

Parameters

TheUri	Reference to the ParsedURI object.
headers	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
result	Pointer to the buffer in which to store the result.
maxl	Maximum length of the result buffer.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

21.120.2.9 DoGetUpdate() [1/2]

```
int DoGetUpdate (
            const char * pUrl,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a firmware update from the specified URI.

Parameters

pUrl	Pointer to a URL string. A DNS lookup will be done if necessary.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

21.120.2.10 DoGetUpdate() [2/2]

```
int DoGetUpdate (
            ParsedURI & TheUri,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Execute a firmware update from the specified URI.

Parameters

TheUri	Reference to the ParsedURI object.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

21.120.2.11 DoJsonPost() [1/4]

```
bool DoJsonPost (
            const char * pUrl,
            const char * Json_Data_To_Post,
            buffer_object & result_buffer,
            const char * AdditionalHeaders = NULL,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Post a JSON file using a HTTP POST and a URL string and pointer to JSON data.

Parameters

pUrl	Pointer to a URL string.
Json_Data_To_Post	Pointer to serialized JSON data
&result_buffer	Reference to the location in which to store the result.
AdditionalHeaders	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.12 DoJsonPost() [2/4]

```
bool DoJsonPost (
            const char * pUrl,
            ParsedJsonDataSet & jsonout,
            buffer_object & result_buffer,
            const char * AdditionalHeaders,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Post a JSON file using a HTTP POST and a URL string.

Parameters

pUrl	Pointer to a URL string.
jsonout	Reference to NetBurner data object. Please ref JSON lexer.
&result_buffer	Reference to the location in which to store the result.

Parameters

AdditionalHeaders	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.13 DoJsonPost() [3/4]

```
bool DoJsonPost (
            ParsedURI & TheUri,
            const char * Json_Data_To_Post,
            buffer_object & result_buffer,
            const char * AdditionalHeaders = NULL,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Post a JSON file using a HTTP POST and a ParsedURI object.

Parameters

&TheUri	Reference to the pre-parsed URI.
Json_Data_To_Post	Serialized JSON data\
&result_buffer	Reference to the location in which to store the result.
AdditionalHeaders	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.14 DoJsonPost() [4/4]

```
bool DoJsonPost (
            ParsedURI & TheUri,
            ParsedJsonDataSet & jsonout,
            buffer_object & result_buffer,
            const char * AdditionalHeaders,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Post a JSON file using HTTP and a ParsedURI object.

Parameters

&TheUri	Reference to the pre-parsed URI.
jsonout	Reference to NetBurner data object. Please ref JSON lexer.
&result_buffer	Reference to the location in which to store the result.
AdditionalHeaders	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.15 DoJsonPostHttpFile() [1/2]

```
bool DoJsonPostHttpFile (
            const char * pUrl,
            const char * FragmentName,
            buffer_object & result_buffer,
            const char * AdditionalHeaders,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Post a JSON file using HTTP and a URL string.

Parameters

pUrl	Pointer to a URL string.
FragmentName	
&result_buffer	Reference to the location in which to store the result.
AdditionalHeaders	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.16 DoJsonPostHttpFile() [2/2]

```
bool DoJsonPostHttpFile (
            ParsedURI & TheUri,
            const char * FragmentName,
            buffer_object & result_buffer,
            const char * AdditionalHeaders,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Post a JSON file using HTTP and a ParsedURI object.

Parameters

&TheUri	Reference to the pre-parsed URI.
FragmentName	
&result_buffer	Reference to the location in which to store the result.
AdditionalHeaders	
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.17 DoMultipartBoundary()

```
void DoMultipartBoundary (
            int tcpfd,
```

```
            const char * Disposition,
            const char * separator )
```

Send a multipart boundary.

Parameters

tcpfd	TCP file descriptor returned from DoMultipartStartPost()
Disposition	Content disposition field of multipart body
separator	Separator string to use as the part delimiter between multipart posts.

21.120.2.18 DoMultipartFinished()

```
bool DoMultipartFinished (
            int tcpfd,
            const char * separator,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND )
```

Send a multipart item.

Parameters

tcpfd	TCP file descriptor returned from DoMultipartStartPost()
separator	Separator string to use as the part delimiter between multipart posts.
result_buffer	Reference to buffer to store result of xxxxx
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

****** Need to understand in http_funcs.cpp

21.120.2.19 DoMultipartItem()

```
void DoMultipartItem (
            int tcpfd,
            const char * Disposition,
            const char * separator,
            const unsigned char * data,
            int len )
```

Send a multipart item.

Parameters

tcpfd	TCP file descriptor returned from DoMultipartStartPost()
Disposition	Content disposition field of multipart body
separator	Separator string to use as the part delimiter between multipart posts.
data	Item data
len	Data length

21.120.2.20 DoMultipartStartPost() [1/2]

```
int DoMultipartStartPost (
            const char * pUrl,
            const char * separator,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND,
            uint32_t contentLength = 0 )
```

Start a multipart HTTP post using a pointer to a URL.

Parameters

pUrl	Pointer to a URL string.
separator	Separator string to use as the part delimiter between multipart posts.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.
contentLength	If given, content length for the entire multipart post.

Returns

A TCP/TLS file descriptor greater than 0 if a socket was successfully opened, or TCP Socket Status on failure.

21.120.2.21 DoMultipartStartPost() [2/2]

```
int DoMultipartStartPost (
            ParsedURI & TheUri,
            const char * separator,
            uint16_t TIMEOUT_WAIT = 10 *TICKS_PER_SECOND,
            uint32_t contentLength = 0 )
```

Start a multipart HTTP post using a pre-parsed URI object.

Parameters

&TheUri	Reference to the pre-parsed URI object.
separator	Separator string to use as the part delimiter between multipart posts.
TIMEOUT_WAIT	Timeout in seconds to wait for a response.
contentLength	If given, content length for the entire multipart post.

Returns

A TCP/TLS file descriptor greater than 0 if a socket was successfully opened, or TCP Socket Status on failure.

21.120.2.22 DoUrlEncodedFormPost() [1/2]

```
bool DoUrlEncodedFormPost (
            const char * pUrl,
            char * headers,
            char * form_data,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT )
```

Post a JSON file using a HTTP POST and a URL string and pointer to JSON data.

Parameters

pUrl	Pointer to a URL string.

Parameters

headers	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
form_data	Pointer to the JSON data you are posting.
&result_buffer	Reference to the location in which to store the result.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.23 DoUrlEncodedFormPost() [2/2]

```
bool DoUrlEncodedFormPost (
            ParsedURI & TheUri,
            char * headers,
            char * form_data,
            buffer_object & result_buffer,
            uint16_t TIMEOUT_WAIT )
```

Post a JSON file using a HTTP Form Post and a a ParsedURI object.

Parameters

&TheUri	Reference to the pre-parsed URI.
headers	Additional HTTP header fields. Multiple header fields should be concatenated together and separated by by \r\n Do not put \r\n for last header (no trailing \r\n)
form_data	Pointer to the JSON data you are posting.
&result_buffer	Reference to the location in which to store the result.
TIMEOUT_WAIT	Timeout in in system time ticks to wait for a response.

Returns

True on success, false on failure.

21.120.2.24 PopulateAuthHeader()

```
int PopulateAuthHeader (
            const char * user,
            const char * password,
            char * buffer,
            int maxlen )
```

Fill in a username and password into a buffer for use as an extra header.

Parameters

user	Pointer to username
password	Pointer to password
buffer	Destination buffer to put the authenticate header in.
maxlen	Maximum number of chars to put in buffer

Returns

number of chars written, negative if the buffer was too small.

21.120.2.25 SetHttpDiag()

```
void SetHttpDiag (
            bool b )
```

Enable/disable Web Client HTTP diagnostics to the console port.

Parameters

b	Set true to enable, false to disable.

21.120.2.26 StartWebClient() [1/3]

```
bool StartWebClient (
            int prio,
            const char * url1,
            const char * url2 = NULL,
            bool bDoNtp = false )
```

Start the web client using a URL string.

Parameters

prio	Priority
url1	Pointer to a string representing the URL.
url2	Pointer to a second URL. Optional parameter, default is NULL.
bDoNtp	Enable the Network Time Protocol (NTP). Optional parameter, default is false.

Returns

True on success, false on failure.

21.120.2.27 StartWebClient() [2/3]

```
bool StartWebClient (
            int prio,
            const NBString & url1,
            bool bDoNtp = false )
```

Start the web client using a NBString.

Parameters

prio	Priority
url1	Reference to a NBString representing the URL.
bDoNtp	Enable the Network Time Protocol (NTP). Optional parameter, default is false.

Returns

True on success, false on failure.

21.120.2.28 StartWebClient() [3/3]

```
bool StartWebClient (
            int prio,
            const NBString & url1,
            const NBString & url2,
            bool bDoNtp = false )
```

Start the web client using a NBString.

Parameters

prio	Priority
url1	Reference to a string representing the URL.
url2	Reference to a second URL. Optional parameter, default is NULL.
bDoNtp	Enable the Network Time Protocol (NTP). Optional parameter, default is false.

Returns

True on success, false on failure.

21.121 Web Client Error Codes

Macros

- #define **WEB_CLIENT_ERROR_NO_ETHERNET** (1)

  No Ethernet connection.
- #define **WEB_CLIENT_ERROR_NO_ADDRESS** (2)

  Interface has no IP address.
- #define **WEB_CLIENT_ERROR_NO_GATEWAY** (3)

  Interface has no gateway IP address.
- #define **WEB_CLIENT_ERROR_GATEWAY_WRONG** (4)

  Interface has incorrect gateway IP address.
- #define **WEB_CLIENT_ERROR_NO_DNS_ADDR** (5)

  Interface has a DNS errror.
- #define **WEB_CLIENT_ERROR_NO_DNS_RESOLVE** (6)

  DNS was not able to resolve the URL name.
- #define **WEB_CLIENT_ERROR_NO_NTP** (7)

  NTP failed.
- #define **WEB_CLIENT_ERROR_NO_SERVER_RESPONSE** (8)

  No response from server.
- #define **WEB_CLIENT_ERROR_NO_SERVER_CONNECT** (9)

  Unable to connect to server.
- #define **WEB_CLIENT_ERROR_NO_ERROR** (10)

  No error.

21.121.1 Detailed Description

Web Client error codes

21.122 Wifi

Modules

- BSS Options

  Option list BSS type values.
- Cipher Options
- Configuration Errors
- Connect Request Errors
- Save Config Record Errors
- Scan Request Errors
- Security Options

Classes

- struct wifi_init

Functions

- nbWifiScanResult * WifiInitScan_SPI (int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver over the SPI bus, and performs an AP scan.
- int WifiInitScanAndShow_SPI (int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver using the SPI bus, performs an AP scan, and prints the scan results via serial output.
- int InitWifi_SPI (const char *SSID="", const char *password="", int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver using the SPI bus, and attempts to establish the specified connection.
- int InitAP_SPI (const char *SSID="", const char *password="", uint8_t channel=NBWIFI_DEFAULT_WIFICHANNEL, int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver using the SPI bus, and attempts to establish the specified access point.
- nbWifiScanResult * WifiInitScan_Serial (int portNum=-1, int resetPinNum=-1, int connectorNum=-1)

  Initializes the WiFi hardware, initializes the driver using the UART interface, performs an AP scan, and prints the scan results via serial output.
- int WifiInitScanAndShow_Serial (int portNum=-1, int resetPinNum=-1, int connectorNum=-1)

  Initializes the WiFi hardware, initializes the driver using the UART interface, performs an AP scan, and prints the scan results via serial output.
- int InitWifi_Serial (const char *SSID="", const char *password="", int portNum=-1, int resetPinNum=-1, int connectorNum=-1)

  Initializes the WiFi hardware, initializes the driver using the UART interface, and attempts to establish the specified access point.
- void SetWifiSPISpeed (int busSpeed)

  Set SPI bus speed.
- void ScanAndShowNetworks ()

  Scan for surrounding access points and print the results via iprintf.
- nbWifiScanResult * ScanForNetworks ()

  Scan for surrounding access points.

21.122.1 Detailed Description

21.122.2 Function Documentation

21.122.2.1 InitAP_SPI()

```
int InitAP_SPI (
            const char * SSID = "",
            const char * password = "",
            uint8_t channel = NBWIFI_DEFAULT_WIFICHANNEL,
            int irqNum = -1,
            int moduleNum = -1,
            int csNum = -1,
            int connectorNum = -1,
            int gpioPinNum = -1,
            int resetPinNum = -1 )
```

Initializes the WiFi hardware, initializes the driver using the SPI bus, and attempts to establish the specified access point.

Parameters

in	*SSID*	Service Set Identity (SSID), nullptr uses SSID stored in the configuration record.
in	*password*	key shared with access point/peer; nullptr uses the key stored in the configuration record.
in	*channel*	802.11 channel to establish an access point on.
in	*irqNum*	IRQ signal used to communicate with the WiFi module.
in	*moduleNum*	SPI module number used to communicate with the WiFi module.
in	*csNum*	Chip Select signal used for SPI communication.
in	*connectorNum*	physical header number of the NetBurner module used.
in	*gpioPinNum*	GPIO pin number to use for the SPI chip select if not using the SPI module's native chip select signal.
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.

Returns

returns the Wifi interface number, if successful. Otherwise, returns NB::Error::GeneralErrors or NB::Error::Init::InitializationErrors

21.122.2.2 InitWifi_Serial()

```
int InitWifi_Serial (
            const char * SSID = "",
            const char * password = "",
            int portNum = -1,
            int resetPinNum = -1,
            int connectorNum = -1 )
```

Initializes the WiFi hardware, initializes the driver using the UART interface, and attempts to establish the specified access point.
NOTE: Serial functions pertaining to WiFi are only available for the NBWIFIIN.

Parameters

in	*SSID*	Service Set Identity (SSID), nullptr uses SSID stored in the configuration record.
in	*password*	key shared with access point/peer; nullptr uses the key stored in the configuration record.
in	*portNum*	UART number
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.
in	*connectorNum*	physical header number of the NetBurner module used.

Returns

returns zero if successful, otherwise returns NB::Error::GeneralErrors or NB::Error::Init::InitializationErrors

21.122.2.3 InitWifi_SPI()

```
int InitWifi_SPI (
            const char * SSID = "",
            const char * password = "",
            int irqNum = -1,
            int moduleNum = -1,
            int csNum = -1,
            int connectorNum = -1,
            int gpioPinNum = -1,
            int resetPinNum = -1 )
```

Initializes the WiFi hardware, initializes the driver using the SPI bus, and attempts to establish the specified connection.

Parameters

in	*SSID*	Service Set Identity (SSID), nullptr uses SSID stored in the configuration record.
in	*password*	key shared with access point/peer; nullptr uses the key stored in the configuration record.
in	*irqNum*	IRQ signal used to communicate with the WiFi module.
in	*moduleNum*	SPI module number used to communicate with the WiFi module.
in	*csNum*	Chip Select signal used for SPI communication.
in	*connectorNum*	physical header number of the NetBurner module used.
in	*gpioPinNum*	GPIO pin number to use for the SPI chip select, if not using the SPI's native chip select signal.
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.

Returns

returns the Wifi interface number, if successful. Otherwise, returns NB::Error::GeneralErrors or NB::Error::Init::InitializationErrors

21.122.2.4 ScanAndShowNetworks()

```
void ScanAndShowNetworks ( )
```

Scan for surrounding access points and print the results via iprintf.
This function is bus interface agnostic.

21.122.2.5 ScanForNetworks()

```
nbWifiScanResult * ScanForNetworks ( )
```

Scan for surrounding access points.

Returns

nbWifiScanResult Head of linked list containing the AP scan results. nullptr if scan failed or no access points were found

This function is bus interface agnostic.

21.122.2.6 SetWifiSPISpeed()

```
void SetWifiSPISpeed (
            int busSpeed )
```

Set SPI bus speed.

Parameters

in	*busSpeed*	Bus speed in Hz.

21.122.2.7 WifiInitScan_Serial()

```
nbWifiScanResult * WifiInitScan_Serial (
            int portNum = -1,
            int resetPinNum = -1,
            int connectorNum = -1 )
```

Initializes the WiFi hardware, initializes the driver using the UART interface, performs an AP scan, and prints the scan results via serial output.
NOTE: Serial functions pertaining to WiFi are only available for the NBWIFIIN.

Parameters

in	*portNum*	UART number
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.
in	*connectorNum*	physical header number of the NetBurner module used.

Returns

nbWifiScanResult Head of linked list containing the AP scan results. nullptr if scan failed or no access points were found

21.122.2.8 WifiInitScan_SPI()

```
nbWifiScanResult * WifiInitScan_SPI (
            int irqNum = -1,
            int moduleNum = -1,
            int csNum = -1,
            int connectorNum = -1,
            int gpioPinNum = -1,
            int resetPinNum = -1 )
```

Initializes the WiFi hardware, initializes the driver over the SPI bus, and performs an AP scan.

Parameters

in	*irqNum*	IRQ signal used to communicate with the WiFi module.
in	*moduleNum*	SPI module number used to communicate with the WiFi module.
in	*csNum*	Chip Select signal used for SPI communication.
in	*connectorNum*	physical header number of the NetBurner module used.
in	*gpioPinNum*	GPIO pin number to use for the SPI chip select if not using the SPI's native chip select signal.
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.

Returns

nbWifiScanResult Head of linked list containing the AP scan results. nullptr if scan failed or no access points were found

Initializes the WiFi Driver to communicate with the WiFi module over the SPI bus and performs a scan for surrounding access points (AP). The result of the scan is returned as the head of a linked list that contains the scan results. If the WiFi driver has already been initialized by a previous call to one of the WifiInit function variations, then only a scan will be performed.

21.122.2.9 WifiInitScanAndShow_Serial()

```
int WifiInitScanAndShow_Serial (
            int portNum = -1,
            int resetPinNum = -1,
            int connectorNum = -1 )
```

Initializes the WiFi hardware, initializes the driver using the UART interface, performs an AP scan, and prints the scan results via serial output.
NOTE: Serial functions pertaining to WiFi are only available for the NBWIFIIN.

Parameters

in	*portNum*	UART number
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.
in	*connectorNum*	physical header number of the NetBurner module used.

Returns

the Wifi interface number if successful. Otherwise, returns NB::Error::GeneralErrors or NB::Error::Init::InitializationErrors

21.122.2.10 WifiInitScanAndShow_SPI()

```
int WifiInitScanAndShow_SPI (
            int irqNum = -1,
            int moduleNum = -1,
            int csNum = -1,
            int connectorNum = -1,
            int gpioPinNum = -1,
            int resetPinNum = -1 )
```

Initializes the WiFi hardware, initializes the driver using the SPI bus, performs an AP scan, and prints the scan results via serial output.

Parameters

in	*irqNum*	IRQ signal used to communicate with the WiFi module.
in	*moduleNum*	SPI module number used to communicate with the WiFi module.
in	*csNum*	Chip Select signal used for SPI communication.
in	*connectorNum*	physical header number of the NetBurner module used.
in	*gpioPinNum*	GPIO pin number to use for the SPI chip select if not using the SPI's native chip select signal.
in	*resetPinNum*	pin number to use as an reset signal output to the wifi module.

Returns

returns the Wifi interface number, if successful. Otherwise, returns NB::Error::GeneralErrors or NB::Error::Init::InitializationErrors

Chapter 22

Namespace Documentation

22.1 canMCF5441x Namespace Reference

canMCF5441x namespace

Classes

- class CanRxMessage

  Class to hold received CAN messages.

22.1.1 Detailed Description

canMCF5441x namespace

22.2 mcanMODM7AE70 Namespace Reference

mcanMODM7AE70 namespace

Classes

- class CanRxMessage

  Class to hold received CAN messages.
- class mcan_config

  MCAN configuration structure.
- class mcan_module

  MCAN Module Class.

Variables

- const uint32_t CONF_MCAN_RX_FIFO_0_NUM = 32
- const uint32_t CONF_MCAN_RX_FIFO_1_NUM = 1
- const uint32_t CONF_MCAN_RX_BUFFER_NUM = 1
- const uint32_t CONF_MCAN_TX_BUFFER_NUM = 8
- const uint32_t CONF_MCAN_TX_FIFO_QUEUE_NUM = 1
- const uint32_t CONF_MCAN_TX_EVENT_FIFO = 8
- const uint32_t CONF_MCAN_RX_STANDARD_ID_FILTER_NUM = 32
- const uint32_t CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM = 32

22.2.1 Detailed Description

mcanMODM7AE70 namespace

22.3 NB::Error Namespace Reference

Namespaces

- namespace Connect
- namespace Init
- namespace Scan

Enumerations

- enum GeneralErrors {
  NoError = 0 , Timeout = -256 , BusTimeout = -257 , InvalidArgument = -258 ,
  TooManyPendingCommands = -259 , InvalidRequest = -512 }

22.3.1 Detailed Description

NB::Error namespace that describes WiFi request errors. See also InitWifi_SPI(), InitAP_SPI(), InitWifi_Serial(), NB::Master::ConnectToAP(), NB::Master::StartAP(), WifiInitScanAndShow_SPI(), WifiInitScanAndShow_Serial()

22.3.2 Enumeration Type Documentation

22.3.2.1 GeneralErrors

`enum NB::Error::GeneralErrors`

22.3.3 GeneralErrors

8 bit values reserved for command specific errors

Enumerator

NoError	Successful execution.
Timeout	Request timed out.
BusTimeout	Request timed out on the bus.
InvalidArgument	Parameter/arguement not defined or is invalid.
TooManyPendingCommands	Too many pending commands.
InvalidRequest	Invalid request.

22.4 NB::Error::Connect Namespace Reference

Enumerations

- enum ConnectErrors {
  Success = 0 , NotInitialized = -1 , AlreadyConnected = -2 , Option = -3 ,
  CouldNotConfig = -4 , SSID_NotFound = -5 , BSSID_NotFound = -6 , Sec_NotFound = -7 ,
  Cipher_NotFound = -8 , ConnectFailed = -9 }

22.4.1 Detailed Description

NB::Error::Connect namespace that describes WiFi connect request errors. See also InitWifi_SPI(), InitAP_SPI(), InitWifi_Serial(), NB::Master::ConnectToAP(), NB::Master::StartAP()

22.4.2 Enumeration Type Documentation

22.4.2.1 ConnectErrors

enum NB::Error::Connect::ConnectErrors

22.4.3 ConnectErrors

Enumerator

Success	Success.
NotInitialized	Not Initialized.
AlreadyConnected	Already Connected.
Option	Option.
CouldNotConfig	Could Not Configure.
SSID_NotFound	SSID Not Found.
BSSID_NotFound	BSSID Not Found.
Sec_NotFound	Sec Not Found.
Cipher_NotFound	Cipher Not Found.
ConnectFailed	Connect Failed.

22.5 NB::Error::Init Namespace Reference

Enumerations

- enum InitializationErrors {
  Success = 0 , AlreadyInit = -1 , NoDevice = -2 , InvalidInfo = -3 ,
  DevFirmVer = -4 , DevHwVer = -5 , OptionTables = -6 }

22.5.1 Detailed Description

NB::Error::Init namespace that describes WiFi initialization request errors. See also WifiInitScanAndShow_SPI(), WifiInitScanAndShow_Serial(), InitWifi_SPI(), InitAP_SPI(), InitWifi_Serial()

22.5.2 Enumeration Type Documentation

22.5.2.1 InitializationErrors

enum NB::Error::Init::InitializationErrors

22.5.3 InitializationErrors

Enumerator

Success	Success.
AlreadyInit	Already Inittialized.
NoDevice	No Device detected or attempting to use WiFi functions on an un-initialized device.
InvalidInfo	Invalid Info.
DevFirmVer	Device Firmware Version.
DevHwVer	Device Hardware Version.
OptionTables	Option Tables.

22.6 NB::Error::Scan Namespace Reference

Enumerations

- enum ScanErrors { Success = 0 , NotInitialized = -1 , InProgress = -2 , Option = -3 }

22.6.1 Detailed Description

NB::Error::Scan namespace that describes WiFi access point scan request errors. See also WifiInitScanAndShow_SPI(), WifiInitScanAndShow_Serial()

22.6.2 Enumeration Type Documentation

22.6.2.1 ScanErrors

enum NB::Error::Scan::ScanErrors

22.6.3 ScanErrors

Enumerator

Success	Success.
NotInitialized	Not Initialized.
InProgress	In Progress.
Option	Option.

Chapter 23

Class Documentation

23.1 _FlexSPIConfig Struct Reference

FlexSPI Memory Configuration Block.
`#include <evkmimxrt1060_flexspi_nor_config.h>`

Public Attributes

- uint32_t **tag**

  [0x000-0x003] Tag, fixed value 0x42464346UL
- uint32_t **version**

  [0x004-0x007] Version,[31:24] -'V', [23:16] - Major, [15:8] - Minor, [7:0] - bugfix
- uint32_t **reserved0**

  [0x008-0x00b] Reserved for future use
- uint8_t **readSampleClkSrc**

  [0x00c-0x00c] Read Sample Clock Source, valid value: 0/1/3
- uint8_t **csHoldTime**

  [0x00d-0x00d] CS hold time, default value: 3
- uint8_t **csSetupTime**

  [0x00e-0x00e] CS setup time, default value: 3
- uint8_t columnAddressWidth
- uint8_t deviceModeCfgEnable

  Serial NAND, need to refer to datasheet.
- uint8_t deviceModeType
- uint16_t waitTimeCfgCommands

  Generic configuration, etc.
- flexspi_lut_seq_t deviceModeSeq

  DPI/QPI/OPI switch or reset command.
- uint32_t deviceModeArg

  sequence number, [31:16] Reserved
- uint8_t **configCmdEnable**

  [0x01c-0x01c] Configure command Enable Flag, 1 - Enable, 0 - Disable
- uint8_t **configModeType** [3]

  [0x01d-0x01f] Configure Mode Type, similar as deviceModeTpe
- flexspi_lut_seq_t **configCmdSeqs** [3]

  [0x020-0x02b] Sequence info for Device Configuration command, similar as deviceModeSeq
- uint32_t **reserved1**

  [0x02c-0x02f] Reserved for future use
- uint32_t **configCmdArgs** [3]

[0x030-0x03b] Arguments/Parameters for device Configuration commands

- uint32_t **reserved2**

  [0x03c-0x03f] Reserved for future use
- uint32_t controllerMiscOption
- uint8_t deviceType

  details
- uint8_t **sflashPadType**

  [0x045-0x045] Serial Flash Pad Type: 1 - Single, 2 - Dual, 4 - Quad, 8 - Octal
- uint8_t serialClkFreq
- uint8_t lutCustomSeqEnable

  Chapter for more details.
- uint32_t reserved3 [2]

  be done using 1 LUT sequence, currently, only applicable to HyperFLASH
- uint32_t **sflashA1Size**

  [0x050-0x053] Size of Flash connected to A1
- uint32_t **sflashA2Size**

  [0x054-0x057] Size of Flash connected to A2
- uint32_t **sflashB1Size**

  [0x058-0x05b] Size of Flash connected to B1
- uint32_t **sflashB2Size**

  [0x05c-0x05f] Size of Flash connected to B2
- uint32_t **csPadSettingOverride**

  [0x060-0x063] CS pad setting override value
- uint32_t **sclkPadSettingOverride**

  [0x064-0x067] SCK pad setting override value
- uint32_t **dataPadSettingOverride**

  [0x068-0x06b] data pad setting override value
- uint32_t **dqsPadSettingOverride**

  [0x06c-0x06f] DQS pad setting override value
- uint32_t **timeoutInMs**

  [0x070-0x073] Timeout threshold for read status command
- uint32_t **commandInterval**

  [0x074-0x077] CS deselect interval between two commands
- uint16_t **dataValidTime** [2]

  [0x078-0x07b] CLK edge to data valid time for PORT A and PORT B, in terms of 0.1ns
- uint16_t **busyOffset**

  [0x07c-0x07d] Busy offset, valid value: 0-31
- uint16_t busyBitPolarity
- uint32_t lookupTable [64]

  busy flag is 0 when flash device is busy
- flexspi_lut_seq_t **lutCustomSeq** [12]

  [0x180-0x1af] Customizable LUT Sequences
- uint32_t **reserved4** [4]

  [0x1b0-0x1bf] Reserved for future use

23.1.1 Detailed Description

FlexSPI Memory Configuration Block.

23.1.2 Member Data Documentation

23.1.2.1 busyBitPolarity

uint16_t _FlexSPIConfig::busyBitPolarity
[0x07e-0x07f] Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -

23.1.2.2 columnAddressWidth

uint8_t _FlexSPIConfig::columnAddressWidth
[0x00f-0x00f] Column Address with, for HyperBus protocol, it is fixed to 3, For

23.1.2.3 controllerMiscOption

uint32_t _FlexSPIConfig::controllerMiscOption
[0x040-0x043] Controller Misc Options, see Misc feature bit definitions for more

23.1.2.4 deviceModeArg

uint32_t _FlexSPIConfig::deviceModeArg
sequence number, [31:16] Reserved
[0x018-0x01b] Argument/Parameter for device configuration

23.1.2.5 deviceModeCfgEnable

uint8_t _FlexSPIConfig::deviceModeCfgEnable
Serial NAND, need to refer to datasheet.
[0x010-0x010] Device Mode Configure enable flag, 1 - Enable, 0 - Disable

23.1.2.6 deviceModeSeq

flexspi_lut_seq_t _FlexSPIConfig::deviceModeSeq
DPI/QPI/OPI switch or reset command.
[0x014-0x017] Device mode sequence info, [7:0] - LUT sequence id, [15:8] - LUt

23.1.2.7 deviceModeType

uint8_t _FlexSPIConfig::deviceModeType
[0x011-0x011] Specify the configuration command type:Quad Enable, DPI/QPI/OPI switch,

23.1.2.8 deviceType

uint8_t _FlexSPIConfig::deviceType
details
[0x044-0x044] Device Type: See Flash Type Definition for more details

23.1.2.9 lookupTable

uint32_t _FlexSPIConfig::lookupTable[64]
busy flag is 0 when flash device is busy
[0x080-0x17f] Lookup table holds Flash command sequences

23.1.2.10 lutCustomSeqEnable

uint8_t _FlexSPIConfig::lutCustomSeqEnable
Chapter for more details.
[0x047-0x047] LUT customization Enable, it is required if the program/erase cannot

23.1.2.11 reserved3

uint32_t _FlexSPIConfig::reserved3[2]
be done using 1 LUT sequence, currently, only applicable to HyperFLASH
[0x048-0x04f] Reserved for future use

23.1.2.12 serialClkFreq

`uint8_t _FlexSPIConfig::serialClkFreq`
[0x046-0x046] Serial Flash Frequencey, device specific definitions, See System Boot

23.1.2.13 waitTimeCfgCommands

`uint16_t _FlexSPIConfig::waitTimeCfgCommands`
Generic configuration, etc.
[0x012-0x013] Wait time for all configuration commands, unit: 100us, Used for
The documentation for this struct was generated from the following file:

- evkmimxrt1060_flexspi_nor_config.h

23.2 _lut_sequence Struct Reference

FlexSPI LUT Sequence structure.
`#include <evkmimxrt1060_flexspi_nor_config.h>`

Public Attributes

- uint8_t **seqNum**

  Sequence Number, valid number: 1-16.
- uint8_t **seqId**

  Sequence Index, valid number: 0-15.

23.2.1 Detailed Description

FlexSPI LUT Sequence structure.
The documentation for this struct was generated from the following file:

- evkmimxrt1060_flexspi_nor_config.h

23.3 _ocotp_timing Struct Reference

OCOTP timing structure. Note that, these value are used for calcalating the read/write timings. And the values should statisfy below rules:
`#include <fsl_ocotp.h>`

Public Attributes

- uint32_t wait
- uint32_t relax
- uint32_t strobe_prog
- uint32_t strobe_read

23.3.1 Detailed Description

OCOTP timing structure. Note that, these value are used for calcalating the read/write timings. And the values should statisfy below rules:
Tsp_rd=(WAIT+1)/ipg_clk_freq should be >= 150ns; Tsp_pgm=(RELAX+1)/ipg_clk_freq should be >= 100ns; Trd = ((STROBE_READ+1)- 2∗(RELAX_READ+1)) /ipg_clk_freq, The Trd is required to be larger than 40 ns. Tpgm = ((STROBE_PROG+1)- 2∗(RELAX_PROG+1)) /ipg_clk_freq; The Tpgm should be configured within the range of 9000 ns < Tpgm < 11000 ns;
The documentation for this struct was generated from the following files:

- PlatformSpecific/SOMRT1061/EFFS_MultiPart/src/fsl_ocotp.h
- Web/SimpleHtml/src/fsl_ocotp.h

23.4 _PinVector Class Reference

GPIO Pin Vector Base Class.
`#include <cpu_pins.h>`
Inherited by PinVector< n >.

Public Member Functions

- uint32_t operator= (uint32_t val)

  Assign a value to the _PinVector Bus.
- PinIO operator[ ] (int idx)

  Access the PinIO for a specific bit position in the _PinVector.
- void config (uint32_t idx, PinIO cfg)

  Set the PinIO that will be used for a given bit position in the _PinVector.
- void config (PinIO *pinCfgs, uint32_t count)

  Configure the _PinVector based on an array of PinIOs. The index of the PinIO in the configuration array will determine the bit position within the _PinVector that that PinIO represents.
- operator uint32_t () const

  Read the line state of the _PinVector bus.

23.4.1 Detailed Description

GPIO Pin Vector Base Class.
This class is for building semi-parallel buses using GPIO pins. It facilitates driving a numeric value across a parallel bus of GPIO pins. It should not be confused with a true parallel bus, as each individual pin in the bus is modified sequentially/independently during a bus assignment.

Parameters

_Pin::len	Number of pins managed by the _PinVector.

23.4.2 Member Function Documentation

23.4.2.1 config() [1/2]

```
void _PinVector::config (
            PinIO * pinCfgs,
            uint32_t count )
```

Configure the _PinVector based on an array of PinIOs. The index of the PinIO in the configuration array will determine the bit position within the _PinVector that that PinIO represents.

Parameters

pinCfgs	The configuration to be used.
count	The number of PinIOs in the configuration array to be used.

23.4.2.2 config() [2/2]

```
void _PinVector::config (
            uint32_t idx,
            PinIO cfg )
```

Set the PinIO that will be used for a given bit position in the _PinVector.

Parameters

idx	The bit position to be configured.
cfg	The PinIO that bit position will represent.

23.4.2.3 operator uint32_t()

```
_PinVector::operator uint32_t ( ) const
```

Read the line state of the _PinVector bus.

Returns

The current bus value as read from the line state.

23.4.2.4 operator=()

```
uint32_t _PinVector::operator= (
            uint32_t val )
```

Assign a value to the _PinVector Bus.

Parameters

val	The value to assign to the bus.

Returns

The value driven on the bus.

23.4.2.5 operator[]()

```
PinIO _PinVector::operator[] (
            int idx )
```

Access the PinIO for a specific bit position in the _PinVector.

Parameters

idx	The bit index to access in the _PinVector.

Returns

The PinIO for the given bit position.

The documentation for this class was generated from the following file:

- cortex-m7/cpu/SAME70/include/cpu_pins.h

23.5 aes_context Struct Reference

AES context structure.

```
#include <aes.h>
```

Public Attributes

- unsigned long erk [64]
- unsigned long drk [64]
- int nr

23.5.1 Detailed Description

AES context structure.

23.5.2 Member Data Documentation

23.5.2.1 drk

`unsigned long aes_context::drk[64]`
decryption round keys

23.5.2.2 erk

`unsigned long aes_context::erk[64]`
encryption round keys

23.5.2.3 nr

`int aes_context::nr`
number of rounds

The documentation for this struct was generated from the following file:

- aes.h

23.6 ARM_MPU_Region_t Struct Reference

`#include <mpu_armv7.h>`

Public Attributes

- uint32_t **RBAR**

  The region base address register value (RBAR)
- uint32_t **RASR**

  The region attribute and size register value (RASR)

23.6.1 Detailed Description

Struct for a single MPU Region
The documentation for this struct was generated from the following file:

- mpu_armv7.h

23.7 CallBackFunctionPageHandler Class Reference

Implements the HtmlPageHandler class as a function pointer callback for GET requests.
`#include <http.h>`
Inherits HtmlPageHandler.

Public Member Functions

- virtual int ProcessRaw (int sock, HTTP_Request &pdt)

  This class will do a callback with data for each request to the specified url.
- CallBackFunctionPageHandler (const char *pUrl, http_gethandlerfunc *pFunction, HTTP_RequestTypes reqType=tGet, int accessGroup=0, bool beforeFiles=false)

  Constructor for HTTP GET callback function.
- CallBackFunctionPageHandler (const char *pUrl, http_gethandlerfunc *pFunction, http_matchhandlerfunc *pMatchFunction, HTTP_RequestTypes reqType=tGet, int accessGroup=0, bool beforeFiles=false)

  Constructor for HTTP GET callback function, includes option for function to match the requested name.

Public Member Functions inherited from HtmlPageHandler

- HtmlPageHandler (const char *url, HTTP_RequestTypes rt=tGet, int accessGroup=0, bool Before_↩ Files=false)

  Register handler.
- virtual int ProcessRaw (int sock, HTTP_Request &pd)=0

  This class will do a callback with data for each request to the specified url.
- int GetGroup ()

  Returns access group setting.

Additional Inherited Members

Protected Member Functions inherited from HtmlPageHandler

- void **InsertSort** (HtmlPageHandler *&ph)

  Insert sort.
- int **SortValue** (HtmlPageHandler *pv)

  Returns the value of the sort compare: -1, 0, 1.

Protected Attributes inherited from HtmlPageHandler

- HtmlPageHandler * **m_pNextHandler**

  Pointer to next page handle object.
- const char * **m_pUrlName**

  Pointer to URL. Performs a length match, an empty string matches everything.
- int **m_access_group**

  The access group for this request see CheckHttpAccess.
- HTTP_RequestTypes **m_requestTypes**

  Type of request, HTTP_RequestTypes.

23.7.1 Detailed Description

Implements the HtmlPageHandler class as a function pointer callback for GET requests.

23.7.2 Constructor & Destructor Documentation

23.7.2.1 CallBackFunctionPageHandler() [1/2]

```
CallBackFunctionPageHandler::CallBackFunctionPageHandler (
            const char * pUrl,
            http_gethandlerfunc * pFunction,
            HTTP_RequestTypes reqType = tGet,
            int accessGroup = 0,
            bool beforeFiles = false )  [inline]
```

Constructor for HTTP GET callback function.

Parameters

pUrl	Pointer to the URL to intercept from normal system processing
pFunction	Pointer to the function to call for the requested URL
reqType	Optional type of request to intercept, HTTP_RequestTypes. Default is tGet
accessGroup	Optional password group access level. Default is 0 = no password
beforeFiles	Optional parameter to specify a check for just a name, or the name of an actual file in the system. true = intercept before checking for a file of the same name false = check for any files of the requested name before intercepting

23.7.2.2 CallBackFunctionPageHandler() [2/2]

```
CallBackFunctionPageHandler::CallBackFunctionPageHandler (
            const char * pUrl,
            http_gethandlerfunc * pFunction,
            http_matchhandlerfunc * pMatchFunction,
            HTTP_RequestTypes reqType = tGet,
            int accessGroup = 0,
            bool beforeFiles = false )  [inline]
```

Constructor for HTTP GET callback function, includes option for function to match the requested name.

Parameters

pUrl	Pointer to the URL to intercept from normal system processing
pFunction	Pointer to the function to call for the requested URL
pMatchFunction	Pointer to function to perform the match check
reqType	Optional type of request to intercept, HTTP_RequestTypes. Default is tGet
accessGroup	Optional parameter to set a password group access level. Default is 0 = no password.
beforeFiles	Optional parameter to process a file name before checking for a compiled-in file of the same name. Default is false = check for any files of the requested name before intercepting

23.7.3 Member Function Documentation

23.7.3.1 ProcessRaw()

```
virtual int CallBackFunctionPageHandler::ProcessRaw (
            int sock,
            HTTP_Request & pd )  [inline], [virtual]
```

This class will do a callback with data for each request to the specified url.

Returns

0 if the request was not processed.

Implements HtmlPageHandler.
The documentation for this class was generated from the following file:

- http.h

23.8 CallBackFunctionPostHandler Class Reference

Implements the HtmlPostHandler class as a function pointer callback for POST requests.

```
#include <httppost.h>
```
Inherits HtmlPostHandler.

Public Member Functions

- virtual int ProcessRaw (int sock, HTTP_Request &pdt)

  This class will do a callback with data for each request to the specified url.

23.8.1 Detailed Description

Implements the HtmlPostHandler class as a function pointer callback for POST requests.

23.8.2 Member Function Documentation

23.8.2.1 ProcessRaw()

```
virtual int CallBackFunctionPostHandler::ProcessRaw (
            int sock,
            HTTP_Request & pd )  [inline], [virtual]
```
This class will do a callback with data for each request to the specified url.

Returns

0 if the request was not processed.

Implements HtmlPageHandler.
The documentation for this class was generated from the following file:

- httppost.h

23.9 canMCF5441x::CanRxMessage Class Reference

Class to hold received CAN messages.
```
#include <multican.h>
```

Public Member Functions

- uint8_t **GetLength** ()

  Returns the amount of data stored in the message.
- uint8_t GetData (uint8_t *buffer, uint8_t max_len)

  Copy the data in the message up to max_len.
- uint32_t **GetId** ()

  Returns the ID of the message.
- uint16_t **GetTimeStamp** ()

  Returns the time stamp of the message.
- BOOL IsValid ()

  Check to verify the CanRxMessage is a valid message.
- CanRxMessage (OS_FIFO *pFifo, uint16_t timeout)

  Build a CanRxMessage from a FIFO.
- CanRxMessage (int moduleNum, uint32_t id, uint16_t timeout)

  Constructor that sends out a RTR request to ID and waits for a response.
- ~**CanRxMessage** ()

  Destructor.
- BOOL GetNewMessage (OS_FIFO *pFifo, uint16_t timeout)

  Get a new message from the FIFO. If no message is available, wait up to the timeout for one to be received.

23.9.1 Detailed Description

Class to hold received CAN messages.

23.9.2 Constructor & Destructor Documentation

23.9.2.1 CanRxMessage() [1/2]

```
canMCF5441x::CanRxMessage::CanRxMessage (
            OS_FIFO * pFifo,
            uint16_t timeout )
```

Build a CanRxMessage from a FIFO.
The FIFO must be registered to listen for incoming messages. If no messages are received in the timeout interval, the returned CanRxMessage will be marked as not valid. A Timeout value of 0 will wait forever.

Parameters

pFifo	Pointer to an OS_FIFO to store the message
timeout	Message timeout in TICKS_PER_SECOND

23.9.2.2 CanRxMessage() [2/2]

```
canMCF5441x::CanRxMessage::CanRxMessage (
            int moduleNum,
            uint32_t id,
            uint16_t timeout )
```

Constructor that sends out a RTR request to ID and waits for a response.
The CAN system uses any unused channel to send and receive the buffer. The constructor can return an invalid message for two reasons:

1. The timeout interval has expired
2. There were no free CAN channels available to send the request

Parameters

moduleNum	CAN module number
id	CAN ID number
timeout	Message timeout in TICKS_PER_SECOND. A value of 0 will wait forever.

23.9.3 Member Function Documentation

23.9.3.1 GetData()

```
uint8_t canMCF5441x::CanRxMessage::GetData (
            uint8_t * buffer,
            uint8_t max_len )
```

Copy the data in the message up to max_len.

Parameters

buffer	Pointer to destination storage buffer

Parameters

max_len	Maximum number of bytes to copy

Returns

The number of bytes copied

23.9.3.2 GetNewMessage()

```
BOOL canMCF5441x::CanRxMessage::GetNewMessage (
            OS_FIFO * pFifo,
            uint16_t timeout )
```

Get a new message from the FIFO. If no message is available, wait up to the timeout for one to be received.

Parameters

pFifo	Pointer to the OS_FIFO.
timeout	Timeout to wait if no messages are currently available. Value is in TICKS_PER_SECOND, a value of 0 will wait forever.

23.9.3.3 IsValid()

```
BOOL canMCF5441x::CanRxMessage::IsValid ( )
```

Check to verify the CanRxMessage is a valid message.
Please refer to the CanRxMessage() constructor for further information on valid messages.
The documentation for this class was generated from the following file:

- multican.h

23.10 mcanMODM7AE70::CanRxMessage Class Reference

Class to hold received CAN messages.
`#include <mcan.h>`

Public Member Functions

- uint8_t GetLength ()

  Returns the amount of data stored in the message.
- uint8_t CopyData (uint8_t *buffer, uint8_t max_len)

  Copy the data in the message up to max_len.
- const uint8_t * GetData ()

  Returns a pointer to the message data.
- uint32_t GetId ()

  Returns the ID of the message.
- uint16_t GetTimeStamp ()

  Returns the time stamp of the message.
- BOOL IsValid ()

  Check to verify the CanRxMessage is a valid message.
- CanRxMessage (OS_FIFO *pFifo, uint32_t timeout=WAIT_FOREVER)

  Build a CanRxMessage from a FIFO.

- ~CanRxMessage ()

  Destructor.
- BOOL GetNewMessage (OS_FIFO *pFifo, uint32_t timeout=WAIT_FOREVER)

  Get a new message from the FIFO. If no message is available, wait up to the timeout for one to be received.

23.10.1 Detailed Description

Class to hold received CAN messages.

23.10.2 Constructor & Destructor Documentation

23.10.2.1 CanRxMessage()

```
mcanMODM7AE70::CanRxMessage::CanRxMessage (
            OS_FIFO * pFifo,
            uint32_t timeout = WAIT_FOREVER )
```

Build a CanRxMessage from a FIFO.
The FIFO must be registered to listen for incoming messages. If no messages are received in the timeout interval, the returned CanRxMessage will be marked as not valid. A Timeout value of 0 will wait forever.

Parameters

pFifo	Pointer to an OS_FIFO to store the message
timeout	Optional message timeout in TICKS_PER_SECOND

23.10.2.2 ~CanRxMessage()

```
mcanMODM7AE70::CanRxMessage::~CanRxMessage ( )
```

Destructor.

23.10.3 Member Function Documentation

23.10.3.1 CopyData()

```
uint8_t mcanMODM7AE70::CanRxMessage::CopyData (
            uint8_t * buffer,
            uint8_t max_len )
```

Copy the data in the message up to max_len.

Parameters

buffer	Pointer to destination storage buffer
max_len	Maximum number of bytes to copy

Returns

The number of bytes copied

23.10.3.2 GetData()

```
const uint8_t * mcanMODM7AE70::CanRxMessage::GetData ( )
```

Returns a pointer to the message data.

Warning

Invalid after destructor is called

23.10.3.3 GetId()

```
uint32_t mcanMODM7AE70::CanRxMessage::GetId ( )
```

Returns the ID of the message.

Returns

The message ID

23.10.3.4 GetLength()

```
uint8_t mcanMODM7AE70::CanRxMessage::GetLength ( )
```

Returns the amount of data stored in the message.

Returns

Length of data stored

23.10.3.5 GetNewMessage()

```
BOOL mcanMODM7AE70::CanRxMessage::GetNewMessage (
            OS_FIFO * pFifo,
            uint32_t timeout = WAIT_FOREVER )
```

Get a new message from the FIFO. If no message is available, wait up to the timeout for one to be received.

Parameters

pFifo	Pointer to the OS_FIFO
timeout	Optional parameter to wait if no messages are currently available. Value is in TICKS_PER_SECOND, a value of 0 will wait forever.

Returns

True on success, otherwise false

23.10.3.6 GetTimeStamp()

```
uint16_t mcanMODM7AE70::CanRxMessage::GetTimeStamp ( )
```

Returns the time stamp of the message.

For timestamp generation the MCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1-16). The counter is readable via MCAN_TSCV.TSC. A write access to the Timestamp Counter Value register (MCAN_TSCV) resets the counter to zero. When the timestamp counter wraps around, interrupt flag MCAN_IR.TSW is set. On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.
The internal/external Timestamp Counter value is captured on start of frame (both Receive and Transmit). When MCAN_TSCC.TSS = 1, the Timestamp Counter is incremented in multiples of CAN bit times [1-16] depending on the configuration of MCAN_TSCC.TCP. A wrap around sets interrupt flag MCAN_IR.TSW. Write access resets the counter to zero.

Returns

Time stamp of the message

23.10.3.7 IsValid()

`BOOL mcanMODM7AE70::CanRxMessage::IsValid ( )`
Check to verify the CanRxMessage is a valid message.
Please refer to the CanRxMessage() constructor for further information on valid messages.

Returns

True if valid, otherwise false

The documentation for this class was generated from the following file:

- mcan.h

23.11 config_bool Class Reference

Boolean Configuration Variable.
`#include <config_obj.h>`
Inherits config_value.
Inherited by reboot_obj.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Copy the object value as a text string to the specified NBString object.
- config_bool (config_obj &owner, bool def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_bool (bool def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator bool () const

  Return the object value.
- config_bool & operator= (const bool v)

  Assign the config_bool object value to the specified bool value.
- config_bool & operator= (const config_bool &cb)

  Copy one config_bool object to another.
- config_bool & operator= (const int i)

  Assign a config_bool object value to the specified integer value.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.11.1 Detailed Description

Boolean Configuration Variable.

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.11.2 Constructor & Destructor Documentation

23.11.2.1 config_bool() [1/2]

```
config_bool::config_bool (
            config_obj & owner,
            bool def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.

Parameters

owner	Reference to parent leaf
def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.11.2.2 config_bool() [2/2]

```
config_bool::config_bool (
            bool def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.

Parameters

def_val	Default value
name	Field name assigned to the value
desc	Description, used for info/tool tips

23.11.3 Member Function Documentation

23.11.3.1 GetTextValue()

```
virtual void config_bool::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Copy the object value as a text string to the specified NBString object.
Writes either "true" or "false". This is a virtual function that can be overridden in a derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.11.3.2 GetTypeValue()

```
virtual void config_bool::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value to the specified NBString object.

Parameters

s	NBString object to hold the type value string

23.11.3.3 operator bool()

```
config_bool::operator bool ( ) const  [inline]
```

Return the object value.

Returns

The object value as a bool

Example: for a variable named configInt of type config_double:

```
bool boolValue = bool(NV_Settings.m_nvBool);
iprintf("boolValue: %d\r\n", boolValue);
```

23.11.3.4 operator=() [1/3]

```
config_bool & config_bool::operator= (
            const bool v )  [inline]
```

Assign the config_bool object value to the specified bool value.

Parameters

v	Value of type bool

23.11.3.5 operator=() [2/3]

```
config_bool & config_bool::operator= (
            const config_bool & cb )  [inline]
```

Copy one config_bool object to another.

Parameters

cb	Reference to a config_bool object

23.11.3.6 operator=() [3/3]

```
config_bool & config_bool::operator= (
            const int i )  [inline]
```

Assign a config_bool object value to the specified integer value.

Parameters

i	Integer value, either 0 (false) or 1 (true)

The documentation for this class was generated from the following file:

- config_obj.h

23.12 config_chooser Class Reference

Chooser Configuration Variable - Select From a List of Items.
`#include <config_obj.h>`
Inherits config_obj.

Public Member Functions

- config_chooser (config_obj &owner, const char *name, const char *in_value, const char *in_choices, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_chooser (const char *name, const char *in_value, const char *in_choices, const char *desc=NULL)

  Object constructor.
- bool IsSelected (const char *choice)

  Check if a particular choice option is selected.
- bool IsSelected (const NBString &s)

  Check if a particular choice option is selected.
- bool IsInChoices (const char *str, size_t strLen)

  Check if a string is in the list of possible choices. A comparison will continue until a null character is found, or the length of the longer of the two strings being compared is reached (strLen is used to determine if str is longer than the current element being compared.)
- bool IsInChoices (const NBString &str, size_t strLen)

  Check if a string is in the list of possible choices. A comparison will continue until a null character is found, or the length of the longer of the two strings being compared is reached (strLen is used to determine if str is longer than the current element being compared.)
- const config_string & GetChoices ()

  Get the list of choices.
- const config_string & SetChoices (const char *in_choices)

  Set the list of choices.
- operator NBString () const

  Returns the object value.
- config_chooser & operator= (const char *p)

  Assign the selected list item from a const char value.*
- config_chooser & operator= (const NBString &s)

  Assign the config_string object value from a NBString object.
- config_chooser & operator= (const config_chooser &ci)

  Copy one config_chooser object to another.
- virtual void GetTypeValue (NBString &s)

  Assigns the object type value to the specified NBString object.

Public Member Functions inherited from config_obj

- config_obj (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_obj (const char *name, const char *desc)

  Object constructor.
- virtual void GetTextValue (NBString &s)

  Get the object value as a text string to the specified NBString object.
- virtual void GetTypeValue (NBString &s)

  Assigns the object type value to the specified NBString object.

23.12.1 Detailed Description

Chooser Configuration Variable - Select From a List of Items.
The config_chooser class provides a list of options with the ability to select one. If used on a web page it would appear as a drop down box. For example, the choices could be "one", "two", or "three", and "two" could be the item selected. Since the items are JSON, the selected item is identified by the option name, not a list index.

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.12.2 Constructor & Destructor Documentation

23.12.2.1 config_chooser() [1/2]

```
config_chooser::config_chooser (
            config_obj & owner,
            const char * name,
            const char * in_value,
            const char * in_choices,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.

Parameters

owner	Reference to owner/parent leaf
name	Field name assigned to the value
in_value	The option value selected
in_choices	The list of option choices
desc	Description used for info/tool tip

23.12.2.2 config_chooser() [2/2]

```
config_chooser::config_chooser (
            const char * name,
            const char * in_value,
            const char * in_choices,
            const char * desc = NULL )  [inline]
```

Object constructor.

Parameters

name	Field name assigned to the value
in_value	The option value selected
in_choices	The list of option choices
desc	Description used for info/tool tip

23.12.3 Member Function Documentation

23.12.3.1 GetChoices()

```
const config_string & config_chooser::GetChoices ( )  [inline]
```

Get the list of choices.

Returns

A config_string object containing the list of choices

23.12.3.2 GetTypeValue()

```
virtual void config_chooser::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Assigns the object type value to the specified NBString object.
The type value "object" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

Reimplemented from config_obj.

23.12.3.3 IsInChoices() [1/2]

```
bool config_chooser::IsInChoices (
            const char * str,
            size_t strLen )  [inline]
```

Check if a string is in the list of possible choices. A comparison will continue until a null character is found, or the length of the longer of the two strings being compared is reached (strLen is used to determine if str is longer than the current element being compared.)

Parameters

str	NBString to attempt to find inside the list of choices
strLen	Length of str. This value is used to determine the amount of characters to compare for each element in the list of choices.

Return values

true	if str is found as an option in list of choices
false	if str is not found in the list of choices

23.12.3.4 IsInChoices() [2/2]

```
bool config_chooser::IsInChoices (
            const NBString & str,
            size_t strLen ) [inline]
```

Check if a string is in the list of possible choices. A comparison will continue until a null character is found, or the length of the longer of the two strings being compared is reached (strLen is used to determine if str is longer than the current element being compared.)

Parameters

str	NBString to attempt to find inside the list of choices
strLen	Length of str. This value is used to determine the amount of characters to compare for each element in the list of choices.

Return values

true	if str is found as an option in list of choices
false	if str is not found in the list of choices

23.12.3.5 IsSelected() [1/2]

```
bool config_chooser::IsSelected (
            const char * choice ) [inline]
```

Check if a particular choice option is selected.

Parameters

choice	Choice to test as a string type

Return values

true	Selected
false	Not Selected

23.12.3.6 IsSelected() [2/2]

```
bool config_chooser::IsSelected (
            const NBString & s ) [inline]
```

Check if a particular choice option is selected.

Parameters

s	Choice to test as a NBString type

Return values

true	Selected
false	Not Selected

23.12.3.7 operator NBString()

```
config_chooser::operator NBString ( ) const  [inline]
```

Returns the object value.

Returns

The currently selected list option as a NBString

23.12.3.8 operator=() [1/3]

```
config_chooser & config_chooser::operator= (
            const char * p )  [inline]
```

Assign the selected list item from a const char* value.

Parameters

p	Selected list item to assign

23.12.3.9 operator=() [2/3]

```
config_chooser & config_chooser::operator= (
            const config_chooser & ci )  [inline]
```

Copy one config_chooser object to another.

Parameters

ci	Reference to a config_chooser object

23.12.3.10 operator=() [3/3]

```
config_chooser & config_chooser::operator= (
            const NBString & s )  [inline]
```

Assign the config_string object value from a NBString object.

Parameters

s	Reference to a NBString object

23.12.3.11 SetChoices()

```
const config_string & config_chooser::SetChoices (
            const char * in_choices )  [inline]
```

Set the list of choices.

Parameters

in_choices	The list of option choices

Returns

A config_string object containing the list of choices

The documentation for this class was generated from the following file:

- config_obj.h

23.13 config_double Class Reference

Double Float Configuration Variable.
`#include <config_obj.h>`
Inherits config_value.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Copy the object value as a text string to the specified NBString object.
- config_double (config_obj &owner, double def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_double (double def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator int () const

  Return the object value as an int.
- operator float () const

  Return the object value as a float.
- operator double () const

  Returns the object value as a double float.
- config_double & operator= (const double d)

  Assign the config_double object value from a double value.
- config_double & operator= (const config_double &ci)

  Copy one config_double object to another.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value in the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.13.1 Detailed Description

Double Float Configuration Variable.

Note

All modifications to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.13.2 Constructor & Destructor Documentation

23.13.2.1 config_double() [1/2]

```
config_double::config_double (
            config_obj & owner,
            double def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.

Parameters

owner	Reference to parent leaf
def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.13.2.2 config_double() [2/2]

```
config_double::config_double (
            double def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.

Parameters

def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.13.3 Member Function Documentation

23.13.3.1 GetTextValue()

```
virtual void config_double::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Copy the object value as a text string to the specified NBString object.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.13.3.2 GetTypeValue()

```
virtual void config_double::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value in the specified NBString object.
The type value "float" is written to the NBString object.

Parameters

s	Reference to a NBString object

23.13.3.3 operator double()

```
config_double::operator double ( ) const  [inline]
```

Returns the object value as a double float.

Returns

The double float value of the object

Example: for a variable named configInt of type config_double:

```
double value = double(configInt);
iprintf("int = %f\r\n", value);
```

23.13.3.4 operator float()

```
config_double::operator float ( ) const  [inline]
```

Return the object value as a float.

Returns

The float value of the object

Example: for a variable named configFloat of type config_double:

```
float value = float(configFloat);
iprintf("int = %f\r\n", value);
```

23.13.3.5 operator int()

```
config_double::operator int ( ) const  [inline]
```

Return the object value as an int.

Returns

The integer value of the object

Example: for a variable named configInt of type config_double:

```
int value = int(configInt);
iprintf("int = %d\r\n", value);
```

23.13.3.6 operator=() [1/2]

```
config_double & config_double::operator= (
            const config_double & ci )  [inline]
```

Copy one config_double object to another.

Parameters

ci	Reference to a config_double object

23.13.3.7 operator=() [2/2]

```
config_double & config_double::operator= (
            const double d )  [inline]
```

Assign the config_double object value from a double value.

Parameters

d	Value of type double

The documentation for this class was generated from the following file:

- config_obj.h

23.14 config_int Class Reference

Signed 32-bit Integer Configuration Variable.
`#include <config_obj.h>`
Inherits config_value.
Inherited by version_obj.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Copy the object value to the specified NBString object.
- config_int (config_obj &owner, int def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_int (int def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator int () const

  Return the object value as an int.
- config_int & operator= (const int i)

  Assign the config_int object value to the specified int value.
- config_int & operator= (const config_int &ci)

  Copy one config_int object to another.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value in the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.14.1 Detailed Description

Signed 32-bit Integer Configuration Variable.

Note

All modifications to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.14.2 Constructor & Destructor Documentation

23.14.2.1 config_int() [1/2]

```
config_int::config_int (
            config_obj & owner,
            int def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.

Parameters

owner	Reference to owner/parent leaf
def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.14.2.2 config_int() [2/2]

```
config_int::config_int (
            int def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.

Parameters

def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.14.3 Member Function Documentation

23.14.3.1 GetTextValue()

```
virtual void config_int::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Copy the object value to the specified NBString object.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.14.3.2 GetTypeValue()

```
virtual void config_int::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value in the specified NBString object.
The type value "integer" is written to the NBString object.

Parameters

s	Reference to a NBString object

23.14.3.3 operator int()

```
config_int::operator int ( ) const  [inline]
```

Return the object value as an int.

Returns

The integer value of the object

Example: for a variable named configInt of type config_int:

```
int value = int(configInt);
iprintf("int = %d\r\n", value);
```

23.14.3.4 operator=() [1/2]

```
config_int & config_int::operator= (
            const config_int & ci )  [inline]
```

Copy one config_int object to another.

Parameters

ci	Reference to a config_int object

23.14.3.5 operator=() [2/2]

```
config_int & config_int::operator= (
            const int i )  [inline]
```

Assign the config_int object value to the specified int value.

Parameters

i	Integer value

The documentation for this class was generated from the following file:

- config_obj.h

23.15 config_IPADDR Class Reference

Configuration Variable for IPADDR (IPv6) object type.

`#include <config_obj.h>`

Inherits config_value.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Get the object value as a text string with quotations to the specified NBString object.
- config_IPADDR (config_obj &owner, IPADDR def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.

- config_IPADDR (IPADDR def_val, const char *name, const char *desc=NULL)

  Object constructor.
- config_IPADDR (config_obj &owner, const char *def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_IPADDR (const char *def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator IPADDR () const

  Returns the object value.
- bool IsNull () const

  Check if the IP address is null.
- bool NotNull () const

  Check if the IP address is not null.
- void **SetNull** ()

  Set the IP address value of an config_IPADDR object to null.
- config_IPADDR & operator= (const IPADDR &i6)

  Copy an IPADDR object value to a config_IPADDR object.
- config_IPADDR & operator= (const config_IPADDR &ci)

  Copy one config_IPADDR object to another.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.15.1 Detailed Description

Configuration Variable for IPADDR (IPv6) object type.

See also

config_IPADDR4

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.15.2 Constructor & Destructor Documentation

23.15.2.1 config_IPADDR() [1/4]

```
config_IPADDR::config_IPADDR (
            config_obj & owner,
            IPADDR def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with an IPADDR object

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type IPADDR
name	Field name assigned to the value
desc	Description used for info/tool tip

23.15.2.2 config_IPADDR() [2/4]

```
config_IPADDR::config_IPADDR (
            IPADDR def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with an IPADDR object

Parameters

def_val	Default value of type NBString
name	Field name assigned to the value
desc	Description used for info/tool tip

23.15.2.3 config_IPADDR() [3/4]

```
config_IPADDR::config_IPADDR (
            config_obj & owner,
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with an const char * string. IP address is converted using the SetFromAscii() IPADDR member function.

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type char * in IPv6 format
name	Field name assigned to the value
desc	Description used for info/tool tip

23.15.2.4 config_IPADDR() [4/4]

```
config_IPADDR::config_IPADDR (
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with an const char * string. IP address is converted using the SetFromAscii() IPADDR member function.

Parameters

def_val	Default value of type char * in IPv6 format
name	Field name assigned to the value
desc	Description used for info/tool tip

23.15.3 Member Function Documentation

23.15.3.1 GetTextValue()

```
virtual void config_IPADDR::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Get the object value as a text string with quotations to the specified NBString object.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.15.3.2 GetTypeValue()

```
virtual void config_IPADDR::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value to the specified NBString object.
The type value "string" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

23.15.3.3 IsNull()

```
bool config_IPADDR::IsNull ( ) const  [inline]
```

Check if the IP address is null.

Return values

true	if the IP address value is null

See also

NotNull()

23.15.3.4 NotNull()

```
bool config_IPADDR::NotNull ( ) const  [inline]
```

Check if the IP address is not null.

Return values

true	if the IP address value is not null

See also

IsNull()

23.15.3.5 operator IPADDR()

```
config_IPADDR::operator IPADDR ( ) const  [inline]
```

Returns the object value.

Returns

The value of the object as an IPADDR object

23.15.3.6 operator=() [1/2]

```
config_IPADDR & config_IPADDR::operator= (
            const config_IPADDR & ci )  [inline]
```

Copy one config_IPADDR object to another.

Parameters

ci	Reference to a config_IPADDR object

23.15.3.7 operator=() [2/2]

```
config_IPADDR & config_IPADDR::operator= (
            const IPADDR & i6 )  [inline]
```

Copy an IPADDR object value to a config_IPADDR object.

Parameters

i6	Reference to an IPADDR object

The documentation for this class was generated from the following file:

- config_obj.h

23.16 config_IPADDR4 Class Reference

Configuration Variable for IPADDR4 (IPv4) object types.
`#include <config_obj.h>`
Inherits config_value.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Get the object value as a text string with quotations to the specified NBString object.
- config_IPADDR4 (config_obj &owner, IPADDR4 def_val, const char *name, const char *desc=NULL)

Object constructor with the parent/owner leaf parameter.

- config_IPADDR4 (IPADDR4 def_val, const char *name, const char *desc=NULL)

  Object constructor.
- config_IPADDR4 (config_obj &owner, const char *def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_IPADDR4 (const char *def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator IPADDR4 () const

  Returns the object value.
- bool IsNull () const

  Check if the IP address is null.
- bool NotNull () const

  Check if the IP address is not null.
- void SetNull ()

  Set the IP address to null.
- config_IPADDR4 & operator= (const IPADDR4 &i4)

  Copy an IPADDR4 object value to a config_IPADDR4 object.
- config_IPADDR4 & operator= (const config_IPADDR4 &ci)

  Copy one config_IPADDR4 object to another.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.16.1 Detailed Description

Configuration Variable for IPADDR4 (IPv4) object types.

See also

config_IPADDR

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.16.2 Constructor & Destructor Documentation

23.16.2.1 config_IPADDR4() [1/4]

```
config_IPADDR4::config_IPADDR4 (
            config_obj & owner,
            IPADDR4 def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with an IPADDR4 object

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type IPADDR4
name	Field name assigned to the value
desc	Description used for info/tool tip

23.16.2.2 config_IPADDR4() [2/4]

```
config_IPADDR4::config_IPADDR4 (
            IPADDR4 def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with an IPADDR4 object

Parameters

def_val	Default value of type NBString
name	Field name assigned to the value
desc	Description used for info/tool tip

23.16.2.3 config_IPADDR4() [3/4]

```
config_IPADDR4::config_IPADDR4 (
            config_obj & owner,
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with an const char * string. IP address is converted using the IPADDR4 SetFromAscii() member function.

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type char *. Format is: "xxx.xxx.xxx.xxx". For example, "10.1.1.100"
name	Field name assigned to the value
desc	Description used for info/tool tip

23.16.2.4 config_IPADDR4() [4/4]

```
config_IPADDR4::config_IPADDR4 (
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with an const char * string. IP address is converted using the IPADDR4 SetFromAscii() member function.

Parameters

def_val	Default value of type char *. Format is: "xxx.xxx.xxx.xxx". For example, "10.1.1.100"
name	Field name assigned to the value
desc	Description used for info/tool tip

23.16.3 Member Function Documentation

23.16.3.1 GetTextValue()

```
virtual void config_IPADDR4::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Get the object value as a text string with quotations to the specified NBString object.
The format of the string is: "xxx.xxx.xxx.xxx". This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.16.3.2 GetTypeValue()

```
virtual void config_IPADDR4::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value to the specified NBString object.
The type value "string" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

23.16.3.3 IsNull()

```
bool config_IPADDR4::IsNull ( ) const  [inline]
```

Check if the IP address is null.

Return values

true	if the IP address value is null

See also

NotNull(), SetNull()

23.16.3.4 NotNull()

```
bool config_IPADDR4::NotNull ( ) const  [inline]
```

Check if the IP address is not null.

Return values

true	if the IP address value is not null

See also

IsNull(), SetNull()

23.16.3.5 operator IPADDR4()

```
config_IPADDR4::operator IPADDR4 ( ) const  [inline]
```

Returns the object value.

Returns

The value of the object as an IPADDR4

23.16.3.6 operator=() [1/2]

```
config_IPADDR4 & config_IPADDR4::operator= (
            const config_IPADDR4 & ci )  [inline]
```

Copy one config_IPADDR4 object to another.

Parameters

ci	Reference to a config_IPADDR4 object

23.16.3.7 operator=() [2/2]

```
config_IPADDR4 & config_IPADDR4::operator= (
            const IPADDR4 & i4 )  [inline]
```

Copy an IPADDR4 object value to a config_IPADDR4 object.

Parameters

i4	Reference to an IPADDR4 object

23.16.3.8 SetNull()

```
void config_IPADDR4::SetNull ( )  [inline]
```

Set the IP address to null.

See also

IsNull(), NotNull()

The documentation for this class was generated from the following file:

- config_obj.h

23.17 config_MACADR Class Reference

Configuration Variable for MACADR object type.

#include <config_obj.h>
Inherits config_value.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Get the object value as a text string with quotations to the specified NBString object.
- config_MACADR (config_obj &owner, MACADR def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_MACADR (MACADR def_val, const char *name, const char *desc=NULL)

  Object constructor.
- config_MACADR (config_obj &owner, const char *def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_MACADR (const char *def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator MACADR () const

  Returns the object value.
- config_MACADR & operator= (const config_MACADR &ci)

  Copy one config_MACADR object to another.
- config_MACADR & operator= (const MACADR &ci)

  Copy a MACADR object value to a MACADR object.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.17.1 Detailed Description

Configuration Variable for MACADR object type.

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.17.2 Constructor & Destructor Documentation

23.17.2.1 config_MACADR() [1/4]

```
config_MACADR::config_MACADR (
            config_obj & owner,
            MACADR def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with an MACADR object

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type MACADR
name	Field name assigned to the value
desc	Description used for info/tool tip

23.17.2.2 config_MACADR() [2/4]

```
config_MACADR::config_MACADR (
            MACADR def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with an MACADR object

Parameters

def_val	Default value of type MACADR
name	Field name assigned to the value
desc	Description used for info/tool tip

23.17.2.3 config_MACADR() [3/4]

```
config_MACADR::config_MACADR (
            config_obj & owner,
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value is assigned with a const char * string. The AsciiToMac() function is used to convert the string to a MACADR.

Parameters

owner	Reference to owner/parent leaf
def_val	Default MAC Address value of type char *. The hexidical format must be 6 octets, with or without ':' separators. For example, "0123456789AB", or "01:23:45:67:89:AB"
name	Field name assigned to the value
desc	Description used for info/tool tip

23.17.2.4 config_MACADR() [4/4]

```
config_MACADR::config_MACADR (
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value is assigned with a const char * string. The AsciiToMac() function is used to convert the string to a MACADR.

Parameters

def_val	Default MAC Address value of type char *. The hexidical format must be 6 octets, with or without ':' separators. For example, "0123456789AB", or "01:23:45:67:89:AB"
name	Field name assigned to the value
desc	Description used for info/tool tip

23.17.3 Member Function Documentation

23.17.3.1 GetTextValue()

```
virtual void config_MACADR::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Get the object value as a text string with quotations to the specified NBString object.
The format is the hexadecimal address: "xx:xx:xx:xx:xx:xx" This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.17.3.2 GetTypeValue()

```
virtual void config_MACADR::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value to the specified NBString object.
The type value "string" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

23.17.3.3 operator MACADR()

```
config_MACADR::operator MACADR ( ) const  [inline]
```

Returns the object value.

Returns

The value of the object as a MACADR

23.17.3.4 operator=() [1/2]

```
config_MACADR & config_MACADR::operator= (
            const config_MACADR & ci )  [inline]
```

Copy one config_MACADR object to another.

Parameters

ci	Reference to a config_IPADDR object

23.17.3.5 operator=() [2/2]

```
config_MACADR & config_MACADR::operator= (
            const MACADR & ci )  [inline]
```

Copy a MACADR object value to a MACADR object.

Parameters

ci	Reference to an MACADR object

The documentation for this class was generated from the following file:

- config_obj.h

23.18 config_obj Class Reference

Base class used to create configuration objects.
`#include <config_obj.h>`
Inherits config_leaf.
Inherited by AcmeObject, AcmeServletObject, ControlSet, EmailNotify, I4Record, I6Record, InterfaceBlock, LLDPEntity, MonitorRecord, NV_OneI2CPortSetting, NV_OnePortSetting, NV_SettingsStruct, OneConfigRecord, PIDSet, SysRecord, TempRange, ThermoGroup, Thermostat1, cfg_class_slot, cfg_class_slots, config_MqttClient, config_MqttLWT, config_chooser, config_preserver_obj, config_uart, detached_root_obj, empty_config_obj, and root_obj.

Public Member Functions

- config_obj (config_obj &owner, const char ∗name, const char ∗desc)

  Object constructor with the parent/owner leaf parameter.
- config_obj (const char ∗name, const char ∗desc)

  Object constructor.
- virtual void GetTextValue (NBString &s)

  Get the object value as a text string to the specified NBString object.
- virtual void GetTypeValue (NBString &s)

  Assigns the object type value to the specified NBString object.

23.18.1 Detailed Description

Base class used to create configuration objects.
Base that can be used to create your own custom configuration objects in a JSON compatible format. The object can hold any number of JSON compatible types: integer, boolean, null, string, object or array.
A config_obj can hold multiple values in a JSON object. A config_value holds only a single JSON value.

See also

config_value

The example below demonstrates how to create a thermostat object with a temperature range derived from a config_obj:

Note

Please check the Configuration Class example in the \nburn\examples folder for the latest code updates.

```
 The use of config objects should be done at a global scope.  They can contain any number of members,
 which should be given a default value and a name used as an identifier.  The name value for each
 member variable should be unique, otherwise it can lead to issues inside the config tree.

class TempRange : public config_obj
{
   public:
    // Class variables
    config_int m_maxTemp{100, "MaxTemp"};
    config_int m_setTemp{30,  "SetTemp"};
    config_int m_minTemp{10,  "MinTemp"};
    ConfigEndMarker;   // No new data members below this line

    // Class constructors
    TempRange(const char *name, const char *desc = nullptr) : config_obj(name, desc){};
    TempRange(config_obj &owner, const char *name, const char *desc = nullptr) : config_obj(owner, name,
      desc){};
};

class Thermostat : public TempRange
{
   public:
    config_int      m_tempRec1{101, "TempRecord_1"};
    config_int      m_tempRec2{102, "TempRecord_2"};
    config_bool     m_active{true, "Active"};
    config_string   m_location{"Warehouse 1", "Thermostat Location"};
    config_chooser  m_tempScale{"Temperature Scale", "Fahrenheit", "Fahrenheit,Celsius,Kelvin"};
    ConfigEndMarker;   // No new data members below this line

    Thermostat(const char *name, const char *desc = nullptr) : TempRange(name, desc){};
    Thermostat(config_obj &owner, const char *name, const char *desc = nullptr) : TempRange(owner, name,
      desc){};
};

static Thermostat thermo(appdata, "Thermostat", "The primary thermostat");
```

Note

All modifications to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.18.2 Constructor & Destructor Documentation

23.18.2.1 config_obj() [1/2]

```
config_obj::config_obj (
            config_obj & owner,
            const char * name,
            const char * desc )  [inline]
```

Object constructor with the parent/owner leaf parameter.
For example,

```
myObject(config_obj &owner, const char *name, const char *desc = nullptr) : config_obj(owner, name, desc){};
```

Parameters

owner	Reference to owner/parent leaf
name	Field name
desc	Description used for info/tool tip

23.18.2.2 config_obj() [2/2]

```
config_obj::config_obj (
            const char * name,
            const char * desc )  [inline]
```

Object constructor.
This constructor should be called as part of the derived class constructor. For example,
```
myObject(const char *name, const char *desc = nullptr) : config_obj(name, desc){}
```

Parameters

name	Field name
desc	Description used for info/tool tip

23.18.3 Member Function Documentation

23.18.3.1 GetTextValue()

```
virtual void config_obj::GetTextValue (
            NBString & s )  [virtual]
```
Get the object value as a text string to the specified NBString object.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

23.18.3.2 GetTypeValue()

```
virtual void config_obj::GetTypeValue (
            NBString & s )  [inline], [virtual]
```
Assigns the object type value to the specified NBString object.
The type value "object" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

Reimplemented in config_chooser.
The documentation for this class was generated from the following file:

- config_obj.h

23.19 config_pass Class Reference

Password string Configuration Variable.
`#include <config_obj.h>`
Inherits config_string.

Public Member Functions

- config_pass (config_obj &owner, NBString def_val, const char *name, const char *desc=NULL)
  Object constructor with the parent/owner leaf parameter.
- config_pass (NBString def_val, const char *name, const char *desc=NULL)
  Object constructor.
- config_pass (config_obj &owner, const char *def_val, const char *name, const char *desc=NULL)
  Object constructor with the parent/owner leaf parameter.

- config_pass (const char *def_val, const char *name, const char *desc=NULL)

  Object constructor.
- virtual void GetTextValue (NBString &s)

  Get the config_pass object value as a text string.
- virtual void GetRawValue (NBString &s)

  Copy the raw config_string object value to the NBString object.
- operator NBString () const

  Returns the object value.
- config_pass & operator= (const char *p)

  Assign the config_pass object value from a const char value.*
- config_pass & operator= (const NBString &s)

  Assign the config_pass object value from a NBString object.
- config_pass & operator= (const config_string &ci)

  Copy a config_string object to a config_pass object.
- config_pass & operator= (const config_pass &ci)

  Copy one config_pass object to another.

Public Member Functions inherited from config_string

- virtual void GetTextValue (NBString &s)

  Get the object value (as a text string with quotations) to the specified NBString object.
- config_string (config_obj &owner, NBString def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_string (NBString def_val, const char *name, const char *desc=NULL)

  Object constructor.
- config_string (config_obj &owner, const char *def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_string (const char *def_val, const char *name, const char *desc=NULL)

  Object constructor.
- void SetEnumList (NBString s)

  Renders the data used to explain the schema/descriptions for the list of choices.
- operator NBString () const

  Return the object value.
- config_string & operator= (const char *p)

  Assign the config_string object value from a const char string.*
- config_string & operator= (const NBString &s)

  Assign the config_string object value from a NBString object.
- config_string & operator= (const config_string &ci)

  Copy one config_string object to another.
- const char * c_str () const

  Returns the object value as a string.
- size_t length () const

  Returns the string length in bytes.
- const char & operator[ ] (size_t pos) const

  Return the value of a character in the string.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.19.1 Detailed Description

Password string Configuration Variable.
A config_pass is like a config_string, but the data is hidden by '*' characters.

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.19.2 Constructor & Destructor Documentation

23.19.2.1 config_pass() [1/4]

```
config_pass::config_pass (
            config_obj & owner,
            NBString def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value is assigned with a NBString

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type NBString
name	Field name assigned to the value
desc	Description used for info/tool tip

23.19.2.2 config_pass() [2/4]

```
config_pass::config_pass (
            NBString def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value is assigned with a NBString

Parameters

def_val	Default value of type NBString
name	Field name assigned to the value
desc	Description used for info/tool tip

23.19.2.3 config_pass() [3/4]

```
config_pass::config_pass (
            config_obj & owner,
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value is assigned with a character string

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type char *
name	Field name assigned to the value
desc	Description used for info/tool tip

23.19.2.4 config_pass() [4/4]

```
config_pass::config_pass (
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value is assigned with a character string

Parameters

def_val	Default value of type char *
name	Field name assigned to the value
desc	Description used for info/tool tip

23.19.3 Member Function Documentation

23.19.3.1 GetRawValue()

```
virtual void config_pass::GetRawValue (
            NBString & s )  [virtual]
```

Copy the raw config_string object value to the NBString object.

Parameters

s	Reference to a NBString object

This is a virtual function that can be overridden in any derived class.

23.19.3.2 GetTextValue()

```
virtual void config_pass::GetTextValue (
            NBString & s )  [virtual]
```

Get the config_pass object value as a text string.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Reimplemented from config_string.

23.19.3.3 operator NBString()

```
config_pass::operator NBString ( ) const  [inline]
```

Returns the object value.

Returns

The value of the object as a NBString

23.19.3.4 operator=() [1/4]

```
config_pass & config_pass::operator= (
            const char * p )  [inline]
```

Assign the config_pass object value from a const char* value.

Parameters

p	String to assign

23.19.3.5 operator=() [2/4]

```
config_pass & config_pass::operator= (
            const config_pass & ci )  [inline]
```

Copy one config_pass object to another.

Parameters

ci	Reference to a config_pass object

23.19.3.6 operator=() [3/4]

```
config_pass & config_pass::operator= (
            const config_string & ci )  [inline]
```

Copy a config_string object to a config_pass object.

Parameters

ci	Reference to a config_string object

23.19.3.7 operator=() [4/4]

```
config_pass & config_pass::operator= (
            const NBString & s )  [inline]
```

Assign the config_pass object value from a NBString object.

Parameters

s	Reference to a NBString object

The documentation for this class was generated from the following file:

- config_obj.h

23.20 config_string Class Reference

String Configuration Variable.
`#include <config_obj.h>`
Inherits config_value.
Inherited by config_localname, config_pass, and config_time_t.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Get the object value (as a text string with quotations) to the specified NBString object.
- config_string (config_obj &owner, NBString def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_string (NBString def_val, const char *name, const char *desc=NULL)

  Object constructor.
- config_string (config_obj &owner, const char *def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_string (const char *def_val, const char *name, const char *desc=NULL)

  Object constructor.
- void SetEnumList (NBString s)

  Renders the data used to explain the schema/descriptions for the list of choices.
- operator NBString () const

  Return the object value.
- config_string & operator= (const char *p)

  Assign the config_string object value from a const char string.*
- config_string & operator= (const NBString &s)

  Assign the config_string object value from a NBString object.
- config_string & operator= (const config_string &ci)

  Copy one config_string object to another.
- const char * c_str () const

  Returns the object value as a string.
- size_t length () const

  Returns the string length in bytes.
- const char & operator[ ] (size_t pos) const

  Return the value of a character in the string.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Friends

- class **config_pass**
- class **config_chooser**

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.20.1 Detailed Description

String Configuration Variable.

Note

All modifictions to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.20.2 Constructor & Destructor Documentation

23.20.2.1 config_string() [1/4]

```
config_string::config_string (
            config_obj & owner,
            NBString def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with a NBString

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type NBString
name	Field name assigned to the value
desc	Description used for info/tool tip

23.20.2.2 config_string() [2/4]

```
config_string::config_string (
            NBString def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with a NBString

Parameters

def_val	Default value of type NBString
name	Field name assigned to the value
desc	Description used for info/tool tip

23.20.2.3 config_string() [3/4]

```
config_string::config_string (
            config_obj & owner,
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.
Default value assigned with a character string

Parameters

owner	Reference to owner/parent leaf
def_val	Default value of type char *
name	Field name assigned to the value
desc	Description used for info/tool tip

23.20.2.4 config_string() [4/4]

```
config_string::config_string (
            const char * def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.
Default value assigned with a character string

Parameters

def_val	Default value of type char *
name	Field name assigned to the value
desc	Description used for info/tool tip

23.20.3 Member Function Documentation

23.20.3.1 c_str()

```
const char * config_string::c_str ( ) const  [inline]
```

Returns the object value as a string.

Returns

String value of object

23.20.3.2 GetTextValue()

```
virtual void config_string::GetTextValue (
            NBString & s )  [virtual]
```

Get the object value (as a text string with quotations) to the specified NBString object.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.
Reimplemented in config_pass.

23.20.3.3 GetTypeValue()

```
virtual void config_string::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value to the specified NBString object.
The type value "string" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

23.20.3.4 length()

```
size_t config_string::length ( ) const  [inline]
```

Returns the string length in bytes.

Returns

String length

23.20.3.5 operator NBString()

```
config_string::operator NBString ( ) const  [inline]
```

Return the object value.

Returns

The value of the object as a NBString

23.20.3.6 operator=() [1/3]

```
config_string & config_string::operator= (
            const char * p )  [inline]
```

Assign the config_string object value from a const char* string.

Parameters

p	String to assign

23.20.3.7 operator=() [2/3]

```
config_string & config_string::operator= (
            const config_string & ci )  [inline]
```

Copy one config_string object to another.

Parameters

ci	Reference to a config_string object

23.20.3.8 operator=() [3/3]

```
config_string & config_string::operator= (
            const NBString & s )  [inline]
```

Assign the config_string object value from a NBString object.

Parameters

s	Reference to a NBString object

23.20.3.9 operator[]()

```
const char & config_string::operator[] (
            size_t pos ) const  [inline]
```

Return the value of a character in the string.

Parameters

pos	Character index/position in the string

Returns

Character value at the specified index

23.20.3.10 SetEnumList()

```
void config_string::SetEnumList (
            NBString s )  [inline]
```

Renders the data used to explain the schema/descriptions for the list of choices.

Parameters

s	Enumerated list that is a string with choices separated by commas (',') with no spaces. For example, "one,two,three"

The documentation for this class was generated from the following file:

- config_obj.h

23.21 config_uint Class Reference

Unsigned 32-bit Integer Configuration Variable.
`#include <config_obj.h>`
Inherits config_value.
Inherited by config_hex_uint.

Public Member Functions

- virtual void GetTextValue (NBString &s)

  Copy the object value to the specified NBString object.
- config_uint (config_obj &owner, uint32_t def_val, const char *name, const char *desc=NULL)

  Object constructor with the parent/owner leaf parameter.
- config_uint (uint32_t def_val, const char *name, const char *desc=NULL)

  Object constructor.
- operator uint32_t () const

  Return the object value.
- config_uint & operator= (const uint32_t i)

  Assign the config_uint object value from a uint32_t value.
- config_uint & operator= (const config_uint &ci)

  Copy one config_uint object to another.
- virtual void GetTypeValue (NBString &s)

  Copy the object type value to the specified NBString object.

Additional Inherited Members

Protected Member Functions inherited from config_value

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.21.1 Detailed Description

Unsigned 32-bit Integer Configuration Variable.

Note

All modifications to configuration objects are marked as pending. A call to SaveConfigToStorage is required to save changes to flash memory.

23.21.2 Constructor & Destructor Documentation

23.21.2.1 config_uint() [1/2]

```
config_uint::config_uint (
            config_obj & owner,
            uint32_t def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor with the parent/owner leaf parameter.

Parameters

owner	Reference to owner/parent leaf
def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.21.2.2 config_uint() [2/2]

```
config_uint::config_uint (
            uint32_t def_val,
            const char * name,
            const char * desc = NULL )  [inline]
```

Object constructor.

Parameters

def_val	Default value
name	Field name assigned to the value
desc	Description used for info/tool tip

23.21.3 Member Function Documentation

23.21.3.1 GetTextValue()

```
virtual void config_uint::GetTextValue (
            NBString & s )  [inline], [virtual]
```

Copy the object value to the specified NBString object.
This is a virtual function that can be overridden in any derived class.

Parameters

s	Reference to a NBString object to store the result

Implements config_value.

23.21.3.2 GetTypeValue()

```
virtual void config_uint::GetTypeValue (
            NBString & s )  [inline], [virtual]
```

Copy the object type value to the specified NBString object.
The type value "integer" is written to the NBString object.

Parameters

s	NBString object to hold the type value string

23.21.3.3 operator uint32_t()

```
config_uint::operator uint32_t ( ) const  [inline]
```

Return the object value.

Returns

The value of the object as a uint32_t

Example: for a variable named configuInt of type config_uint:

```
uint32_t value = uint32_t(configuInt);
iprintf("int = %ld\r\n", value);
```

23.21.3.4 operator=() [1/2]

```
config_uint & config_uint::operator= (
            const config_uint & ci )  [inline]
```

Copy one config_uint object to another.

Parameters

ci	Reference to a config_uint object

23.21.3.5 operator=() [2/2]

```
config_uint & config_uint::operator= (
            const uint32_t i )  [inline]
```

Assign the config_uint object value from a uint32_t value.

Parameters

i	uint32_t value

The documentation for this class was generated from the following file:

- config_obj.h

23.22 config_value Class Reference

Base class used to create a configuration value.
`#include <config_obj.h>`
Inherits config_leaf.
Inherited by CUR_IPADDR4, Dynamic_IPADDR, config_IPADDR, config_IPADDR4, config_MACADR, config_bool, config_double, config_int, config_report, config_string, and config_uint.

Protected Member Functions

- config_value (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_value (const char *name, const char *desc)

  Object constructor.

23.22.1 Detailed Description

Base class used to create a configuration value.
Base class that can be used to create custom configuration values in a JSON compatible format. The JSON compatible types are: integer, boolean, null, string, object or array.

The system already provides the common JSON value types, including: config_int, config_uint, config_bool, config_chooser, config_string, config_pass, config_IPADDR4, config_IPADDR, config_MACADR,
A config_obj can hold multiple values in a JSON object. A config_value holds only a single JSON value.

See also

config_obj

23.22.2 Constructor & Destructor Documentation

23.22.2.1 config_value() [1/2]

```
config_value::config_value (
            config_obj & owner,
            const char * name,
            const char * desc )  [inline], [protected]
```

Object constructor with the parent/owner leaf parameter.
For example,

```
myValue(config_obj &owner, const char *name, const char *desc = nullptr) : config_value(owner, name,
    desc){};
```

Parameters

owner	Reference to owner/parent leaf
name	Field name
desc	Description used for info/tool tip

23.22.2.2 config_value() [2/2]

```
config_value::config_value (
            const char * name,
            const char * desc )  [inline], [protected]
```

Object constructor.
This constructor should be called as part of the derived class constructor. For example,

```
myValue(const char *name, const char *desc = nullptr) : config_value(name, desc){}
```

Parameters

name	Field name
desc	Description used for info/tool tip

The documentation for this class was generated from the following file:

- config_obj.h

23.23 DelayObject Class Reference

Microsecond Delay Class.

```
#include <HiResDelay.h>
```

Public Member Functions

- DelayObject (int Timer=FIRST_UNUSED_TIMER)

  Microsecond Delay Timer Constructor.
- void DelayUsec (uint32_t usec)

  Microsecond Delay Timer.
- bool valid ()

  Verify a valid delay object was constructed.

23.23.1 Detailed Description

Microsecond Delay Class.
Creates a high resolution microsecond timer object by allocating one a free processor system timer. Once the delay has expired, the system timer is releaed back to the free pool.
Example Usage:

```
static DelayObject myHiResDelay;    // Create the object
```

```
myHiDrsDelay.DelayUsec(100);        // Delay 100 us
myHiDrsDelay.DelayUsec(200);        // Delay 200 us
myHiDrsDelay.DelayUsec(300);        // Delay 300 us
```

See also

Interval Timer

Stopwatch Timer

OSTimeDly()

23.23.2 Constructor & Destructor Documentation

23.23.2.1 DelayObject()

```
DelayObject::DelayObject (
            int Timer = FIRST_UNUSED_TIMER )
```

Microsecond Delay Timer Constructor.

Parameters

Timer	Optional parameter to select a specific time. Recommend use is the to not specify a timer so the first free timer will be used.

23.23.3 Member Function Documentation

23.23.3.1 DelayUsec()

```
void DelayObject::DelayUsec (
            uint32_t usec )
```

Microsecond Delay Timer.

Parameters

usec	The number of microseconds to delay

23.23.3.2 valid()

```
bool DelayObject::valid ( )  [inline]
```

Verify a valid delay object was constructed.

Return values

True	- A timer was correctly allocated and initialized
False	- No free timers were available

The documentation for this class was generated from the following file:

- HiResDelay.h

23.24 DhcpObject Class Reference

DHCP client class.

`#include <dhcpclient.h>`
Inherits TimeOutElement.

Public Member Functions

- void StartDHCP ()

  Manually start the DHCP Client.
- void StopDHCP ()

  Manually stop the DHCP client and release the DHCP lease.
- void RestartDHCP ()

  Manually stop are resetart the DHCP client.
- bool bDoFallBack ()

  Returns true if a DHCP fallback event has occurred.
- void RenewDHCP ()

  Manually force a DHCP renew of a lease.
- void RebindDHCP ()

  Manually force a DHCP rebind of a lease.
- BOOL ValidDhcpLease ()

  Returns the status of a DHCP lease.
- uint32_t GetRemainingDhcpLeaseTime ()

  Returns number of seconds remaining for current lease.
- uint32_t GetDhcpRenewTime ()

  Returns number of seconds remaining before the lease will be renewed.
- uint32_t GetDhcpRebindTime ()

  Returns number of seconds remaining before the current lease will have to rebind.
- uint32_t GetDhcpExpirationTime ()

  Returns number of seconds remaining before the current lease will expire.
- int32_t GetDHCPState ()

  Returns current state of the DHCP lease.

23.24.1 Detailed Description

DHCP client class.

23.24.2 Member Function Documentation

23.24.2.1 bDoFallBack()

`bool DhcpObject::bDoFallBack ( )`
Returns true if a DHCP fallback event has occurred.
The init() function called at the beginning of an application will handle DHCP client services for all network interfaces. It is rare that you would need to call these functions manually.
If the fallback functionality has been enabled in the device configuration, this function can be used to determine if a fallback to a static IP settings has occurred.

Returns

True if a fallback condition has occurred.

23.24.2.2 GetDhcpExpirationTime()

```
uint32_t DhcpObject::GetDhcpExpirationTime ( )  [inline]
```

Returns number of seconds remaining before the current lease will expire.

Returns

The number of seconds remaining before the current lease will expire.

See also

GetRemainingDhcpLeaseTime(), GetDhcpRenewTime(), GetDhcpRebindTime()

23.24.2.3 GetDhcpRebindTime()

```
uint32_t DhcpObject::GetDhcpRebindTime ( )  [inline]
```

Returns number of seconds remaining before the current lease will have to rebind.

Returns

The number of seconds remaining before the current lease will have to rebind.

See also

GetRemainingDhcpLeaseTime(), GetDhcpRenewTime(), GetDhcpExpirationTime()

23.24.2.4 GetDhcpRenewTime()

```
uint32_t DhcpObject::GetDhcpRenewTime ( )  [inline]
```

Returns number of seconds remaining before the lease will be renewed.

Returns

The number of seconds remaining before the lease will be renewed.

See also

GetRemainingDhcpLeaseTime(), GetDhcpRebindTime(), GetDhcpExpirationTime()

23.24.2.5 GetDHCPState()

```
int32_t DhcpObject::GetDHCPState ( )
```

Returns current state of the DHCP lease.

Returns

DHCP State

See also

ValidDhcpLease()

23.24.2.6 GetRemainingDhcpLeaseTime()

`uint32_t DhcpObject::GetRemainingDhcpLeaseTime ( )`
Returns number of seconds remaining for current lease.

Returns

The number of seconds remaining for current lease.

See also

GetDhcpRenewTime(), GetDhcpRebindTime(), GetDhcpExpirationTime()

23.24.2.7 RebindDHCP()

`void DhcpObject::RebindDHCP ( )`
Manually force a DHCP rebind of a lease.
The init() function called at the beginning of an application will handle DHCP client services for all network interfaces. It is rare that you would need to call these functions manually.

See also

RenewDHCP()

23.24.2.8 RenewDHCP()

`void DhcpObject::RenewDHCP ( )`
Manually force a DHCP renew of a lease.
The init() function called at the beginning of an application will handle DHCP client services for all network interfaces. It is rare that you would need to call these functions manually.

See also

RebindDHCP()

23.24.2.9 RestartDHCP()

`void DhcpObject::RestartDHCP ( )`
Manually stop are resetart the DHCP client.
The init() function called at the beginning of an application will handle DHCP client services for all network interfaces. It is rare that you would need to call these functions manually.

See also

DhcpObject::StartDHCP(), DhcpObject::StopDHCP()

23.24.2.10 StartDHCP()

`void DhcpObject::StartDHCP ( )`
Manually start the DHCP Client.
The init() function called at the beginning of an application will handle DHCP client services for all network interfaces. It is rare that you would need to call these functions manually.
This function will start the DHCP client and return immediately. Check the DHCP status to verify you have successfully obtained a lease. GetDHCPState()

See also

DhcpObject::StopDHCP()

23.24.2.11 StopDHCP()

```
void DhcpObject::StopDHCP ( )
```

Manually stop the DHCP client and release the DHCP lease.
The init() function called at the beginning of an application will handle DHCP client services for all network interfaces. It is rare that you would need to call these functions manually.

See also

DhcpObject::StartDHCP()

23.24.2.12 ValidDhcpLease()

```
BOOL DhcpObject::ValidDhcpLease ( )
```

Returns the status of a DHCP lease.

Returns

True if the interface has a valid lease.

See also

RenewDHCP()

The documentation for this class was generated from the following file:

- dhcpclient.h

23.25 NB::Wifi::driverStatusStruct Struct Reference

```
#include <wifiDriver.h>
```

Public Attributes

- uint8_t **connected**
  1 if connected, 0 if disconnected
- uint8_t **ssidLength**
  Length of the SSID string.
- uint16_t **txPower**
  Transmit power.
- int16_t **rssi**
  Recieved Signal Strength Indicator.
- uint8_t **band**
  Frequency band.
- uint8_t **channel**
  802.11 channel
- uint32_t **maxTxRate**
  Max transmit data throughput.
- uint8_t **security**
  Security used: WEP, WPA, WPA2, open, uknown.
- uint8_t **cipher**
  refer to OPTIONLISTCIPH for defenitions
- MACADR **bssid**
  Basic Service Set Identifier.
- uint8_t **bssType**
  refer to OPTIONLISTBSSTYPE for defenitions

- char **ssid** [SSID_MAX_LEN+1]

  Placeholder for char array pointer.
- uint32_t **tickLastUpdated**

  CPU tick count of when this struct was last updated.

23.25.1 Detailed Description

A struct used to describe the status of the current network connection.
The documentation for this struct was generated from the following file:

- wifiDriver.h

23.26 DSPIModule Class Reference

DSPIModule is a SPI communications driver. It is an object based driver, which allows for low overhead multiplexing between peripherals with different bus configurations.
`#include <dspi.h>`

Public Member Functions

- DSPIModule (uint8_t SPIModule)

  The minimum DSPIModule Constructor. Requires that configuration settings be applied after construction before use.
- DSPIModule (uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  DSPIModule Full DSPIModule Constructor. Will initialize all hardware settings.
- uint8_t Init (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  DSPIModule Full DSPIModule Constructor. Will initialize all hardware settings.
- uint8_t Start (uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Start begins a SPI transaction using the provided buffers.
- uint8_t Tx (uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Tx begins a transmit only SPI transaction using the provided buffer. Silently discards the received data.
- uint8_t Rx (uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Rx begins a receive only SPI transaction using the provided buffer. Holds output line low during transfers.
- bool EnableDMA (bool enableDMA=true)

  EnableDMA configures whether DMA is allowed for qualifying transactions.
- bool DisableDMA ()

  DisableDMA disables DMA for all transactions.
- bool RegisterSem (OS_SEM *finishedSem)

  Registers a semaphore to be posted to upon completion of transactions.
- bool ClrSem ()

  Clears the semaphore registration.
- OS_SEM * GetSem ()

  Gets a pointer to the registered Semaphore.
- bool Done ()

  Returns whether the previously started transaction is complete.
- uint32_t GetActualBaudrate ()

  Returns the actual bus speed to be used.
- bool SetCS (uint8_t CS)

  Modifies the chip select configuration for the driver context such that only the requested chip select is active during transfers.

Static Public Member Functions

- static BOOL Done (uint8_t SPIModule)

  Returns whether the previously started transaction for a given hardware module is complete.

Friends

- uint8_t DSPIInit (uint8_t SPIModule, uint32_t Baudrate, uint8_t QueueBitSize, uint8_t CS, uint8_t CSPol, uint8_t ClkPolarity, uint8_t ClkPhase, BOOL DoutHiz, uint8_t QCD, uint8_t DTL)

  Initialize a DSPI module.

23.26.1 Detailed Description

DSPIModule is a SPI communications driver. It is an object based driver, which allows for low overhead multiplexing between peripherals with different bus configurations.

23.26.2 Constructor & Destructor Documentation

23.26.2.1 DSPIModule() [1/2]

```
DSPIModule::DSPIModule (
            uint8_t SPIModule )
```

The minimum DSPIModule Constructor. Requires that configuration settings be applied after construction before use.

Parameters

SPIModule	The DSPI module to use, from 1 to 3. Default is 1.

See also

DSPIModule::DSPIModule(uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits, uint8_t peripheralChipSelects, uint8_t chipSelectPolarity, uint8_t clockPolarity, uint8_t clockPhase, BOOL doutHiz, uint8_t csToClockDelay, uint8_t delayAfterTransfer)

23.26.2.2 DSPIModule() [2/2]

```
DSPIModule::DSPIModule (
            uint8_t SPIModule,
            uint32_t baudRateInBps,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )
```

DSPIModule Full DSPIModule Constructor. Will initialize all hardware settings.

Parameters

SPIModule	The DSPI module to use, from 1 to 3. Default is 1.
baudRateInBps	Maximum master clock frequency for SPI Bus in bits per second.
transferSizeInBits	Bit width of the transfers to be made.

Parameters

peripheralChipSelects	Bit mask of which chipselects are active during transfers.
chipSelectPolarity	Bit mask of the inactive state of chipselects.
clockPolarity	Inactive level for the clock signal.
clockPhase	Clock phase: • 0: data captured on leading edge of DSPI_CLK, changed on following edge. • 1: data changed on leading edge of DSPI_CLK, captured on following edge.

Parameters

doutHiz	Data output is high impedance between transfers (instead of driven).
csToClockDelay	Delay from chip select to valid clock. The default of 0 will set the delay as close to 1/2 DSPI_CLK without going under, keeping with the interface to QSPI driver.
delayAfterTransfer	Delay between data transfers. The default of 0 is 17/(system clock / 2), in keeping with interface to QSPI

See also

DSPIModule::DSPIModule(uint8_t SPIModule)

DSPIModule::Init(uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits, uint8_t peripheral↩ ChipSelects, uint8_t chipSelectPolarity, uint8_t clockPolarity, uint8_t clockPhase, BOOL doutHiz, uint8_t cs↩ ToClockDelay, uint8_t delayAfterTransfer)

23.26.3 Member Function Documentation

23.26.3.1 ClrSem()

`bool DSPIModule::ClrSem ( ) [inline]`

Clears the semaphore registration.

Returns

Returns whether the registration was successfully cleared.

23.26.3.2 DisableDMA()

`bool DSPIModule::DisableDMA ( ) [inline]`

DisableDMA disables DMA for all transactions.

Returns

Returns whether DMA is allowed

23.26.3.3 Done() [1/2]

`bool DSPIModule::Done ( ) [inline]`

Returns whether the previously started transaction is complete.

Returns

Whether the previously started transaction is complete

23.26.3.4 Done() [2/2]

```
static BOOL DSPIModule::Done (
            uint8_t SPIModule )  [static]
```

Returns whether the previously started transaction for a given hardware module is complete.

Parameters

SPIModule	The hardware instance to check the transaction state for.

Returns

Whether the previously started transaction is complete

23.26.3.5 EnableDMA()

```
bool DSPIModule::EnableDMA (
            bool enableDMA = true )
```

EnableDMA configures whether DMA is allowed for qualifying transactions.

Parameters

enableDMA	Whether DMA is allowed

Returns

Returns whether DMA is allowed

23.26.3.6 GetActualBaudrate()

```
uint32_t DSPIModule::GetActualBaudrate ( )  [inline]
```

Returns the actual bus speed to be used.

Returns

The actual bus speed to be used.

23.26.3.7 GetSem()

```
OS_SEM * DSPIModule::GetSem ( )  [inline]
```

Gets a pointer to the registered Semaphore.

Returns

Pointer to the registered Semaphore

23.26.3.8 Init()

```
uint8_t DSPIModule::Init (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
```

```
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )
```

DSPIModule Full DSPIModule Constructor. Will initialize all hardware settings.
param SPIModule The instance number of the hardware module to use with this driver context.

Parameters

baudRateInBps	Maximum master clock frequency for SPI Bus
transferSizeInBits	Bit width of the transfers to be made
peripheralChipSelects	Bit mask of which chipselects are active during transfers
chipSelectPolarity	Bit mask of the inactive state of chipselects
clockPolarity	Inactive level for the clock signal
clockPhase	• 0: data captured on leading edge of DSPI_CLK, changed on following edge • 1: data changed on leading edge of DSPI_CLK, captured on following edge

Parameters

doutHiz	Data output is high impedance between transfers (instead of driven)
csToClockDelay	Delay form chip select to valid clock. The default of 0 will set the delay as close to 1/2 DSPI_CLK without going under, keeping with the interface to QSPI driver.
delayAfterTransfer	Delay between data transfers. The default of 0 is 17/(system clock / 2), in keepin with interface to QSPI

See also

DSPIModule::DSPIModule(uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits, uint8_t peripheralChipSelects, uint8_t chipSelectPolarity, uint8_t clockPolarity, uint8_t clockPhase, BOOL doutHiz, uint8_t csToClockDelay, uint8_t delayAfterTransfer)

23.26.3.9 RegisterSem()

```
bool DSPIModule::RegisterSem (
            OS_SEM * finishedSem )
```

Registers a semaphore to be posted to upon completion of transactions.

Parameters

finishedSem	A pointer to the semaphore to be posted to.

Returns

Returns whether the semaphore was successfully registered.

23.26.3.10 Rx()

```
uint8_t DSPIModule::Rx (
            uint8_t * receiveBufferPtr,
```

```
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline]
```

Rx begins a receive only SPI transaction using the provided buffer. Holds output line low during transfers.

Parameters

receiveBufferPtr	A pointer to the buffer to write received data to. If the buffer is NULL, then receive data will silently be discarded.
byteCount	The number of bytes to send and receive in this transaction. The value must be a multiple of the number of bytes in each individual transfer.
csReturnToInactive	Configures when the ChipSelect that is used for the transaction should be deasserted and return to being inactive.

See also

csReturnType

Returns

Non-zero on error

23.26.3.11 SetCS()

```
bool DSPIModule::SetCS (
            uint8_t CS )  [inline]
```

Modifies the chip select configuration for the driver context such that only the requested chip select is active during transfers.

Parameters

CS	The chip select number to assert during transfers.

Returns

Whether the active chip select was successfully updated.

23.26.3.12 Start()

```
uint8_t DSPIModule::Start (
            uint8_t * transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )
```

Start begins a SPI transaction using the provided buffers.

Parameters

transmitBufferPtr	A pointer to the buffer containing data to be sent out. If the buffer is NULL, the bus will be held low instead.
receiveBufferPtr	A pointer to the buffer to write received data to. If the buffer is NULL, then receive data will silently be discarded.
byteCount	The number of bytes to send and receive in this transaction. The value must be a multiple of the number of bytes in each individual transfer.
csReturnToInactive	Configures when the ChipSelect that is used for the transaction should be deasserted and return to being inactive.

See also

csReturnType

Returns

Non-zero on error

23.26.3.13 Tx()

```
uint8_t DSPIModule::Tx (
            uint8_t * transmitBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline]
```

Tx begins a transmit only SPI transaction using the provided buffer. Silently discards the received data.

Parameters

transmitBufferPtr	A pointer to the buffer containing data to be sent out. If the buffer is NULL, the bus will be held low instead.
byteCount	The number of bytes to send and receive in this transaction. The value must be a multiple of the number of bytes in each individual transfer.
csReturnToInactive	Configures when the ChipSelect that is used for the transaction should be deasserted and return to being inactive.

See also

csReturnType

Returns

Non-zero on error

23.26.4 Friends And Related Function Documentation

23.26.4.1 DSPIInit

```
uint8_t DSPIInit (
            uint8_t SPIModule,
            uint32_t Baudrate,
            uint8_t QueueBitSize,
            uint8_t CS,
            uint8_t CSPol,
            uint8_t ClkPolarity,
            uint8_t ClkPhase,
            BOOL DoutHiz,
            uint8_t QCD,
            uint8_t DTL )  [friend]
```

Initialize a DSPI module.
Notes:

- The maximum baud rate is CPU_CLOCK / 3
- Will initialize to the highest available baud rate that does not exceed the maximum
- If configured for 8 bits per transfer then the data must be uint8_t aligned

- If configured for greater than 8 bits per transfer then the data must be uint16_t aligned
- If configured for greater than 16 bits per transfer then the data must be uint32_t aligned
- Chip select is based on chipSelectPolarity
- 0 data captured leading edge of DSPI_CLK, changed following edge.
- 1 data changed leading edge of DSPI_CLK, captured following edge.
- 0 default is as close to 1/2 DSPI_CLK without going under, keeping with the interface to SPI
- 0 default is 17/(system clock / 2), in keeping with interface to QSPI

Parameters

SPIModule	SPI module number, 0 - 1
Baudrate	Maximum baud rate requested
QueueBitSize	Number of bits per transfer: 8, 16 or 32
CS	SPI chip selects to use for transfer
CSPol	0 = inactive logic level low, 1 = high
ClkPolarity	0 = inactive logic level low, 1 = high
ClkPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
DoutHiz	Data output high impedance between transfers
QCD	Delay from chip select to valid clock (default is 0). QCD is a value in the QDLYR register and will change the delay between the assertion of the chip select and the start of the DSPI clock. Default setting of one half DSPI clk will be used if parameter is specified as 0x0 or not included.
DTL	DTL is a value in the QDLYR register and will change the delay following a tranfer of a single uint16_t in the DSPI queue. Default reset value of 17/(fsys/2) will be used if parameter is specified 0 or not included.

Returns

Current stat of DSPI bus DSPI state

Notes:

- The maximum baud rate is CPU_CLOCK / 3
- Will initialize to the highest available baud rate that does not exceed the maximum
- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for greater than 8 bits per transfer then the data must be uint16_t aligned
- If configured for greater than 16 bits per transfer then the data must be uint32_t aligned
- Chip select is based on chipSelectPolarity
- 0 data captured leading edge of DSPI_CLK, changed following edge.
- 1 data changed leading edge of DSPI_CLK, captured following edge.
- 0 default is as close to 1/2 DSPI_CLK without going under, keeping with the interface to SPI
- 0 default is 17/(system clock / 2), in keeping with interface to QSPI

Parameters

SPIModule	SPI module number, 0 - 1

Parameters

Baudrate	Maximum baud rate requested
QueueBitSize	Number of bits per transfer: 8, 16 or 32
CS	SPI chip selects to use for transfer
CSPol	0 = inactive logic level low, 1 = high
ClkPolarity	0 = inactive logic level low, 1 = high
ClkPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
DoutHiz	Data output high impedance between transfers
QCD	Delay from chip select to valid clock (default is 0). QCD is a value in the QDLYR register and will change the delay between the assertion of the chip select and the start of the DSPI clock. Default setting of one half DSPI clk will be used if parameter is specified as 0x0 or not included.
DTL	DTL is a value in the QDLYR register and will change the delay following a tranfer of a single uint16_t in the DSPI queue. Default reset value of 17/(fsys/2) will be used if parameter is specified 0 or not included.

Returns

Current stat of DSPI bus DSPI state

Notes:

- The maximum baud rate is CPU_CLOCK / 3
- Will initialize to the highest available baud rate that does not exceed the maximum
- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for greater than 8 bits per transfer then the data must be uint16_t aligned
- If configured for greater than 16 bits per transfer then the data must be uint32_t aligned
- Chip select is based on chipSelectPolarity
- 0 data captured leading edge of DSPI_CLK, changed following edge.
- 1 data changed leading edge of DSPI_CLK, captured following edge.
- 0 default is as close to 1/2 DSPI_CLK without going under, keeping with the interface to SPI
- 0 default is 17/(system clock / 2), in keeping with interface to QSPI

Parameters

SPIModule	SPI module number, 0 - 1
Baudrate	Maximum baud rate requested
QueueBitSize	Number of bits per transfer: 8, 16 or 32
CS	SPI chip selects to use for transfer
CSPol	0 = inactive logic level low, 1 = high
ClkPolarity	0 = inactive logic level low, 1 = high
ClkPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
DoutHiz	Data output high impedance between transfers
QCD	Delay from chip select to valid clock (default is 0)
DTL	Chip select mode dspiChipSelectMode

Returns

Current stat of DSPI bus dspiState

The documentation for this class was generated from the following file:

- coldfire/cpu/MCF5441X/include/dspi.h

23.27 EBI_CS_cfg_t Struct Reference

Configuration structure for an External Bus Interface (EBI) chip select.
`#include <ebi.h>`

Public Attributes

- uint8_t ncs_rd_setup
- uint8_t nrd_setup
- uint8_t ncs_wr_setup
- uint8_t nwe_setup
- uint8_t ncs_rd_pulse
- uint8_t nrd_pulse
- uint8_t ncs_wr_pulse
- uint8_t nwe_pulse
- uint16_t nrd_cycles
- uint16_t nwe_cycles
- uint8_t tdf_cycles
- EBI_CS_BusWidth_t busWidth
- EBI_CS_ByteAccess_t byteAccess
- EBI_CS_NWait_t nWait
- EBI_CS_WrMode_t wrMode
- EBI_CS_RdMode_t rdMode

23.27.1 Detailed Description

Configuration structure for an External Bus Interface (EBI) chip select.
The documentation for this struct was generated from the following file:

- ebi.h

23.28 HtmlPageHandler Class Reference

Base class for all GET handlers. To handle GET requests for a specific URL in your application, build a GET handler object for that specifif URL. A NULL name will be a catch all for all GET requests.
`#include <http.h>`
Inherited by AcmeAuthItem, AcmeObject, CallBackFunctionPageHandler, HtmlConfigExposer, and HtmlPost↩ Handler.

Public Member Functions

- HtmlPageHandler (const char *url, HTTP_RequestTypes rt=tGet, int accessGroup=0, bool Before_↩ Files=false)

  Register handler.
- virtual int ProcessRaw (int sock, HTTP_Request &pd)=0

  This class will do a callback with data for each request to the specified url.
- int GetGroup ()

  Returns access group setting.

Protected Member Functions

- void **InsertSort** (HtmlPageHandler *&ph)

  Insert sort.
- int **SortValue** (HtmlPageHandler *pv)

  Returns the value of the sort compare: -1, 0, 1.

Protected Attributes

- HtmlPageHandler * **m_pNextHandler**

  Pointer to next page handle object.
- const char * **m_pUrlName**

  Pointer to URL. Performs a length match, an empty string matches everything.
- int **m_access_group**

  The access group for this request see CheckHttpAccess.
- HTTP_RequestTypes **m_requestTypes**

  Type of request, HTTP_RequestTypes.

23.28.1 Detailed Description

Base class for all GET handlers. To handle GET requests for a specific URL in your application, build a GET handler object for that specifif URL. A NULL name will be a catch all for all GET requests.

23.28.2 Constructor & Destructor Documentation

23.28.2.1 HtmlPageHandler()

```
HtmlPageHandler::HtmlPageHandler (
            const char * url,
            HTTP_RequestTypes rt = tGet,
            int accessGroup = 0,
            bool Before_Files = false )
```

Register handler.

Parameters

url	The URL name to register. A NULL string matches everything
rt	The GET request type to respond to
accessGroup	Access group of the URL
Before_Files	Specify if the response should be before or after the compiled in HTML files

23.28.3 Member Function Documentation

23.28.3.1 GetGroup()

```
int HtmlPageHandler::GetGroup ( )  [inline]
```

Returns access group setting.

Returns

HTTP_ACCESS

23.28.3.2 ProcessRaw()

```
virtual int HtmlPageHandler::ProcessRaw (
            int sock,
            HTTP_Request & pd )  [pure virtual]
```

This class will do a callback with data for each request to the specified url.

Returns

0 if the request was not processed.

Implemented in CallBackFunctionPageHandler, and CallBackFunctionPostHandler.
The documentation for this class was generated from the following file:

- http.h

23.29 HtmlPostVariableListCallback Class Reference

Implements the HtmlPostVariableListHandler class as a function pointer callback for HTTP POST submissions.
`#include <httppost.h>`
Inherits HtmlPostVariableListHandler.

Public Member Functions

- HtmlPostVariableListCallback (const char *pUrl, postvarhandler *pCallback, int accessGroup=0)

  Custom HTTP POST handler callback function constructor.

23.29.1 Detailed Description

Implements the HtmlPostVariableListHandler class as a function pointer callback for HTTP POST submissions.

23.29.2 Constructor & Destructor Documentation

23.29.2.1 HtmlPostVariableListCallback()

```
HtmlPostVariableListCallback::HtmlPostVariableListCallback (
            const char * pUrl,
            postvarhandler * pCallback,
            int accessGroup = 0 )  [inline]
```

Custom HTTP POST handler callback function constructor.

Parameters

pUrl	Pointer to URL character string
pCallback	Pointer to callback function that will process the POST
accessGroup	Password group access

The documentation for this class was generated from the following file:

- httppost.h

23.30 HTTP_Request Struct Reference

HTTP Request Structure.
`#include <http.h>`

Public Member Functions

- void **ProcessLine** (PSTR startofline)

  internal function
- int **Respond** (int socket)

  internal function
- bool **ExtractAuthentication** (char **pPassword, char **pUser)

  used to extract user name and password in authentication test

Public Attributes

- PSTR **pURL**

  Request URL.
- PSTR **pAuthorization**

  Authorization header if present, otherwise null.
- PSTR **pFirstCookie**

  First cookie if present, otherwise null. More may follow.
- PSTR **pData**

  Pointer to entire data set. Note: not null terminated.
- PSTR **pSep**

  Separator for multipart forms, null if not present.
- PSTR **pHost**

  Pointer to host record null if not present.
- PSTR **last_datarx**

  Pointer to last data read, internal use only.
- PSTR **wsKey**

  Web socket ket, internal use only.
- PSTR **wsProtocol**

  Web socket prototocl, internal use only.
- uint16_t **sep_len**

  Length of separator for multipart forms.
- uint32_t **content_length**

  Content length field from HTML header.
- uint32_t **rx_length**

  Total bytes receive from request, internal use only.
- IPADDR **client_IPaddr**

  IP address of client.
- uint8_t **websocketFlags**

  Web socket flags.
- HTTP_RequestTypes **req**

  Type of request HTTP_RequestTypes.

23.30.1 Detailed Description

HTTP Request Structure.
This structure is populated before a HTTP request is sent out.
The documentation for this struct was generated from the following file:

- http.h

23.31 I2C Class Reference

I2C Peripheral Class.
`#include <bb_i2c.h>`

Public Types

- enum Result_t {
  I2C_RES_ACK , I2C_RES_NACK , I2C_RES_ARB_LST , I2C_RES_BUSY ,
  I2C_RES_ARG }

  Return result types.

Public Member Functions

- I2C (int module)

  Constructor for the I2C peripheral module.
- void setup (uint32_t busSpeed)

  Setup the I2C peripheral interface.
- void resetBus ()

  Reset the I2C bus.
- Result_t DoTransaction (I2CTxn_t *pTransaction, bool bRepeatedStart=false)

  Start an I2C transaction.
- void setNumAddressBytes (uint8_t numAddressBytes=1)

  Specify the register address size for a read or write transaction. A number of bytes, from 0 to 3, can be sent to specify an address up to 24-bit (3 bytes): 0 = none, 1 = 8-bits, 2 = 16-bits, 3 = 24-bits.
- Result_t writeReg8 (uint8_t devAddr, uint32_t reg, uint8_t data)

  Write an 8-bit value to a I2C slave device register.
- Result_t readReg8 (uint8_t devAddr, uint32_t reg, uint8_t &data)

  Read an 8-bit value form an I2C slave device register.
- Result_t writeRegN (uint8_t devAddr, uint32_t reg, const uint8_t *buf, uint32_t blen)

  Write a number of 8-bit values to an I2C slave to the specified register address.
- Result_t readRegN (uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen)

  Read a number of 8-bit values from an I2C slave at the specified register address.

Friends

- class **WireIntf**

23.31.1 Detailed Description

I2C Peripheral Class.
For I2C communication, you can choose the WireIntf class, or the I2C class. The WireIntf class is simpler to use, while the I2C class provides more low level control.
Note that the system automatically instantiates I2C and WireIntf objects for each of the I2C peripheral modules. The WireIntf objects can be accessed with: Wire, Wire1 and Wire2. The I2C objects can be accesses with I2C[0], I2C[1] and I2C[2].
For I2C communication, you can choose the WireIntf class, or the I2C class. The WireIntf class is simpler to use, while the I2C class provides more low level control.
Note that the system automatically instantiates I2C and WireIntf objects for each of the I2C peripheral modules. The WireIntf objects can be accessed with: Wire, Wire1 and Wire2. The I2C objects can be accesses with I2C[0], I2C[1] and I2C[2].

23.31.2 Member Enumeration Documentation

23.31.2.1 Result_t

`enum I2C::Result_t`
Return result types.

Enumerator

I2C_RES_ACK	Acknowledged.
I2C_RES_NACK	Not acknowledged.
I2C_RES_ARB_LST	Arbitration listening.
I2C_RES_BUSY	Bus is busy.
I2C_RES_ARG	Bad argument.

23.31.3 Constructor & Destructor Documentation

23.31.3.1 I2C()

```
I2C::I2C (
            int module )
```

Constructor for the I2C peripheral module.

Parameters

module	The I2C module to instantiate (0, 1 or 2)

23.31.4 Member Function Documentation

23.31.4.1 DoTransaction()

```
Result_t I2C::DoTransaction (
            I2CTxn_t * pTransaction,
            bool bRepeatedStart = false )
```

Start an I2C transaction.
Execute the transaction pointed to by pTransaction

Parameters

∗pTransaction	Pointer to the transaction to execute
bRepeatedStart	Repeat the I2C start sequence

Returns

Result_t

23.31.4.2 readReg8()

```
Result_t I2C::readReg8 (
            uint8_t devAddr,
            uint32_t reg,
            uint8_t & data )
```

Read an 8-bit value form an I2C slave device register.

Parameters

devAddr	Address of I2C device

Parameters

reg	Register address to read
data	Reference to variable in which to store the register data

Returns

Result_t

23.31.4.3 readRegN()

```
Result_t I2C::readRegN (
            uint8_t devAddr,
            uint32_t reg,
            uint8_t * buf,
            uint32_t blen )
```

Read a number of 8-bit values from an I2C slave at the specified register address.

Parameters

devAddr	Address of I2C device
reg	Register address to read
∗buf	Pointer to buffer to store received data
blen	Number of bytes to read

Returns

Result_t

23.31.4.4 resetBus()

```
void I2C::resetBus ( )
```

Reset the I2C bus.

23.31.4.5 setNumAddressBytes()

```
void I2C::setNumAddressBytes (
            uint8_t numAddressBytes = 1 )  [inline]
```

Specify the register address size for a read or write transaction. A number of bytes, from 0 to 3, can be sent to specify an address up to 24-bit (3 bytes): 0 = none, 1 = 8-bits, 2 = 16-bits, 3 = 24-bits.
If you are using a single I2C device, or all devices use the same address size, this function only needs to be called once. If you have multiple I2C devices with different address/register bit sizes, then this function should be called before each read/write operation for each different device.

Parameters

numAddressBytes	The number of address bytes to send: 0 - 3. The default value is 1 byte.

23.31.4.6 setup()

```
void I2C::setup (
```

```
            uint32_t busSpeed )
```

Setup the I2C peripheral interface.

Parameters

busSpeed	Target bus speed. It the specified speed cannot be achieved, a lower bus speed may be enabled.

23.31.4.7 writeReg8()

```
Result_t I2C::writeReg8 (
            uint8_t devAddr,
            uint32_t reg,
            uint8_t data )
```

Write an 8-bit value to a I2C slave device register.

Parameters

devAddr	Address of I2C device
reg	Register address to write
data	Data to write to register

Returns

Result_t

23.31.4.8 writeRegN()

```
Result_t I2C::writeRegN (
            uint8_t devAddr,
            uint32_t reg,
            const uint8_t * buf,
            uint32_t blen )
```

Write a number of 8-bit values to an I2C slave to the specified register address.
Executing this function will send the following: a start bit, the slave device address and read/write bit, the 8-bit data bytes, and finally a stop bit. Note that the address is not incremented with each 8-bit data value.

Parameters

devAddr	Address of I2C device
reg	Register address to write
∗buf	Pointer to buffer containing data to write
blen	Number of bytes to write

Returns

Result_t

The documentation for this class was generated from the following file:

- i2c.h

23.32 I2CDevice Class Reference

I2C Device Class (recommended)

```
#include <i2c.h>
```

Public Member Functions

- I2CDevice (I2C &pInterface, uint8_t deviceAddress, uint8_t numAddressBytes=1)

  Initialize the I2C module.
- uint8_t getI2CAddress ()

  Get device's I2C address.
- void setup (uint32_t busSpeed)

  Setup the I2C peripheral module.
- void resetBus ()

  Reset the I2C bus.
- I2C::Result_t writeReg8 (uint32_t reg, uint8_t data)

  Write an 8-bit value to an I2C slave device register.
- I2C::Result_t readReg8 (uint32_t reg, uint8_t &data)

  Read an 8-bit value form an I2C slave device register.
- I2C::Result_t writeRegN (uint32_t reg, const uint8_t *buf, uint32_t blen)

  Write a number of 8-bit values to an I2C slave to the specified register address.
- I2C::Result_t readRegN (uint32_t reg, uint8_t *buf, uint32_t blen)

  Read a number of 8-bit values from an I2C slave at the specified register address.

23.32.1 Detailed Description

I2C Device Class (recommended)
For I2C communication, you can choose the I2CDevice class, the I2C class, or the WireIntf class. The recommended interface is the I2CDevice class.
For example, to create an object for an analog to digital converter:

```
I2CDevice AdcDevice(i2c[I2C_MODULE_NUM], I2C_ADC_ADDRESS);
```

To read a 16 bit value:

```
uint8_t I2CStat = AdcDevice.readRegN(0x00, buf, 2);
```

23.32.2 Constructor & Destructor Documentation

23.32.2.1 I2CDevice()

```
I2CDevice::I2CDevice (
            I2C & pInterface,
            uint8_t deviceAddress,
            uint8_t numAddressBytes = 1 )  [inline]
```

Initialize the I2C module.

Parameters

pInterface	Pointer to the I2C interface to use. This will normally be one of the preallocated I2C modules in the i2c[] array: i2c[0] (default), i2c[1] or i2c[2].
deviceAddress	I2C device address
numAddressBytes	Number of address bytes to send to specify the register or address to access for a read or write operation. Refer to your I2C device data sheet to determine the correct number. The I2C address range is 24-bits maximum (3 bytes). The default value is 1 byte, representing an address range of 0 to 255.

23.32.3 Member Function Documentation

23.32.3.1 getI2CAddress()

```
uint8_t I2CDevice::getI2CAddress ( )  [inline]
```

Get device's I2C address.

Note that the return address will be just the I2C address bits, not including the read/write bit. If your device data sheet specifies the address byte to include the read/write bit, that value would be this address left shifted by 1. For example, if a device data sheet specifies the address as 0xA2 and includes the read/write bit, the actual address would be 0x51.

Returns

The device's I2C address

23.32.3.2 readReg8()

```
I2C::Result_t I2CDevice::readReg8 (
            uint32_t reg,
            uint8_t & data )  [inline]
```

Read an 8-bit value form an I2C slave device register.

Parameters

reg	Register address to read
data	Reference to variable in which to store the register data

Returns

I2C::Result_t

23.32.3.3 readRegN()

```
I2C::Result_t I2CDevice::readRegN (
            uint32_t reg,
            uint8_t * buf,
            uint32_t blen )  [inline]
```

Read a number of 8-bit values from an I2C slave at the specified register address.

Parameters

reg	Register address to read
**buf*	Pointer to buffer to store received data
blen	Number of bytes to read

Returns

I2C::Result_t

23.32.3.4 resetBus()

```
void I2CDevice::resetBus ( )  [inline]
```

Reset the I2C bus.

23.32.3.5 setup()

```
void I2CDevice::setup (
            uint32_t busSpeed )  [inline]
```

Setup the I2C peripheral module.

Parameters

busSpeed	Target bus speed. It the specified speed cannot be achieved, a lower bus speed may be enabled.

23.32.3.6 writeReg8()

```
I2C::Result_t I2CDevice::writeReg8 (
            uint32_t reg,
            uint8_t data )  [inline]
```

Write an 8-bit value to an I2C slave device register.

Parameters

reg	Register address to write
data	Data to write to register

Returns

I2C::Result_t

23.32.3.7 writeRegN()

```
I2C::Result_t I2CDevice::writeRegN (
            uint32_t reg,
            const uint8_t * buf,
            uint32_t blen )  [inline]
```

Write a number of 8-bit values to an I2C slave to the specified register address.
Executing this function will send the following: a start bit, the slave device address and read/write bit, the 8-bit data bytes, and finally a stop bit. Note that the address is not incremented with each 8-bit data value.

Parameters

reg	Register address to write
∗buf	Pointer to buffer containing data to write
blen	Number of bytes to write

Returns

I2C::Result_t

The documentation for this class was generated from the following file:

- i2c.h

23.33 InterfaceBlock Class Reference

Network interface configuration block class for interface control and configuration.
`#include <netinterface.h>`
Inherits config_obj.
Inherited by EtherLikeInterface, MultiHomeInterface, and PPPInterface.

Public Member Functions

- InterfaceBlock (const char *name, const char *desc=NULL)

  Constructor with interface name and description.
- InterfaceBlock (config_obj &owner, const char *name, const char *desc=NULL)

  Constructor with config_obj owner, interface name and description.
- void RegisterInterface ()

  Register a network interface with system.
- virtual void EnableMulticast (MACADR macAddress, BOOL addAddress)=0

  Enable Multicast on this interface.
- virtual bool LinkActive ()=0

  Returns the link status of the network interface.
- virtual int LinkSpeed ()=0

  Returns the link speed of the network interface.
- virtual bool LinkDuplex ()=0

  Returns the full or half link duplex of the network interface.
- void InterfaceLinkChange (bool link)

  Change the link status of the network interface.
- virtual const char * GetInterfaceName ()

  Returns the interface name network interface.
- int GetInterfaceNumber ()

  Returns the Interface Number of the network interface.
- bool IsRootInterface ()

  Returns true if this is the first interface in the interface list.

Public Member Functions inherited from config_obj

- config_obj (config_obj &owner, const char *name, const char *desc)

  Object constructor with the parent/owner leaf parameter.
- config_obj (const char *name, const char *desc)

  Object constructor.
- virtual void GetTextValue (NBString &s)

  Get the object value as a text string to the specified NBString object.
- virtual void GetTypeValue (NBString &s)

  Assigns the object type value to the specified NBString object.

23.33.1 Detailed Description

Network interface configuration block class for interface control and configuration.

- Network Interfaces are configured and controlled C++ InterfaceBlock objects. The InterfaceBlock contains variables and member functions for things such as:
- The Configuration Server interface
- IP settings for both IPv4 and IPv6 types
- Configuration to static or DHCP, and DHCP status (both v4 and v6)

- Multihome
- Routing
- AutoIP

Note

While the recommended method for getting and setting network interface parameters is to obtain a pointer to an InterfaceBlock, there are also non-member functions of the object that can be used for situations that are best served by a single function call, such as just getting a single IP address of an interface. When using those functions pay special attention to the return value type. While IPADDR should be used because it supports both IPADDR4 and IPADDR6 types automatically, some of these helper functions may specifically return an IPADDR4 type.

23.33.2 Constructor & Destructor Documentation

23.33.2.1 InterfaceBlock() [1/2]

```
InterfaceBlock::InterfaceBlock (
            const char * name,
            const char * desc = NULL )
```

Constructor with interface name and description.

Parameters

name	Interface name.
desc	Interface description. If not used, default is null.

23.33.2.2 InterfaceBlock() [2/2]

```
InterfaceBlock::InterfaceBlock (
            config_obj & owner,
            const char * name,
            const char * desc = NULL )
```

Constructor with config_obj owner, interface name and description.

Parameters

owner	Reference to config_obj owner of the interface
name	Interface name.
desc	Interface description. If not used, default is null.

23.33.3 Member Function Documentation

23.33.3.1 EnableMulticast()

```
virtual void InterfaceBlock::EnableMulticast (
            MACADR macAddress,
            BOOL addAddress )  [pure virtual]
```

Enable Multicast on this interface.
Calls registered interface multicast routine.

Parameters

macAddress	Multicast MAC address.
addAddress	True to add address/interface, false to remove it

See also

EnableMulticast(MACADR macAddress, int interface), DisableMulticast()

23.33.3.2 GetInterfaceName()

```
virtual const char * InterfaceBlock::GetInterfaceName ( )  [virtual]
```

Returns the interface name network interface.

Returns

Pointer to a character string containing the name of the network interface

See also

InterfaceName()

23.33.3.3 GetInterfaceNumber()

```
int InterfaceBlock::GetInterfaceNumber ( )  [inline]
```

Returns the Interface Number of the network interface.

Returns

Interface number of the InterfaceBlock.

See also

GetInterfaceNumber(InterfaceBlock *pifb)

23.33.3.4 InterfaceLinkChange()

```
void InterfaceBlock::InterfaceLinkChange (
            bool link )
```

Change the link status of the network interface.

Parameters

link	Set link active status to true or false

23.33.3.5 IsRootInterface()

```
bool InterfaceBlock::IsRootInterface ( )  [inline]
```

Returns true if this is the first interface in the interface list.

Returns

True if this is the first interface in the interface list, otherwise false

23.33.3.6 LinkActive()

```
virtual bool InterfaceBlock::LinkActive ( )  [pure virtual]
```
Returns the link status of the network interface.

Returns

True if link is active, otherwise false

See also

GetInterfaceLink(), InterfaceLinkActive()

23.33.3.7 LinkDuplex()

```
virtual bool InterfaceBlock::LinkDuplex ( )  [pure virtual]
```
Returns the full or half link duplex of the network interface.

Returns

True if full duplex, false if half duplex

See also

InterfaceLinkDuplex()

23.33.3.8 LinkSpeed()

```
virtual int InterfaceBlock::LinkSpeed ( )  [pure virtual]
```
Returns the link speed of the network interface.

Returns

10 or 100 representing the Mbps speed

See also

InterfaceLinkSpeed()

23.33.3.9 RegisterInterface()

```
void InterfaceBlock::RegisterInterface ( )
```
Register a network interface with system.
InterfaceBlock member function to register a new network interface

See also

Removeinterface(), EnableMulticast(), DisableMulticast()

The documentation for this class was generated from the following file:

- netinterface.h

23.34 IPADDR4 Class Reference

Used to store and manipulate IPv4 addresses in dual stack mode.
`#include <nettypes.h>`

Public Member Functions

- IPADDR4 ()=default

  Constructor to create an IPv4 object initialized to null.
- IPADDR4 (const IPADDR4 &v)=default

  Constructor to create an IPv4 object initialized to to the specified IPADDR4 IP address.
- bool IsNull () const

  Check if the IP address is null.
- bool NotNull () const

  Check if the IP address is not null.
- void SetNull ()

  Set the IP address to null.
- bool IsLoopBack () const

  Check if the IP address is the loopback address for the interface.
- bool IsMultiCast () const

  Check if the IPADDR4 object contains a Multicast IP address the interface.
- bool IsmDns ()

  Check if the IPADDR4 object contains a mDNS IP address the interface.
- bool IsGlobalBroadCast () const

  Check if the IPADDR4 object contains a global broadcast address: 255.255.255.255.
- bool IsAutoIP ()

  Check if the IPADDR4 object contains an AutoIP address.
- void print () const

  Print an IPv4 address to the stdout serial port.
- void fdprint (int fd) const

  Print an IPv4 address to the specified file descriptor.
- int sprintf (char *cp, int maxl) const

  sprintf an IPv4 address to the specified buffer
- void SetFromAscii (const char *cp)

  Set the IPv4 address from a character string.

Static Public Member Functions

- static IPADDR4 NullIP ()

  C++ static function for a null IP address.
- static IPADDR4 GlobalBroadCast ()

  C++ static function for a global broadcast IP address.

23.34.1 Detailed Description

Used to store and manipulate IPv4 addresses in dual stack mode.
It is recommended to use the IPADDR object, which can hold either an IPv4 or IPv6 address, rather than explicitly specifying IPv4.

23.34.2 Constructor & Destructor Documentation

23.34.2.1 IPADDR4() [1/2]

```
IPADDR4::IPADDR4 ( )  [default]
```

Constructor to create an IPv4 object initialized to null.

23.34.2.2 IPADDR4() [2/2]

```
IPADDR4::IPADDR4 (
            const IPADDR4 & v )  [default]
```

Constructor to create an IPv4 object initialized to to the specified IPADDR4 IP address.

Parameters

v	IPADDR4 object used for initialization.

23.34.3 Member Function Documentation

23.34.3.1 fdprint()

```
void IPADDR4::fdprint (
            int fd ) const
```

Print an IPv4 address to the specified file descriptor.

Parameters

fd	File descriptor used to send IP address

23.34.3.2 GlobalBroadCast()

```
static IPADDR4 IPADDR4::GlobalBroadCast ( )  [inline], [static]
```

C++ static function for a global broadcast IP address.

23.34.3.3 IsAutoIP()

```
bool IPADDR4::IsAutoIP ( )  [inline]
```

Check if the IPADDR4 object contains an AutoIP address.

Return values

true	if the IP address value is an AutoIP address

See also

IsLoopBack(), IsLinkLocal()

23.34.3.4 IsGlobalBroadCast()

```
bool IPADDR4::IsGlobalBroadCast ( ) const  [inline]
```

Check if the IPADDR4 object contains a global broadcast address: 255.255.255.255.

Return values

true	if the IP address value is a global broadcast

See also

IsLoopBack(), IsLinkLocal()

23.34.3.5 IsLoopBack()

`bool IPADDR4::IsLoopBack ( ) const [inline]`
Check if the IP address is the loopback address for the interface.

Return values

true	if the IP address value is the loopback address

See also

IsMultiCast(), IsLinkLocal()

23.34.3.6 IsmDns()

`bool IPADDR4::IsmDns ( ) [inline]`
Check if the IPADDR4 object contains a mDNS IP address the interface.

Return values

true	if the IP address value is a mDNS address

See also

IsLoopBack(), IsLinkLocal()

23.34.3.7 IsMultiCast()

`bool IPADDR4::IsMultiCast ( ) const [inline]`
Check if the IPADDR4 object contains a Multicast IP address the interface.

Return values

true	if the IP address value is a Multicast address

See also

IsLoopBack(), IsLinkLocal()

23.34.3.8 IsNull()

`bool IPADDR4::IsNull ( ) const [inline]`
Check if the IP address is null.

Return values

true	if the IP address value is null

See also

NotNull(), SetNull()

23.34.3.9 NotNull()

```
bool IPADDR4::NotNull ( ) const  [inline]
```

Check if the IP address is not null.

Return values

true	if the IP address value is not null

See also

IsNull(), SetNull()

23.34.3.10 NullIP()

```
static IPADDR4 IPADDR4::NullIP ( )  [inline], [static]
```

C++ static function for a null IP address.

23.34.3.11 print()

```
void IPADDR4::print ( ) const
```

Print an IPv4 address to the stdout serial port.

23.34.3.12 SetFromAscii()

```
void IPADDR4::SetFromAscii (
              const char * cp )
```

Set the IPv4 address from a character string.

Parameters

cp	Character string. For example, "192.168.1.10"

23.34.3.13 SetNull()

```
void IPADDR4::SetNull ( )  [inline]
```

Set the IP address to null.

See also

IsNull(), NotNull()

23.34.3.14 sprintf()

```
int IPADDR4::sprintf (
              char * cp,
              int maxl ) const
```

sprintf an IPv4 address to the specified buffer

Parameters

cp	Destination character buffer
maxl	Maximum number of characters

Returns

Number of characters

The documentation for this class was generated from the following file:

- nettypes.h

23.35 IPADDR6 Class Reference

Used to hold and manipulate IPv4 and IPv6 addresses in dual stack mode.
`#include <ipv6_addr.h>`

Public Member Functions

- bool IsEmbeddedIPV4 () const

  An IPADDR6 object can store a IPv4 or IPv6 address. This function returns true if the instance contains an IPv4 address.
- IPADDR4 Extract4 () const

  Extracts an IPv4 address from the object.
- bool IsNull () const

  Check if the IP address is null.
- bool NotNull () const

  Check if the IP address is not null.
- bool IsLoopBack () const

  Check if the IP address is the loopback address for the interface.
- bool IsMultiCast () const

  Check if the IPADDR6 object contains a Multicast IP address the interface.
- bool IsLinkLocal () const

  Check if the IP address is the link-local address for the interface.
- MACADR McastMac () const

  Return the MAC address used for Multicasts for the interface.
- void print (bool bCompact=true, bool bShowV4Raw=false) const

  Print the IP address value to stdout.
- void fdprint (int fd, bool bCompact=true, bool bShowV4Raw=false) const

  Print the IP address to the specified file descriptor.
- int sprintf (char *cp, int maxl, bool bCompact=true, bool bShowV4Raw=false) const

  Print the IP address to the specified buffer.
- void SetFromAscii (const char *cp, bool bembed_v4addresses=true)

  Set the IP address value of an IPADDR6 object.
- void SetFromIP4 (IPADDR4 ip)

  Set the IP address value of an IPADDR6 object from an IPADD4 object.
- void SetFromUint32Shortcut (uint32_t w0, uint32_t w1, uint32_t w2, uint32_t w3)

  Set the IP address value of an IPADDR6 object from 4 discrete uint32_t values.
- void SetNull ()

  Set the IP address value of an IPADDR6 object to null.

Static Public Member Functions

- static IPADDR6 AsciiToIp6 (const char *cp, bool bembed_v4addresses=true)

  Static function to return an IPADDR6 object created from an ASCII value IPv4 or IPv6 address.
- static IPADDR6 NullIP ()

  Static function to return a null IPADDR6 object.

23.35.1 Detailed Description

Used to hold and manipulate IPv4 and IPv6 addresses in dual stack mode.

Note

It is recommended to use the IPADDR object, which can hold either an IPv4 or IPv6 address, rather than explicitly specifying IPv6.

23.35.2 Member Function Documentation

23.35.2.1 AsciiToIp6()

```
static IPADDR6 IPADDR6::AsciiToIp6 (
            const char * cp,
            bool bembed_v4addresses = true )  [static]
```

Static function to return an IPADDR6 object created from an ASCII value IPv4 or IPv6 address.

Parameters

**cp*	Pointer to the ASCII string representing an IPv4 or IPv6 address
bembed_v4addresses	If false function will only process an IPv6 address

Return values

IPADDR6	object

See also

NullIP()

23.35.2.2 Extract4()

```
IPADDR4 IPADDR6::Extract4 ( ) const  [inline]
```

Extracts an IPv4 address from the object.

Return values

IPADDR4	Value of the IPv4 address

23.35.2.3 fdprint()

```
void IPADDR6::fdprint (
            int fd,
            bool bCompact = true,
```

```
bool bShowV4Raw = false ) const
```

Print the IP address to the specified file descriptor.

Parameters

fd	Valid file descriptor to send the data
bCompact	Display IPv6 address in compact notation
bShowV4Raw	Normally set to false. If set to true, will display the IPv4 address 128-bits of the IP address object in the raw format "::FFFF:xx.xx.xx.xx"

See also

print(), sprintf()

23.35.2.4 IsEmbeddedIPV4()

```
bool IPADDR6::IsEmbeddedIPV4 ( ) const [inline]
```

An IPADDR6 object can store a IPv4 or IPv6 address. This function returns true if the instance contains an IPv4 address.
Internally, an IPv4 address is stored in the format: "::FFFF:xx.xx.xx.xx"

Return values

true	If the IPADDR6 object contains and IPv4 address
false	Otherwise

23.35.2.5 IsLinkLocal()

```
bool IPADDR6::IsLinkLocal ( ) const [inline]
```

Check if the IP address is the link-local address for the interface.
Link local is
`FE80::/10`
. Masks the top 10 bits and check.

Return values

true	if the IP address value is the link-local address

See also

IsLoopBack(), IsMultiCast()

23.35.2.6 IsLoopBack()

```
bool IPADDR6::IsLoopBack ( ) const [inline]
```

Check if the IP address is the loopback address for the interface.

Return values

true	if the IP address value is the loopback address

See also

IsMultiCast(), IsLinkLocal()

23.35.2.7 IsMultiCast()

`bool IPADDR6::IsMultiCast ( ) const [inline]`
Check if the IPADDR6 object contains a Multicast IP address the interface.

Return values

true	if the IP address value is a Multicast address

See also

IsLoopBack(), IsLinkLocal()

23.35.2.8 IsNull()

`bool IPADDR6::IsNull ( ) const [inline]`
Check if the IP address is null.

Return values

true	if the IP address value is null

See also

NotNull()

23.35.2.9 McastMac()

`MACADR IPADDR6::McastMac ( ) const`
Return the MAC address used for Multicasts for the interface.

Return values

MACADR	if a Multicast address exists
null	otherwise

23.35.2.10 NotNull()

`bool IPADDR6::NotNull ( ) const [inline]`
Check if the IP address is not null.

Return values

true	if the IP address value is not null

See also

IsNull()

23.35.2.11 NullIP()

```
static IPADDR6 IPADDR6::NullIP ( )  [static]
```

Static function to return a null IPADDR6 object.

Static functions can be called without a specific C++ object. For example,

```
if ( myIPAddress == IPADDR6::NullIP() )
 iprintf("myIPAddress is null\r\n");
else
 iprintf("myIPAddress = %I\r\n", myIPAddress);
```

IPADDR6::NullIP() can be used on IPADDR, IPADDR4 and IPADDR6 objects.

Return values

null	IPADDR6 object

See also

AsciiToIp6()

23.35.2.12 print()

```
void IPADDR6::print (
            bool bCompact = true,
            bool bShowV4Raw = false ) const
```

Print the IP address value to stdout.

Parameters

bCompact	Display IPv6 address in compact notation
bShowV4Raw	Normally set to false. if set to true, will display the IPv4 address 128-bits of the IP address object in the raw format "::FFFF:xx.xx.xx.xx"

See also

fdprint(), sprintf()

23.35.2.13 SetFromAscii()

```
void IPADDR6::SetFromAscii (
            const char * cp,
            bool bembed_v4addresses = true )
```

Set the IP address value of an IPADDR6 object.

Parameters

**cp*	Pointer to the ASCII string representing an IPv4 or IPv6 address
bembed_v4addresses	If false function will only process an IPv6 address

See also

SetFromIP4(), SetNull()

23.35.2.14 SetFromIP4()

```
void IPADDR6::SetFromIP4 (
            IPADDR4 ip )
```

Set the IP address value of an IPADDR6 object from an IPADD4 object.

Parameters

ip	IPADDR4 object

See also

SetFromAscii(), SetNull()

23.35.2.15 SetFromUint32Shortcut()

```
void IPADDR6::SetFromUint32Shortcut (
            uint32_t w0,
            uint32_t w1,
            uint32_t w2,
            uint32_t w3 )  [inline]
```

Set the IP address value of an IPADDR6 object from 4 discrete uint32_t values.

Parameters

w0	Unsigned 32-bit integer
w1	Unsigned 32-bit integer
w2	Unsigned 32-bit integer
w3	Unsigned 32-bit integer

See also

SetFromAscii(), SetFromIP4(), SetNull()

23.35.2.16 SetNull()

```
void IPADDR6::SetNull ( )  [inline]
```

Set the IP address value of an IPADDR6 object to null.

See also

SetFromAscii(), SetFromIP4()

23.35.2.17 sprintf()

```
int IPADDR6::sprintf (
            char * cp,
            int maxl,
```

```
            bool bCompact = true,
            bool bShowV4Raw = false ) const
```

Print the IP address to the specified buffer.

Parameters

**cp*	Pointer to the destination buffer to store the data
maxl	Maximum number of bytes to write to buffer
bCompact	Display IPv6 address in compact notation
bShowV4Raw	Normally set to false. if set to true, will display the IPv4 address 128-bits of the IP address object in the raw format "::FFFF:xx.xx.xx.xx"

Returns

Number of bytes copied to buffer

See also

print(), fdprint()

The documentation for this class was generated from the following file:

- ipv6_addr.h

23.36 JsonAllocString Struct Reference

A list of large strings that are created with malloc.
`#include <json_lexer.h>`

Public Attributes

- JsonAllocString * **pNext**

  A pointer to the next string.
- char **data** [ ]

  The current string data.

23.36.1 Detailed Description

A list of large strings that are created with malloc.
The documentation for this struct was generated from the following file:

- json_lexer.h

23.37 JsonRef Class Reference

Represents a positional reference (pointer) of a location inside a ParsedJsonDataSet object

`#include <json_lexer.h>`

Public Member Functions

Querying the Data Structure

Allows easy retrieval of entities by array index or object key name

Example

```
printf("User %d name %s",
  (int)myJsonRef("users")[0]("id"),
  (const char *)myJsonRef("users")[0]("name")
);
```

- JsonRef operator[ ] (int i)

  Get a JsonRef from an array by numerical index.
- JsonRef object (const char *name)

  Get a JsonRef representing an object from a parent object by key name.
- JsonRef name (const char *name)

  Get a JsonRef representing any entity from a parent object by key name.
- JsonRef operator() (const char *name)

  Get a JsonRef representing any entity from a parent object by key name.

Types and Operators

Allows casting of a JsonRef to the desired type

Example

```
bool b = (bool)myJsonRef("user")("isEnabled");
int i = (int)myJsonRef("user")("id");
const char *str = (const char *)myJsonRef("user")("name");
```

- operator bool () const

  Operator to return a JSON element of type boolean.
- operator float () const

  Operator to return a JSON element of type float.
- operator double () const

  Operator to return a JSON element of type double.
- operator uint8_t () const

  Operator to return a JSON element of type uint8_t.
- operator int () const

  Operator to return a JSON element of type int.
- operator uint16_t () const

  Operator to return a JSON element of type uint16_t.
- operator uint32_t () const

  Operator to return a JSON element of type uint32_t.
- operator int8_t () const

  Operator to return a JSON element of type int8_t.
- operator int16_t () const

  Operator to return a JSON element of type int16_t.
- operator int32_t () const

  Operator to return a JSON element of type int32_t.
- operator time_t () const

  Operator to return a JSON element of type time_t.
- operator const char * () const

  *Operator to return a JSON element of type const char *.*
- operator NBString () const

  Operator to return a JSON element of type NBString.

Checking Validity

- bool IsNumber ()

  Check if the current JsonRef is a valid JSON number element.
- bool IsObject ()

  Check if the current JsonRef is a valid JSON object element.
- bool IsString ()

  Check if the current JsonRef is a valid JSON string element.
- bool IsBool ()

  Check if the current JsonRef is a valid JSON boolean element.
- bool IsNull ()

Check if the current JsonRef is a valid JSON null element.

- bool IsArray ()

  Check if the current JsonRef is a valid JSON array element.

- bool Valid () const

  Check if the current JsonRef is a valid JSON position.

Traversing the Data Structure

- JsonRef start ()

  Begin traversal: check for ref validity and reset the position.

- JsonRef next () const

  Traverse: check for ref validity and increment the position.

- bool IncPosition ()

  Increment the position index of the ParsedJsonDataSet.

- bool JumpPosition (int siz)

  Jump to a specific position index in the ParsedJsonDataSet.

- void ResetPosition ()

  Reset the position index of the ParsedJsonDataSet to the beginning.

- void SkipElement ()

  Increment the position index of the ParsedJsonDataSet beyond this element.

- void SkipArray ()

  Increment the position index of the ParsedJsonDataSet beyond this array.

- void SkipObject ()

  Increment the position index of the ParsedJsonDataSet beyond this object.

Utility

- int PrintChildren (bool pretty=false)

  Output child objects of this JsonRef as JSON text to stdout.

- int PrintChildrenToFd (int fd, bool pretty=false)

  Output child objects of this JsonRef as JSON text to a file descriptor.

- int PrintChildrenToBuffer (char *buffer, int maxlen, bool pretty=false)

  Output child objects of this JsonRef as JSON text to a buffer pointer.

- int GetChildPrintSize (bool pretty=false)

  Return the length of the child objects' JSON text as written with the PrintChildren functions.*

- void DiagDump (const char *lab)

  Output a verbose diagnostic report of the ParsedJsonDatSet to stdout.

- void MakeInvalid ()

Friends

- class **ParsedJsonDataSet**

23.37.1 Detailed Description

Represents a positional reference (pointer) of a location inside a ParsedJsonDataSet object

A JsonRef can be used as a pointer to a variable or sub object inside a ParsedJsonDataSet object to make decoding the hierarchy much simpler and easier to understand. A JsonRef is not limited to just pointing at internal objects; it can point to any JSON parsed token/type.

23.37.2 Member Function Documentation

23.37.2.1 DiagDump()

```
void JsonRef::DiagDump (
            const char * lab )
```

Output a verbose diagnostic report of the ParsedJsonDatSet to stdout.

23.37.2.2 GetChildPrintSize()

```
int JsonRef::GetChildPrintSize (
            bool pretty = false )
```

Return the length of the child objects' JSON text as written with the PrintChildren∗ functions.

Parameters

pretty	Whether indentation and new lines should be provided between elements.

Returns

The number of characters needed to print the JSON data set.

23.37.2.3 IncPosition()

```
bool JsonRef::IncPosition ( )
```

Increment the position index of the ParsedJsonDataSet.

23.37.2.4 IsArray()

```
bool JsonRef::IsArray ( )  [inline]
```

Check if the current JsonRef is a valid JSON array element.

Returns

True if parsed JSON element is an array, otherwise false.

23.37.2.5 IsBool()

```
bool JsonRef::IsBool ( )  [inline]
```

Check if the current JsonRef is a valid JSON boolean element.

Returns

True if parsed JSON element is a boolean, otherwise false.

23.37.2.6 IsNull()

```
bool JsonRef::IsNull ( )  [inline]
```

Check if the current JsonRef is a valid JSON null element.

Returns

True if parsed JSON element is a null, otherwise false.

23.37.2.7 IsNumber()

```
bool JsonRef::IsNumber ( )  [inline]
```

Check if the current JsonRef is a valid JSON number element.

Returns

True if parsed JSON element is a number, otherwise false.

23.37.2.8 IsObject()

```
bool JsonRef::IsObject ( )  [inline]
```

Check if the current JsonRef is a valid JSON object element.

Returns

True if parsed JSON element is an object, otherwise false.

23.37.2.9 IsString()

```
bool JsonRef::IsString ( )  [inline]
```

Check if the current JsonRef is a valid JSON string element.

Returns

True if parsed JSON element is a string, otherwise false. A string can be of type char *, or a NBString object.

23.37.2.10 JumpPosition()

```
bool JsonRef::JumpPosition (
            int siz )
```

Jump to a specific position index in the ParsedJsonDataSet.

23.37.2.11 MakeInvalid()

```
void JsonRef::MakeInvalid ( )  [inline]
```

Todo Document

23.37.2.12 name()

```
JsonRef JsonRef::name (
            const char * name )  [inline]
```

Get a JsonRef representing any entity from a parent object by key name.
Useful if more than one JSON object/element has the same name.

Example

```
JsonRef userThree = myJsonRef.name("users")[3];
```

Parameters

*name	Pointer to name of the entity to find

Returns

The reference to the entity of type JsonRef

See also

JsonRef::operator()(), JsonRef::object()

23.37.2.13 next()

```
JsonRef JsonRef::next ( ) const  [inline]
```

Traverse: check for ref validity and increment the position.

23.37.2.14 object()

```
JsonRef JsonRef::object (
            const char * name )  [inline]
```

Get a JsonRef representing an object from a parent object by key name.

Example

```
JsonRef john = myJsonRef.object("users").object("john");
```

Parameters

**name*	Pointer to name of the object to find

Returns

The reference to the object of type JsonRef

23.37.2.15 operator bool()

```
JsonRef::operator bool ( ) const  [inline]
```

Operator to return a JSON element of type boolean.

Returns

Boolean value

23.37.2.16 operator const char *()

```
JsonRef::operator const char * ( ) const  [inline]
```

Operator to return a JSON element of type const char *.

Returns

const char * value

23.37.2.17 operator double()

```
JsonRef::operator double ( ) const  [inline]
```

Operator to return a JSON element of type double.

Returns

double value

23.37.2.18 operator float()

```
JsonRef::operator float ( ) const  [inline]
```

Operator to return a JSON element of type float.

Returns

float value

23.37.2.19 operator int()

`JsonRef::operator int ( ) const [inline]`
Operator to return a JSON element of type int.

Returns

integer value

23.37.2.20 operator int16_t()

`JsonRef::operator int16_t ( ) const [inline]`
Operator to return a JSON element of type int16_t.

Returns

int16_t value

23.37.2.21 operator int32_t()

`JsonRef::operator int32_t ( ) const [inline]`
Operator to return a JSON element of type int32_t.

Returns

int32_t value

23.37.2.22 operator int8_t()

`JsonRef::operator int8_t ( ) const [inline]`
Operator to return a JSON element of type int8_t.

Returns

int8_t value

23.37.2.23 operator NBString()

`JsonRef::operator NBString ( ) const [inline]`
Operator to return a JSON element of type NBString.

Returns

NBString object

23.37.2.24 operator time_t()

`JsonRef::operator time_t ( ) const [inline]`
Operator to return a JSON element of type time_t.

Returns

time_t value

23.37.2.25 operator uint16_t()

```
JsonRef::operator uint16_t ( ) const  [inline]
```

Operator to return a JSON element of type uint16_t.

Returns

uint16_t value

23.37.2.26 operator uint32_t()

```
JsonRef::operator uint32_t ( ) const  [inline]
```

Operator to return a JSON element of type uint32_t.

Returns

uint32_t value

23.37.2.27 operator uint8_t()

```
JsonRef::operator uint8_t ( ) const  [inline]
```

Operator to return a JSON element of type uint8_t.

Returns

uint8_t value

23.37.2.28 operator()()

```
JsonRef JsonRef::operator() (
            const char * name )  [inline]
```

Get a JsonRef representing any entity from a parent object by key name.
Shortcut for JsonRef::name()

Example

```
JsonRef allUsers = myJsonRef("users");
```

Parameters

∗name	Pointer to name of the entity to find

Returns

The reference to the entity of type JsonRef

See also

JsonRef::name(), JsonRef::object()

23.37.2.29 operator[]()

```
JsonRef JsonRef::operator[] (
            int i )
```

Get a JsonRef from an array by numerical index.

Example

```
JsonRef userThree = myJsonRef("users")[3];
```

Returns

A JsonRef if array element is found

23.37.2.30 PrintChildren()

```
int JsonRef::PrintChildren (
            bool pretty = false )
```

Output child objects of this JsonRef as JSON text to stdout.

23.37.2.31 PrintChildrenToBuffer()

```
int JsonRef::PrintChildrenToBuffer (
            char * buffer,
            int maxlen,
            bool pretty = false )
```

Output child objects of this JsonRef as JSON text to a buffer pointer.

23.37.2.32 PrintChildrenToFd()

```
int JsonRef::PrintChildrenToFd (
            int fd,
            bool pretty = false )
```

Output child objects of this JsonRef as JSON text to a file descriptor.

23.37.2.33 ResetPosition()

```
void JsonRef::ResetPosition ( )
```

Reset the position index of the ParsedJsonDataSet to the beginning.

23.37.2.34 SkipArray()

```
void JsonRef::SkipArray ( )
```

Increment the position index of the ParsedJsonDataSet beyond this array.

23.37.2.35 SkipElement()

```
void JsonRef::SkipElement ( )
```

Increment the position index of the ParsedJsonDataSet beyond this element.

23.37.2.36 SkipObject()

```
void JsonRef::SkipObject ( )
```

Increment the position index of the ParsedJsonDataSet beyond this object.

23.37.2.37 start()

```
JsonRef JsonRef::start ( )  [inline]
```
Begin traversal: check for ref validity and reset the position.

23.37.2.38 Valid()

```
bool JsonRef::Valid ( ) const  [inline]
```
Check if the current JsonRef is a valid JSON position.

Returns

True if a JSON element exists, otherwise false.

The documentation for this class was generated from the following file:

- json_lexer.h

23.38 MACADR Class Reference

Used to store and manipulate MAC addresses.
`#include <nettypes.h>`

Public Member Functions

- bool IsNull ()
  Checks if MAC is null.
- bool IsMultiCast ()
  Checks if MAC is a Multicast MAC.
- bool IsBroadCast ()
  Checks if MAC is a Broadcast MAC.
- uint8_t GetByte (int n) const
  Get the nth byte of the MAC.
- void SetFromBytes (const uint8_t *pb)
  Set the MAC Address from a pointer to an array of 6 bytes.
- void fdprint (int fd)
  print the MAC to a file descriptor
- void print ()
  print the MAC to file descriptor 1
- MACADR operator+ (uint32_t rhs)

23.38.1 Detailed Description

Used to store and manipulate MAC addresses.

23.38.2 Member Function Documentation

23.38.2.1 fdprint()

```
void MACADR::fdprint (
            int fd )
```
print the MAC to a file descriptor

23.38.2.2 GetByte()

```
uint8_t MACADR::GetByte (
            int n ) const  [inline]
```

Get the nth byte of the MAC.

23.38.2.3 IsBroadCast()

```
bool MACADR::IsBroadCast ( )  [inline]
```

Checks if MAC is a Broadcast MAC.

23.38.2.4 IsMultiCast()

```
bool MACADR::IsMultiCast ( )  [inline]
```

Checks if MAC is a Multicast MAC.

23.38.2.5 IsNull()

```
bool MACADR::IsNull ( )  [inline]
```

Checks if MAC is null.

23.38.2.6 operator+()

```
MACADR MACADR::operator+ (
            uint32_t rhs )  [inline]
```

Todo Document

23.38.2.7 print()

```
void MACADR::print ( )  [inline]
```

print the MAC to file descriptor 1

23.38.2.8 SetFromBytes()

```
void MACADR::SetFromBytes (
            const uint8_t * pb )  [inline]
```

Set the MAC Address from a pointer to an array of 6 bytes.
The documentation for this class was generated from the following file:

- nettypes.h

23.39 mcanMODM7AE70::mcan_config Class Reference

MCAN configuration structure.
`#include <mcan.h>`

Public Member Functions

- void set_config_defaults ()

  Initializes an MCAN configuration structure to defaults.

Public Attributes

- bool run_in_standby
- uint8_t watchdog_configuration
- bool transmit_pause
- bool edge_filtering
- bool protocol_exception_handling
- bool automatic_retransmission
- bool clock_stop_request
- bool clock_stop_acknowledge
- uint8_t timestamp_prescaler
- uint16_t timeout_period
- enum mcan_timeout_mode timeout_mode
- bool timeout_enable
- bool tdc_enable
- uint8_t delay_compensation_offset
- uint8_t delay_compensation_filter_window_length
- enum mcan_nonmatching_frames_action nonmatching_frames_action_standard
- enum mcan_nonmatching_frames_action nonmatching_frames_action_extended
- bool remote_frames_standard_reject
- bool remote_frames_extended_reject
- uint32_t extended_id_mask
- bool rx_fifo_0_overwrite
- uint8_t rx_fifo_0_watermark
- bool rx_fifo_1_overwrite
- uint8_t rx_fifo_1_watermark
- bool tx_queue_mode
- uint8_t tx_event_fifo_watermark

23.39.1 Detailed Description

MCAN configuration structure.
Configuration structure for an MCAN instance. This structure should be initialized by the mcan_get_config_defaults() function before being modified by the user application.

23.39.2 Member Function Documentation

23.39.2.1 set_config_defaults()

```
void mcanMODM7AE70::mcan_config::set_config_defaults ( )  [inline]
```

Initializes an MCAN configuration structure to defaults.
Initializes a given MCAN configuration struct to a set of known default values. This function should be called on any new instance of the configuration struct before being modified by the user application.
The default configuration is as follows:

- Not run in standby mode
- Disable Watchdog
- Transmit pause enabled
- Edge filtering during bus integration enabled
- Protocol exception handling enabled
- Automatic retransmission enabled
- Clock stop request disabled

- Clock stop acknowledge disabled
- Timestamp Counter Prescaler 1
- Timeout Period with 0xFFFF
- Timeout Mode: Continuous operation
- Disable Timeout
- Transmitter Delay Compensation Offset is 0
- Transmitter Delay Compensation Filter Window Length is 0
- Reject nonmatching standard frames
- Reject nonmatching extended frames
- Reject remote standard frames
- Reject remote extended frames
- Extended ID Mask is 0x1FFFFFFF
- Rx FIFO 0 Operation Mode: overwrite
- Disable Rx FIFO 0 Watermark
- Rx FIFO 1 Operation Mode: overwrite
- Disable Rx FIFO 1 Watermark
- Tx FIFO/Queue Mode: FIFO
- Disable Tx Event FIFO Watermark

23.39.3 Member Data Documentation

23.39.3.1 automatic_retransmission

`bool mcanMODM7AE70::mcan_config::automatic_retransmission`
Automatic Retransmission.

23.39.3.2 clock_stop_acknowledge

`bool mcanMODM7AE70::mcan_config::clock_stop_acknowledge`
Clock Stop Acknowledge.

23.39.3.3 clock_stop_request

`bool mcanMODM7AE70::mcan_config::clock_stop_request`
Clock Stop Request.

23.39.3.4 delay_compensation_filter_window_length

`uint8_t mcanMODM7AE70::mcan_config::delay_compensation_filter_window_length`
Transmitter Delay Compensation Filter Window Length : 0x0-0x7F

23.39.3.5 delay_compensation_offset

`uint8_t mcanMODM7AE70::mcan_config::delay_compensation_offset`
Transmitter Delay Compensation Offset : 0x0-0x7F

23.39.3.6 edge_filtering

bool mcanMODM7AE70::mcan_config::edge_filtering
Edge Filtering during Bus Integration.

23.39.3.7 extended_id_mask

uint32_t mcanMODM7AE70::mcan_config::extended_id_mask
Extended ID Mask: 0x0-0x1FFFFFFF.

23.39.3.8 nonmatching_frames_action_extended

enum mcan_nonmatching_frames_action mcanMODM7AE70::mcan_config::nonmatching_frames_action_↩
extended
Nonmatching frames action for extended frames.

23.39.3.9 nonmatching_frames_action_standard

enum mcan_nonmatching_frames_action mcanMODM7AE70::mcan_config::nonmatching_frames_action_↩
standard
Nonmatching frames action for standard frames.

23.39.3.10 protocol_exception_handling

bool mcanMODM7AE70::mcan_config::protocol_exception_handling
Protocol Exception Handling.

23.39.3.11 remote_frames_extended_reject

bool mcanMODM7AE70::mcan_config::remote_frames_extended_reject
Reject Remote Extended Frames.

23.39.3.12 remote_frames_standard_reject

bool mcanMODM7AE70::mcan_config::remote_frames_standard_reject
Reject Remote Standard Frames.

23.39.3.13 run_in_standby

bool mcanMODM7AE70::mcan_config::run_in_standby
MCAN run in standby control.

23.39.3.14 rx_fifo_0_overwrite

bool mcanMODM7AE70::mcan_config::rx_fifo_0_overwrite
Rx FIFO 0 Operation Mode.

23.39.3.15 rx_fifo_0_watermark

uint8_t mcanMODM7AE70::mcan_config::rx_fifo_0_watermark
Rx FIFO 0 Watermark: 1-64, other value disable it.

23.39.3.16 rx_fifo_1_overwrite

bool mcanMODM7AE70::mcan_config::rx_fifo_1_overwrite
Rx FIFO 1 Operation Mode.

23.39.3.17 rx_fifo_1_watermark

uint8_t mcanMODM7AE70::mcan_config::rx_fifo_1_watermark
Rx FIFO 1 Watermark: 1-64, other value disable it.

23.39.3.18 tdc_enable

`bool mcanMODM7AE70::mcan_config::tdc_enable`
Transceiver Delay Compensation enable.

23.39.3.19 timeout_enable

`bool mcanMODM7AE70::mcan_config::timeout_enable`
Timeout enable.

23.39.3.20 timeout_mode

`enum mcan_timeout_mode mcanMODM7AE70::mcan_config::timeout_mode`
Timeout Mode.

23.39.3.21 timeout_period

`uint16_t mcanMODM7AE70::mcan_config::timeout_period`
Timeout Period.

23.39.3.22 timestamp_prescaler

`uint8_t mcanMODM7AE70::mcan_config::timestamp_prescaler`
Timestamp Counter Prescaler: 0x0-0xF

23.39.3.23 transmit_pause

`bool mcanMODM7AE70::mcan_config::transmit_pause`
Transmit Pause.

23.39.3.24 tx_event_fifo_watermark

`uint8_t mcanMODM7AE70::mcan_config::tx_event_fifo_watermark`
Tx Event FIFO Watermark: 1-32, other value disable it.

23.39.3.25 tx_queue_mode

`bool mcanMODM7AE70::mcan_config::tx_queue_mode`
Tx FIFO/Queue Mode, 0 for FIFO and 1 for Queue.

23.39.3.26 watchdog_configuration

`uint8_t mcanMODM7AE70::mcan_config::watchdog_configuration`
Start value of the Message RAM Watchdog Counter
The documentation for this class was generated from the following file:

- mcan.h

23.40 mcan_extended_message_filter_element Class Reference

MCAN extended message ID filter element structure.
`#include <mcan_internal.h>`

23.40.1 Detailed Description

MCAN extended message ID filter element structure.
Common element structure for extended message ID filter element.
The documentation for this class was generated from the following file:

- mcan_internal.h

23.41 mcanMODM7AE70::mcan_module Class Reference

MCAN Module Class.
`#include <mcan.h>`

Public Member Functions

- mcan_module (Mcan ∗hw, uint32_t baud)

  Create a MCAN module instance.
- void init (Mcan ∗hw, struct mcan_config ∗config, uint32_t baud)

  Initialize a MCAN module.
- void send_message (uint32_t id_value, uint8_t ∗data, uint32_t data_length, OS_SEM ∗pSem=0)

  Send a MCAN message.
- bool blocking_send_message (uint32_t id_value, uint8_t ∗data, uint32_t data_length, uint32_t TimeOut)

  Send a MCAN message and block until sent.
- int RegisterRxFifo (uint32_t composite_id, OS_FIFO ∗pFifo, int channel=-1)

  Register a FIFO to receive CAN messages.
- int RegisterRxFifoMask (uint32_t composite_id, uint32_t mask, OS_FIFO ∗pFifo, int channel=-1)

  Register a FIFO to receive messages as specified by the address bit mask. A value of 1 = match, a value of 0 = does not match.
- int RegisterRxFifoRange (uint32_t composite_id_low, uint32_t composite_id_hi, OS_FIFO ∗pFifo, int channel=-1)

  Register a FIFO to receive messages in the specified numeric range of addresses from low to high.
- int UnRegisterFifo (int channel)

  Unregister the FIFO for the specified CAN channel.
- int MultiCanSetRTRMessage (uint32_t id, uint8_t ∗data, uint8_t len, int channel=-1)

  Enable an auto-reply to a CAN RTR message for the specified CAN ID. The reply message is specified by the data parameter.
- int MultiCanReplaceRTRMessage (int channel, uint8_t ∗data, uint8_t len)

  Modifies/updates the response for the RTR message registered wotj MultiCanSetRTRMessage().
- int MultiCanStopRTRMessage (int channel)

  Unregister the RTR message for the specified channel.

23.41.1 Detailed Description

MCAN Module Class.
Class to control a MCAN modules (peripherals) on the MODM7AE70

23.41.2 Constructor & Destructor Documentation

23.41.2.1 mcan_module()

```
mcanMODM7AE70::mcan_module::mcan_module (
            Mcan * hw,
            uint32_t baud )
```

Create a MCAN module instance.

Parameters

hw	MCAN hardware peripheral, MCAN0 or MCAN 1
baud	MCAN baud rate. For example, 500000

23.41.3 Member Function Documentation

23.41.3.1 blocking_send_message()

```
bool mcanMODM7AE70::mcan_module::blocking_send_message (
            uint32_t id_value,
            uint8_t * data,
            uint32_t data_length,
            uint32_t TimeOut )
```

Send a MCAN message and block until sent.

Parameters

id_value	CAN ID, 11-bit standard or 29-bit extended
data	Pointer to message data
data_length	Length of data in bytes
TimeOut	Time out value in system time ticks. For example, TICKS_PER_SECOND * 1

23.41.3.2 init()

```
void mcanMODM7AE70::mcan_module::init (
            Mcan * hw,
            struct mcan_config * config,
            uint32_t baud )
```

Initialize a MCAN module.

Parameters

hw	MCAN hardware peripheral, MCAN0 or MCAN 1
config	Pointer to a mcan_config object, initialized with appropriate values.
baud	MCAN baud rate. For example, 500000

23.41.3.3 MultiCanReplaceRTRMessage()

```
int mcanMODM7AE70::mcan_module::MultiCanReplaceRTRMessage (
            int channel,
            uint8_t * data,
            uint8_t len )
```

Modifies/updates the response for the RTR message registered wotj MultiCanSetRTRMessage().

Returns

The channel number registered for the RTR message.

See also

MultiCanSetRTRMessage(), MultiCanStopRTRMessage()

23.41.3.4 MultiCanSetRTRMessage()

```
int mcanMODM7AE70::mcan_module::MultiCanSetRTRMessage (
            uint32_t id,
```

```
            uint8_t * data,
            uint8_t len,
            int channel = -1 )
```

Enable an auto-reply to a CAN RTR message for the specified CAN ID. The reply message is specified by the data parameter.
The response message can be modified with MultiCanReplaceRTRMessage().

Parameters

id	CAN ID, 11-bit standard or 29-bit extended
data	Pointer to message data
len	Length of message
channel	Optional CAN channel

Returns

The channel number corresponding to the ID and RTR message.

See also

MultiCanReplaceRTRMessage(), MultiCanStopRTRMessage()

23.41.3.5 MultiCanStopRTRMessage()

```
int mcanMODM7AE70::mcan_module::MultiCanStopRTRMessage (
            int channel )
```

Unregister the RTR message for the specified channel.

Parameters

channel	CAN channel number

Returns

OS_NO_ERR on success

See also

MultiCanSetRTRMessage(), MultiCanReplaceRTRMessage()

23.41.3.6 RegisterRxFifo()

```
int mcanMODM7AE70::mcan_module::RegisterRxFifo (
            uint32_t composite_id,
            OS_FIFO * pFifo,
            int channel = -1 )
```

Register a FIFO to receive CAN messages.

Parameters

composite_id	CAN ID, 11-bit standard or 29-bit extended
pFifo	Pointer to OS_FIFO for received data
channel	Optional CAN channel

Returns

OS_NO_ERR on success

23.41.3.7 RegisterRxFifoMask()

```
int mcanMODM7AE70::mcan_module::RegisterRxFifoMask (
            uint32_t composite_id,
            uint32_t mask,
            OS_FIFO * pFifo,
            int channel = -1 )
```

Register a FIFO to receive messages as specified by the address bit mask. A value of 1 = match, a value of 0 = does not match.

Parameters

composite_id	CAN ID, 11-bit standard or 29-bit extended
mask	Id mask
pFifo	Pointer to OS_FIFO for received data
channel	Optional CAN channel

Returns

OS_NO_ERR on success

23.41.3.8 RegisterRxFifoRange()

```
int mcanMODM7AE70::mcan_module::RegisterRxFifoRange (
            uint32_t composite_id_low,
            uint32_t composite_id_hi,
            OS_FIFO * pFifo,
            int channel = -1 )
```

Register a FIFO to receive messages in the specified numeric range of addresses from low to high.

Parameters

composite_id_low	Composite ID low value
composite_id_hi	Composite ID high value
pFifo	Pointer to OS_FIFO for received data
channel	Optional CAN channel

Returns

OS_NO_ERR on success

23.41.3.9 send_message()

```
void mcanMODM7AE70::mcan_module::send_message (
            uint32_t id_value,
            uint8_t * data,
            uint32_t data_length,
            OS_SEM * pSem = 0 )
```

Send a MCAN message.

Parameters

id_value	CAN ID, 11-bit standard or 29-bit extended
data	Pointer to message data
data_length	Length of data in bytes
pSem	Optional pointer to semaphore to post to, default is no post

23.41.3.10 UnRegisterFifo()

```
int mcanMODM7AE70::mcan_module::UnRegisterFifo (
            int channel )
```

Unregister the FIFO for the specified CAN channel.

Parameters

channel	CAN channel number

Returns

OS_NO_ERR on success

The documentation for this class was generated from the following file:

- mcan.h

23.42 mcan_rx_element_buffer Struct Reference

MCAN receive element structure for buffer.
`#include <mcan_internal.h>`

23.42.1 Detailed Description

MCAN receive element structure for buffer.
The documentation for this struct was generated from the following file:

- mcan_internal.h

23.43 mcan_tx_element Struct Reference

MCAN transfer element structure.
`#include <mcan_internal.h>`

23.43.1 Detailed Description

MCAN transfer element structure.
Common element structure for transfer buffer and FIFO/QUEUE.
The documentation for this struct was generated from the following file:

- mcan_internal.h

23.44 mcan_tx_event_element Struct Reference

MCAN transfer event FIFO element structure.
`#include <mcan_internal.h>`

23.44.1 Detailed Description

MCAN transfer event FIFO element structure.
Common element structure for transfer event FIFO.
The documentation for this struct was generated from the following file:

- mcan_internal.h

23.45 NBRtosInitObj Class Reference

A simple class to derive from if you are creating tasks that are constructed at global scope and need to do RTOS initalization.
`#include <nbrtos.h>`

Public Member Functions

- virtual void **Notify** ()=0

  This will be called when the RTOS has been setup in initalization.

23.45.1 Detailed Description

A simple class to derive from if you are creating tasks that are constructed at global scope and need to do RTOS initalization.
The vistural function Notify will be called once the RTOS internals are setup so you can create a task, allocate buffers, etc. If you need to do network I/O, then you this will not be sufficent as you will need register to be notifyied when link is active on the interface.

See also

NetInterface::LinkActiveFuncPtr

The documentation for this class was generated from the following file:

- nbrtos.h

23.46 NBString Class Reference

Lightweight alternative to C++ CString class.
`#include <nbstring.h>`
Inherited by NBInterpolatedString.

Public Member Functions

- **NBString** ()

  Construct a NBString object.
- NBString (const NBString &str)

  Construct a new NBString object from an existing NBString object.
- NBString (const NBString &str, size_t pos, size_t len=npos)

  Construct a new NBString object from a substring of an existing NBString object.
- NBString (const char *s)

  Construct a NBString object from a character string (null terminated array of characters)
- NBString (const char *s, size_t n)

  Construct a NBString object from a character string up to the specified amount.
- ~**NBString** ()

  NBString destructor.
- const char * c_str () const

  *Method to pass a NBString as a constant char *.*

- NBString substr (size_t pos=0, size_t len=npos) const

  Creates a new NBString object from a substring of the existing NBString object.
- int compare (const NBString &str) const

  Compares two NBString objects.
- int compare (const char *s) const

  Compares the NBString object to a character string.
- size_t find (const NBString str, size_t pos=0) const

  Find a substring within a NBString object.
- size_t find (const char *str, size_t pos=0) const

  Find a substring within a character string.
- size_t find (const char *s, size_t pos, size_t n) const

  Find a substring within a character string.
- size_t find (char c, size_t pos=0) const

  Find the first occurence of a character within the NBString object.
- size_t replace (const char *findStr, char rep, size_t startPos=0, size_t end=0)

  Replace all occurrences of all "<findStr>" characters within the NBString object.
- size_t replace (char c, char rep, size_t startPos=0, size_t end=0)

  Replace all occurrences of all "<findStr>" characters within the NBString object.
- size_t size () const

  Returns the current size of memory allocated for the string.
- size_t length () const

  Returns the length of the string.
- void clear ()

  Clear a NBString object and free allocated memory.
- bool empty () const

  Check if a string is empty.
- NBString & Append (const char *str, size_t len)

  Append a string to an existing NBString object.
- NBString & FdAppend (int fd, size_t len)

  Append from a file descriptor to an existing NBString object.
- NBString & Reserve (size_t len)

  Reserve an additional buffer amount.
- int vsiprintf (const char *format, va_list &vl)

  Print to a string with formatting.
- int sprintf (const char *format,...)

  sprintf to a string with formatting to a character array
- int sprintf (NBString const format,...)

  sprintf to a string with formatting as a NBString object
- int siprintf (const char *format,...)

  isprintf (integer) to a string with formatting to a character array
- int siprintf (NBString const format,...)

  isprintf (integer) to a string with formatting to a NBString object
- int stoi () const

  Parse the string and convert to an integer number.
- long stol () const

  Parse the string value and convert to a long integer number.
- unsigned int stoui () const

  Parse the string value and convert to a const integer number.
- unsigned long stoul () const

  Parse the string value and convert to a const unsigned long integer number.
- double stod () const

Parse the string value and convert to a const double float number.

- IPADDR to_ipaddr () const

  Parse the string value and convert to an IPADDR IP address.

- size_t copy (char *s, size_t len, size_t pos=0) const

  Copy a substring, does not null terminate.

- size_t strcopy (char *s, size_t len, size_t pos=0) const

  Copy a substring, with null termination.

Friends

- void swap (NBString &x, NBString &y)

  Swaps the contents of two NBString objects.

23.46.1 Detailed Description

Lightweight alternative to C++ CString class.

23.46.2 Constructor & Destructor Documentation

23.46.2.1 NBString() [1/4]

```
NBString::NBString (
            const NBString & str )
```

Construct a new NBString object from an existing NBString object.

Parameters

str	Existing NBString object

23.46.2.2 NBString() [2/4]

```
NBString::NBString (
            const NBString & str,
            size_t pos,
            size_t len = npos )
```

Construct a new NBString object from a substring of an existing NBString object.

Parameters

str	Existing NBString object
pos	Starting position to copy
len	Ending position to copy

23.46.2.3 NBString() [3/4]

```
NBString::NBString (
            const char * s )
```

Construct a NBString object from a character string (null terminated array of characters)

Parameters

s	String to initialize object

23.46.2.4 NBString() [4/4]

```
NBString::NBString (
            const char * s,
            size_t n )
```

Construct a NBString object from a character string up to the specified amount.

Parameters

s	String to initialize object
n	Number of characters to copy

23.46.3 Member Function Documentation

23.46.3.1 Append()

```
NBString & NBString::Append (
            const char * str,
            size_t len )
```

Append a string to an existing NBString object.
Append a string to an existing NBString object

Parameters

str	String to append
len	Number of characters to append

Returns

The updated NBString object

23.46.3.2 c_str()

```
const char * NBString::c_str ( ) const
```

Method to pass a NBString as a constant char *.
Will cause an allocation if it is pointing to a constant string longer than the currently allocated length.

Returns

A null terminated character array representing the string

23.46.3.3 clear()

```
void NBString::clear ( )
```

Clear a NBString object and free allocated memory.
Clear a NBString object and free allocated memory

23.46.3.4 compare() [1/2]

```
int NBString::compare (
            const char * s ) const
```

Compares the NBString object to a character string.

Parameters

s	String to compare

Return values

0	If identical
-1	If NBString object is less than str
1	If NBString object is greater than str

23.46.3.5 compare() [2/2]

```
int NBString::compare (
            const NBString & str ) const
```

Compares two NBString objects.

Parameters

str	NBString object to compare

Return values

0	If identical
-1	If NBString object is less than str
1	If NBString object is greater than str

23.46.3.6 copy()

```
size_t NBString::copy (
            char * s,
            size_t len,
            size_t pos = 0 ) const
```

Copy a substring, does not null terminate.

Parameters

s	Destination
len	Size of destination
pos	Starting position of copy

Returns

Number of bytes copied

23.46.3.7 empty()

```
bool NBString::empty ( ) const
```

Check if a string is empty.
Check if a NBString object is empty

Return values

true	String is null
false	String contains data

23.46.3.8 FdAppend()

```
NBString & NBString::FdAppend (
            int fd,
            size_t len )
```

Append from a file descriptor to an existing NBString object.
Append from a file descriptor to an existing NBString object

Parameters

fd	File descriptor to append from
len	Number of characters to append

Returns

The updated NBString object

23.46.3.9 find() [1/4]

```
size_t NBString::find (
            char c,
            size_t pos = 0 ) const
```

Find the first occurence of a character within the NBString object.

Parameters

c	Character to search for
pos	Starting position to start search

Return values

>=0	Position in string
-1	If character not found

23.46.3.10 find() [2/4]

```
size_t NBString::find (
            const char * s,
            size_t pos,
            size_t n ) const
```

Find a substring within a character string.

Parameters

s	String to search
pos	Starting position to start search
n	Number of characters that have to match

Return values

>=0	Position in searched string
-1	If substring not found

23.46.3.11 find() [3/4]

```
size_t NBString::find (
            const char * str,
            size_t pos = 0 ) const
```

Find a substring within a character string.

Parameters

str	String to search
pos	Starting position to start search

Return values

>=0	Position in searched string
-1	If substring not found

23.46.3.12 find() [4/4]

```
size_t NBString::find (
            const NBString str,
            size_t pos = 0 ) const
```

Find a substring within a NBString object.

Parameters

str	NString object to search
pos	Starting position to start search

Return values

>=0	Position in searched string
-1	If substring not found

23.46.3.13 length()

```
size_t NBString::length ( ) const
```

Returns the length of the string.

Returns

The length of the string

23.46.3.14 replace() [1/2]

```
size_t NBString::replace (
            char c,
            char rep,
            size_t startPos = 0,
            size_t end = 0 )
```

Replace all occurrences of all "<findStr>" characters within the NBString object.

Parameters

c	Character to be replaced within the NBString
rep	Character to replace with
startPos	Starting position to start replacing
end	Replace all matching characters before this position, 0 = end of NBString

Return values

>0	Number of Replaced Characters
-1	If no matching character found

23.46.3.15 replace() [2/2]

```
size_t NBString::replace (
            const char * findStr,
            char rep,
            size_t startPos = 0,
            size_t end = 0 )
```

Replace all occurrences of all "<findStr>" characters within the NBString object.

Parameters

findStr	String of characters to be replaced within the NBString
rep	Character to replace with
startPos	Starting position to start replacing
end	Replace all matching characters before this position, 0 = end of NBString

Return values

>0	Number of Replaced Characters
-1	If no matching character found

23.46.3.16 Reserve()

```
NBString & NBString::Reserve (
            size_t len )
```

Reserve an additional buffer amount.
Reserve an additional buffer amount to prepare for a series of Appends

Parameters

len	Number of additional characters to prepare to store

Returns

The updated NBString object

23.46.3.17 siprintf() [1/2]

```
int NBString::siprintf (
            const char * format,
             ...  )
```

isprintf (integer) to a string with formatting to a character array

Parameters

format	printf style formatting

23.46.3.18 siprintf() [2/2]

```
int NBString::siprintf (
            NBString const format,
             ...  )
```

isprintf (integer) to a string with formatting to a NBString object

Parameters

format	printf style formatting

23.46.3.19 size()

```
size_t NBString::size ( ) const
```

Returns the current size of memory allocated for the string.

Returns

Total size of the object in bytes

23.46.3.20 sprintf() [1/2]

```
int NBString::sprintf (
```

```
            const char * format,
             ...  )
```

sprintf to a string with formatting to a character array

Parameters

format	printf style formatting

23.46.3.21 sprintf() [2/2]

```
int NBString::sprintf (
            NBString const format,
             ...  )
```

sprintf to a string with formatting as a NBString object

Parameters

format	printf style formatting

23.46.3.22 stod()

```
double NBString::stod ( ) const
```

Parse the string value and convert to a const double float number.

Returns

const double float value

23.46.3.23 stoi()

```
int NBString::stoi ( ) const
```

Parse the string and convert to an integer number.

Returns

integer value

23.46.3.24 stol()

```
long NBString::stol ( ) const
```

Parse the string value and convert to a long integer number.

Returns

long integer value

23.46.3.25 stoui()

```
unsigned int NBString::stoui ( ) const
```

Parse the string value and convert to a const integer number.

Returns

unsigned integer value

23.46.3.26 stoul()

```
unsigned long NBString::stoul ( ) const
```

Parse the string value and convert to a const unsigned long integer number.

Returns

unsigned long integer value

23.46.3.27 strcopy()

```
size_t NBString::strcopy (
            char * s,
            size_t len,
            size_t pos = 0 ) const
```

Copy a substring, with null termination.

Parameters

s	Destination
len	Size of destination
pos	Starting position of copy

Returns

Number of bytes copied

23.46.3.28 substr()

```
NBString NBString::substr (
            size_t pos = 0,
            size_t len = npos ) const
```

Creates a new NBString object from a substring of the existing NBString object.

Parameters

pos	Starting position of substring
len	Number of characters to copy

Returns

A NBString object

23.46.3.29 to_ipaddr()

```
IPADDR NBString::to_ipaddr ( ) const
```

Parse the string value and convert to an IPADDR IP address.

Returns

const double float value

23.46.3.30 vsiprintf()

```
int NBString::vsiprintf (
            const char * format,
            va_list & vl )
```

Print to a string with formatting.

Parameters

format	printf style formatting
vl	Variable argument list

23.46.4 Friends And Related Function Documentation

23.46.4.1 swap

```
void swap (
            NBString & x,
            NBString & y )  [friend]
```

Swaps the contents of two NBString objects.
Useful for optimizing sorts. Friend function that takes two NBString objects as parameters and swaps their contents.
The documentation for this class was generated from the following file:

- nbstring.h

23.47 NV_SettingsStruct Struct Reference

Configuration Settings.
`#include <serialburnerdata.h>`
Inherits config_obj.

Additional Inherited Members

Public Member Functions inherited from config_obj

- config_obj (config_obj &owner, const char ∗name, const char ∗desc)

  Object constructor with the parent/owner leaf parameter.
- config_obj (const char ∗name, const char ∗desc)

  Object constructor.
- virtual void GetTextValue (NBString &s)

  Get the object value as a text string to the specified NBString object.
- virtual void GetTypeValue (NBString &s)

  Assigns the object type value to the specified NBString object.

23.47.1 Detailed Description

Configuration Settings.
DeviceName - Device name for DHCP NetBIOSName - NetBIOS name

- SSL and SSH ∗ CertificateRsaLength - Certificate length CertificateData - Certificate KeyHttpsRsaLength - RSA key for HTTPS length, 0 is none KeyHttpsRsaData - RSA key for HTTPS KeyRsaLength - RSA key length, 0 is none KeyRsaData - RSA key KeyDsaLength - DSA key length, 0 is none KeyDsaData - DSA key
- Version change key ∗ VerifyKey - Version change key

The documentation for this struct was generated from the following files:

- Parallax/src/nvsettings.h
- SaveUserParameters/src/main.cpp
- serial/SerialBurner/src/nvsettings.h
- SSH/SecureSerToEthFactoryApp/src/serialburnerdata.h
- SSH/SshServerUserKey/src/nvsettings.h
- SSL/HttpsUploadCert/src/serialburnerdata.h
- SSL/SslClientVerifyPeerEffs/src/serialburnerdata.h
- webif.h

23.48 OS_CRIT Struct Reference

An OS_CRIT object is used to establish critical sections of code that can only be run by one task at a time. Tasks that try to claim a critical section which is currently claimed by another task will stop and wait for that task to release the critical section before continuing execution.
`#include <nbrtos.h>`
Inherits OS_TASK_DLY_OBJ.

Public Member Functions

- OS_CRIT ()

  Create and initialize an OS_CRIT object.
- uint8_t Init ()

  Initialize an OS_CRIT object to its default state.
- uint8_t LockAndEnter (uint32_t timeoutTicks=WAIT_FOREVER)

  Locks the current task to prevent task switching and claims the critical section.
- uint8_t Enter (TickTimeout &t)

  Request to Enter/Claim the critical section.
- uint8_t Enter (uint32_t timeoutTicks=WAIT_FOREVER)

  Request to Enter/Claim the critical section, with a time out parameter.
- uint8_t EnterNoWait ()

  Request to Enter/Claim the critical section. Does not wait if a critical section is not available.
- uint8_t Leave ()

  Release the critical section.
- uint8_t LeaveAndUnlock ()

  Leave/release the critical section and unlock the task.
- bool UsedFromISR ()

  Check if critical section UsedFromISR flag is set.
- void SetUseFromISR (bool enableFromISR)

  Set critical section UsedFromISR flag.
- bool OwnedByCurTask ()

  Check if critical section owned by the current task.
- uint32_t CurDepth ()

  Returns the critical section depth count.

Friends

- void ForceReboot (bool fromIRQ)

  Forces the system hardware to perform a soft reset.

23.48.1 Detailed Description

An OS_CRIT object is used to establish critical sections of code that can only be run by one task at a time. Tasks that try to claim a critical section which is currently claimed by another task will stop and wait for that task to release the critical section before continuing execution.

23.48.2 Constructor & Destructor Documentation

23.48.2.1 OS_CRIT()

```
OS_CRIT::OS_CRIT ( )  [inline]
```

Create and initialize an OS_CRIT object.

See also

Init()

23.48.3 Member Function Documentation

23.48.3.1 CurDepth()

```
uint32_t OS_CRIT::CurDepth ( )  [inline]
```

Returns the critical section depth count.

Returns

Depth count value

23.48.3.2 Enter() [1/2]

```
uint8_t OS_CRIT::Enter (
            TickTimeout & t )
```

Request to Enter/Claim the critical section.

Note

You must call Leave() once for each successful Enter() call to release the critical section so that another task can run it.

Parameters

t	The number of TickTimeout time ticks to wait for the critical section. A timeout of 0 (zero) waits forever.

Return values

OS_NO_ERR	If we were successful in claiming the critical section or if our task owns it
OS_TIMEOUT	If we were unable to claim the section NBRTOS Error Codes

See also

NBRTOS Error Codes, EnterNoWait(), Leave()

23.48.3.3 Enter() [2/2]

```
uint8_t OS_CRIT::Enter (
            uint32_t timeoutTicks = WAIT_FOREVER )  [inline]
```

Request to Enter/Claim the critical section, with a time out parameter.

Note

You must call Leave() once for each successful Enter() call to release the critical section so that another task can run it.

Parameters

timeoutTicks	The number of sytem TimeTicks to wait for the critical section. A timeout of 0 (zero) waits forever.

Return values

OS_NO_ERR	If we were successful in claiming the critical section or if our task owns it
OS_TIMEOUT	If we were unable to claim the section NBRTOS Error Codes

See also

NBRTOS Error Codes, EnterNoWait(), Leave()

23.48.3.4 EnterNoWait()

```
uint8_t OS_CRIT::EnterNoWait ( )
```

Request to Enter/Claim the critical section. Does not wait if a critical section is not available.

Note

You must call Leave() once for each successful Enter() call to release the critical section so that another task can run it.

Return values

OS_NO_ERR	If we were successful in claiming the critical section or if our task owns it
OS_TIMEOUT	If we were unable to claim the section NBRTOS Error Codes

See also

NBRTOS Error Codes, Enter(), Leave()

23.48.3.5 Init()

```
uint8_t OS_CRIT::Init ( )
```

Initialize an OS_CRIT object to its default state.

Returns

OS_NO_ERR if successful, otherwise NBRTOS Error Codes

See also

NBRTOS Error Codes

23.48.3.6 Leave()

```
uint8_t OS_CRIT::Leave ( )
```

Release the critical section.

Note

This function must be called once for each successful Enter() or EnterNoWait() call to release the critical section so another task can run it.

Return values

OS_NO_ERR	If we were successful in claiming the critical section or if our task owns it
OS_CRIT_ERR	If we are trying to release a critical section that we do not own NBRTOS Error Codes

See also

NBRTOS Error Codes, Enter(), EnterNoWait()

23.48.3.7 LeaveAndUnlock()

```
uint8_t OS_CRIT::LeaveAndUnlock ( )
```

Leave/release the critical section and unlock the task.

Note

This function must be called once for each successful LockAndEnter() call to both unlock and release the critical section so another task can run it.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

NBRTOS Error Codes, LockAndEnter()

23.48.3.8 LockAndEnter()

```
uint8_t OS_CRIT::LockAndEnter (
            uint32_t timeoutTicks = WAIT_FOREVER )
```

Locks the current task to prevent task switching and claims the critical section.

Note

If it is unable to claim the critical section and times out, then the lock will be released. Otherwise LeaveAndUnlock() must be called once for each successful LockAndEnter() call to both unlock and release the critical section so that another task might run it.

Parameters

timeoutTicks	The number of TickTimeout time ticks to wait for the critical section. A timeout of 0 (zero) waits forever.

Return values

OS_NO_ERR	If we were successful in claiming the critical section or if our task owns it
OS_TIMEOUT	If we were unable to claim the section

See also

NBRTOS Error Codes, LeaveAndUnlock(), Enter()

23.48.3.9 SetUseFromISR()

```
void OS_CRIT::SetUseFromISR (
            bool enableFromISR )  [inline]
```

Set critical section UsedFromISR flag.

Parameters

enableFromISR	Set true to set flag

See also

UsedFromISR()

23.48.3.10 UsedFromISR()

```
bool OS_CRIT::UsedFromISR ( )  [inline]
```

Check if critical section UsedFromISR flag is set.

See also

SetUseFromISR()

23.48.4 Friends And Related Function Documentation

23.48.4.1 ForceReboot

```
void ForceReboot (
            bool fromIRQ )  [friend]
```

Forces the system hardware to perform a soft reset.
The documentation for this struct was generated from the following file:

- nbrtos.h

23.49 OS_FIFO Struct Reference

`#include <nbrtos.h>`
Inherits OS_TASK_DLY_OBJ.

Public Member Functions

- OS_FIFO ()

  Create and initialize a FIFO object.
- uint8_t Init ()

Sets the FIFO object to its initial state.

- uint8_t Post (OS_FIFO_EL *pToPost)

  Post a message to the next available location in the FIFO.
- uint8_t PostFirst (OS_FIFO_EL *pToPost)

  Post a message to the head of the FIFO.
- OS_FIFO_EL * Pend (uint32_t timeoutTicks, uint8_t &result)

  Wait the specified number of time ticks for some other task to post to the FIFO.
- OS_FIFO_EL * Pend (TickTimeout &t, uint8_t &result)

  Wait the specified number of TickTimeout ticks for some other task to post to the FIFO.
- OS_FIFO_EL * Pend (uint32_t timeoutTicks=WAIT_FOREVER)

  Pend on a FIFO for the specified number of system TimeTicks.
- OS_FIFO_EL * PendUntil (uint32_t timeoutTime, uint8_t &result)

  Wait the specified TimeTicks value for some other task to post to the FIFO.
- OS_FIFO_EL * PendNoWait (uint8_t &result)

  Attempts to pend a structure to the FIFO, but does not wait.
- OS_FIFO_EL * PendNoWait ()

  Attempts to pend a structure to the FIFO, but does not wait.

23.49.1 Detailed Description

A FIFO is similar to a queue, but is specifically designed to pass pointers to OS_FIFO structures. The first parameter of the structure must be a (void *) element, which is used by the operating system to create a linked list of FIFOs. When initializing a FIFO, you do not specify the maximum number of entries as with a queue. Instead, your application has the ability (and responsibility) to allocate memory (static or dynamic) in which to store the structures. This can be done statically by declaring global variables, or dynamically by allocating memory from the heap. As with a queue, the first message posted to the FIFO will be the first message extracted from the queue.

23.49.2 Constructor & Destructor Documentation

23.49.2.1 OS_FIFO()

```
OS_FIFO::OS_FIFO ( )  [inline]
```

Create and initialize a FIFO object.

See also

Init()

23.49.3 Member Function Documentation

23.49.3.1 Init()

```
uint8_t OS_FIFO::Init ( )
```

Sets the FIFO object to its initial state.

Returns

OS_NO_ERR if successful, otherwise NBRTOS Error Codes

See also

NBRTOS Error Codes

23.49.3.2 Pend() [1/3]

```
OS_FIFO_EL * OS_FIFO::Pend (
            TickTimeout & t,
            uint8_t & result )
```

Wait the specified number of TickTimeout ticks for some other task to post to the FIFO.

Parameters

t	The number of system ticks to wait. A value of 0 will wait forever.
result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

OS_FIFO_EL	Pointer to OS_FIFO_EL if successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.49.3.3 Pend() [2/3]

```
OS_FIFO_EL * OS_FIFO::Pend (
            uint32_t timeoutTicks,
            uint8_t & result )  [inline]
```

Wait the specified number of time ticks for some other task to post to the FIFO.

Parameters

timeoutTicks	The number of system ticks to wait. A value of 0 will wait forever.
result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

OS_FIFO_EL	Pointer to OS_FIFO_EL if successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), Pend()

23.49.3.4 Pend() [3/3]

```
OS_FIFO_EL * OS_FIFO::Pend (
            uint32_t timeoutTicks = WAIT_FOREVER )  [inline]
```

Pend on a FIFO for the specified number of system TimeTicks.

Parameters

timeoutTicks	The number of system TimeTicks to wait. If not specified, will wait forever.

Return values

OS_FIFO_EL	Pointer to OS_FIFO_EL if successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), Pend(), PendUntil()

23.49.3.5 PendNoWait() [1/2]

```
OS_FIFO_EL * OS_FIFO::PendNoWait ( )  [inline]
```

Attempts to pend a structure to the FIFO, but does not wait.

Return values

OS_FIFO_EL	Pointer to OS_FIFO_EL if successful
NULL	If it timed out

See also

NBRTOS Error Codes, Pend(), PendUntil()

23.49.3.6 PendNoWait() [2/2]

```
OS_FIFO_EL * OS_FIFO::PendNoWait (
            uint8_t & result )
```

Attempts to pend a structure to the FIFO, but does not wait.

Parameters

result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

OS_FIFO_EL	Pointer to OS_FIFO_EL if successful
NULL	If it timed out

See also

NBRTOS Error Codes, Pend(), PendUntil()

23.49.3.7 PendUntil()

```
OS_FIFO_EL * OS_FIFO::PendUntil (
            uint32_t timeoutTime,
            uint8_t & result )  [inline]
```

Wait the specified TimeTicks value for some other task to post to the FIFO.

Parameters

timeoutTime	The value of system TimeTicks to wait for
result	Reference to store the status of the function call, NBRTOS Error Codes. `OS_NO_ERR` if successful, `OS_TIMEOUT` if it timed out.

Return values

OS_FIFO_EL	Pointer to OS_FIFO_EL if successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), Pend()

23.49.3.8 Post()

```
uint8_t OS_FIFO::Post (
            OS_FIFO_EL * pToPost )
```

Post a message to the next available location in the FIFO.

Parameters

pToPost	A pointer to the structure cast as an OS_FIFO_EL to be posted to the FIFO.

Return values

OS_NO_ERR	If successful
OS_Q_FULL	If the queue is full, NBRTOS Error Codes

See also

PostFirst(), PostUnique(), Pend()

23.49.3.9 PostFirst()

```
uint8_t OS_FIFO::PostFirst (
            OS_FIFO_EL * pToPost )
```

Post a message to the head of the FIFO.

Parameters

pToPost	A pointer to the user's structure cast as an `OS_FIFO_EL` to be posted to the FIFO.

Return values

OS_NO_ERR	If successful
OS_Q_FULL	If the queue is full, NBRTOS Error Codes

See also

Post(), PostUniqueFirst(), Pend()

The documentation for this struct was generated from the following file:

- nbrtos.h

23.50 os_fifo_el Struct Reference

OS_FIFO element definition.
`#include <nbrtos.h>`
Inherited by pool_buffer.

Public Attributes

- union {
  struct os_fifo_el * **pNextFifo_El**
  Pointer to next OS_FIFO element.
  puint8_t **pAsBytePtr**
  Next OS_FIFO element data byte pointer.
  };

  This struture is a union of a pointer to the next OS_FIFO and a byte pointer to the same data.

23.50.1 Detailed Description

OS_FIFO element definition.
This struture is a union of a pointer to the next OS_FIFO and a byte pointer to the same data.

23.50.2 Member Data Documentation

23.50.2.1

`union { ... } os_fifo_el::@74`
This struture is a union of a pointer to the next OS_FIFO and a byte pointer to the same data.
The documentation for this struct was generated from the following file:

- nbrtos.h

23.51 OS_FLAGS Struct Reference

OSFlags enables a function or task to pend on multiple flags or events.
`#include <nbrtos.h>`

Public Member Functions

- OS_FLAGS ()
  Create and initialize an OS_FLAG object.
- void Init ()
  Initialize an OS_FLAG object to its default value.
- void Set (uint32_t bits_to_set)
  Sets the specified flag bits.
- void Clear (uint32_t bits_to_clr)
  Clear the specified flag bits.

- void Write (uint32_t bits_to_force)

  Set the flag bits to match the specified value.
- uint32_t State ()

  Returns the current values of the flags stored in the OS_FLAGS object.
- uint8_t PendAny (uint32_t bit_mask, uint16_t timeout=WAIT_FOREVER)

  Wait the specified number of system TimeTicks for any of the flags in the bit mask to be set.
- uint8_t PendAny (uint32_t bit_mask, TickTimeout &timeout)

  Wait the specified number of TickTimeout time ticks for any of the flags in the bit mask to be set.
- uint8_t PendAnyUntil (uint32_t bit_mask, uint32_t end_time)

  Wait until the specified system TimeTicks value for any of the flags in the bit mask to be set.
- uint8_t PendAnyNoWait (uint32_t bit_mask)

  Check for the specified flags and return immediately.
- uint8_t PendAll (uint32_t bit_mask, uint16_t timeout=WAIT_FOREVER)

  Wait the specified number of system time ticks until all the specified flags are set.
- uint8_t PendAll (uint32_t bit_mask, TickTimeout &timeout)

  Wait the specified number of TickTimeout time ticks until all the specified flags are set.
- uint8_t PendAllUntil (uint32_t bit_mask, uint32_t end_time)

  Wait until the specified system TimeTicks value for all of the flags in the bit mask to be set.
- uint8_t PendAllNoWait (uint32_t bit_mask)

  Wait the specified number of system time ticks until all the specified flags are set.

23.51.1 Detailed Description

OSFlags enables a function or task to pend on multiple flags or events.
In contrast to a OSSemaphore which can pend on only a single event. The OSFlag implementation is essentially a 32-bit bitmap in which each bit position represents a flag. You create a OSFlag object with OSFlagCreate(), then set, clean and read the flags with the appropriate function. There are a number of functions used to monitor or pend on the flags, and provide the ability to pend on any one or more of the flags being set, or pending on all of flags being set at one time.

23.51.2 Constructor & Destructor Documentation

23.51.2.1 OS_FLAGS()

```
OS_FLAGS::OS_FLAGS ( )
```

Create and initialize an OS_FLAG object.

See also

Init()

23.51.3 Member Function Documentation

23.51.3.1 Clear()

```
void OS_FLAGS::Clear (
            uint32_t bits_to_clr )
```

Clear the specified flag bits.

Parameters

bits_to_clr	A bit or set of bits to be cleared

See also

Set(), State()

23.51.3.2 Init()

```
void OS_FLAGS::Init ( )
```

Initialize an OS_FLAG object to its default value.

See also

OS_FLAGS()

23.51.3.3 PendAll() [1/2]

```
uint8_t OS_FLAGS::PendAll (
            uint32_t bit_mask,
            TickTimeout & timeout )
```

Wait the specified number of TickTimeout time ticks until all the specified flags are set.

Parameters

bit_mask	A bit or set of bits to wait on.
timeout	Number of time ticks to wait for all specified flag bits to be set. A value of 0 waits forever.

Return values

OS_NO_ERR	If the flags condition is satisfied
OS_TIMEOUT	If the timeout expired

See also

PendAllNoWait(), PendAny()

23.51.3.4 PendAll() [2/2]

```
uint8_t OS_FLAGS::PendAll (
            uint32_t bit_mask,
            uint16_t timeout = WAIT_FOREVER )  [inline]
```

Wait the specified number of system time ticks until all the specified flags are set.

Parameters

bit_mask	A bit or set of bits to wait on.
timeout	Number of time ticks to wait for all specified flag bits to be set. A value of 0 waits forever.

Return values

OS_NO_ERR	If the flags condition is satisfied
OS_TIMEOUT	If the timeout expired

See also

PendAllNoWait(), PendAny()

23.51.3.5 PendAllNoWait()

```
uint8_t OS_FLAGS::PendAllNoWait (
            uint32_t bit_mask )
```

Wait the specified number of system time ticks until all the specified flags are set.

Parameters

bit_mask	A bit or set of bits to wait on.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

PendAll(), PendAnyNoWait()

23.51.3.6 PendAllUntil()

```
uint8_t OS_FLAGS::PendAllUntil (
            uint32_t bit_mask,
            uint32_t end_time ) [inline]
```

Wait until the specified system TimeTicks value for all of the flags in the bit mask to be set.

Parameters

bit_mask	A bit or set of bits to wait on.
end_time	System TimeTick value to wait for.

Return values

OS_NO_ERR	At least one of the flag bits are set before `end_time`.
OS_TIMEOUT	None of the flag bits are set before `end_time`.

See also

NBRTOS Error Codes, PendAnyNoWait(), PendAll()

23.51.3.7 PendAny() [1/2]

```
uint8_t OS_FLAGS::PendAny (
            uint32_t bit_mask,
            TickTimeout & timeout )
```

Wait the specified number of TickTimeout time ticks for any of the flags in the bit mask to be set.

Parameters

bit_mask	A bit or set of bits to wait on.

Parameters

timeout	Number of time ticks to wait for all specified flag bits to be set. If not specified or 0 will wait forever.

Return values

OS_NO_ERR	At least one of the flag bits are set before `timeout` expires. A value of 0 waits forever.
OS_TIMEOUT	None of the flag bits are set before `timeout` expires.

See also

NBRTOS Error Codes, PendAnyNoWait(), PendAll()

23.51.3.8 PendAny() [2/2]

```
uint8_t OS_FLAGS::PendAny (
            uint32_t bit_mask,
            uint16_t timeout = WAIT_FOREVER )  [inline]
```

Wait the specified number of system TimeTicks for any of the flags in the bit mask to be set.

Parameters

bit_mask	A bit or set of bits to wait on.
timeout	Number of time ticks to wait for all specified flag bits to be set. If not specified or 0 will wait forever.

Return values

OS_NO_ERR	At least one of the flag bits are set before `timeout` expires. A value of 0 waits forever.
OS_TIMEOUT	None of the flag bits are set before `timeout` expires.

See also

NBRTOS Error Codes, PendAnyNoWait(), PendAll()

23.51.3.9 PendAnyNoWait()

```
uint8_t OS_FLAGS::PendAnyNoWait (
            uint32_t bit_mask )
```

Check for the specified flags and return immediately.

Parameters

bit_mask	A bit or set of bits to wait on.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

PendAny(), PendAll()

23.51.3.10 PendAnyUntil()

```
uint8_t OS_FLAGS::PendAnyUntil (
            uint32_t bit_mask,
            uint32_t end_time )  [inline]
```

Wait until the specified system TimeTicks value for any of the flags in the bit mask to be set.

Parameters

bit_mask	A bit or set of bits to wait on.
end_time	System TimeTick value to wait for.

Return values

OS_NO_ERR	At least one of the flag bits are set before `end_time`.
OS_TIMEOUT	None of the flag bits are set before `end_time`.

See also

NBRTOS Error Codes, PendAnyNoWait(), PendAll()

23.51.3.11 Set()

```
void OS_FLAGS::Set (
            uint32_t bits_to_set )
```

Sets the specified flag bits.

Parameters

bits_to_set	A bit or set of bits to be set

See also

Clear(), State()

23.51.3.12 State()

```
uint32_t OS_FLAGS::State ( )
```

Returns the current values of the flags stored in the OS_FLAGS object.

Returns

The state of the OS_FLAGS object

See also

Set(), Clear(), Write()

23.51.3.13 Write()

```
void OS_FLAGS::Write (
            uint32_t bits_to_force )
```

Set the flag bits to match the specified value.

Parameters

bits_to_force	Value to set all flags to

See also

Set(), Clear(), State()

The documentation for this struct was generated from the following file:

- nbrtos.h

23.52 OS_MBOX Struct Reference

Mailboxes single value storage locations used to communicate between tasks.
`#include <nbrtos.h>`
Inherits OS_TASK_DLY_OBJ.

Public Member Functions

- OS_MBOX ()
  Create and initialize a mailbox object with no defalut message.
- OS_MBOX (void *msg)
  Create and initialize a mailbox object with the specified message.
- uint8_t Init (void *msg=NULL)
  Sets the mailbox object to its initial state.
- uint8_t Post (void *msg)
  Post a message to the mailbox.
- void * Pend (uint32_t timeoutTicks, uint8_t &result)
  Wait the specified number of time ticks for some other task to post to the mailbox.
- void * Pend (TickTimeout &t, uint8_t &result)
  Wait the specified number of TickTimeout ticks for some other task to post to the mailbox.
- void * Pend (uint32_t timeoutTicks=WAIT_FOREVER)
  Wait the specified number of time ticks for some other task to post to the mailbox.
- void * PendUntil (uint32_t timeoutTime, uint8_t &result)
  Wait the specified number of TimeTicks for some other task to post to the mailbox.
- void * PendNoWait (uint8_t &result)
  Checks if a message is available in the mailbox and returns immediately.
- void * PendNoWait ()
  Checks if a message is available in the mailbox and returns immediately.

23.52.1 Detailed Description

Mailboxes single value storage locations used to communicate between tasks.
A mailbox is a storage location large enough to hold a single void pointer value, access to which is controlled so that it may be safely utilized by multiple tasks. If a task writes to a mailbox, it is then full and no task can send/post to it until a task does a read/pend on the mailbox or the mailbox is reset.
The void pointer can represent anything, such as a pointer to a variable, object, string, or just used as a 32-bit value. How it is used is defined by the tasks that post and pend to it.

Note

A "Message" is defined as any value other than null.

23.52.2 Constructor & Destructor Documentation

23.52.2.1 OS_MBOX() [1/2]

```
OS_MBOX::OS_MBOX ( )  [inline]
```

Create and initialize a mailbox object with no defalut message.

See also

Init()

23.52.2.2 OS_MBOX() [2/2]

```
OS_MBOX::OS_MBOX (
            void * msg )  [inline]
```

Create and initialize a mailbox object with the specified message.

Parameters

msg	Message that the mailbox should be initialized with

See also

Init()

23.52.3 Member Function Documentation

23.52.3.1 Init()

```
uint8_t OS_MBOX::Init (
            void * msg = NULL )
```

Sets the mailbox object to its initial state.
If called without a msg parameter, it will default to a value of null.

Parameters

msg	Message that the mailbox should be initialized with

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

23.52.3.2 Pend() [1/3]

```
void * OS_MBOX::Pend (
            TickTimeout & t,
            uint8_t & result )
```

Wait the specified number of TickTimeout ticks for some other task to post to the mailbox.

Parameters

t	The number of TicksTimeout ticks to wait. 0 will wait forever.
result	Reference to store the status of the function call, NBRTOS Error Codes. `OS_NO_ERR` if successful, `OS_TIMEOUT` if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.52.3.3 Pend() [2/3]

```
void * OS_MBOX::Pend (
            uint32_t timeoutTicks,
            uint8_t & result )  [inline]
```

Wait the specified number of time ticks for some other task to post to the mailbox.

Parameters

timeoutTicks	The number of system ticks to wait. 0 will wait forever.
result	Reference to store the status of the function call, NBRTOS Error Codes. `OS_NO_ERR` if successful, `OS_TIMEOUT` if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.52.3.4 Pend() [3/3]

```
void * OS_MBOX::Pend (
            uint32_t timeoutTicks = WAIT_FOREVER )  [inline]
```

Wait the specified number of time ticks for some other task to post to the mailbox.

Parameters

timeoutTicks	The number of system ticks to wait. If no value is specified, it will wait forever.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.52.3.5 PendNoWait() [1/2]

```
void * OS_MBOX::PendNoWait ( )  [inline]
```

Checks if a message is available in the mailbox and returns immediately.

Return values

Message	If successful
NULL	If it timed out

See also

Pend(), PendUntil()

23.52.3.6 PendNoWait() [2/2]

```
void * OS_MBOX::PendNoWait (
            uint8_t & result )
```

Checks if a message is available in the mailbox and returns immediately.

Parameters

result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, Pend(), PendUntil()

23.52.3.7 PendUntil()

```
void * OS_MBOX::PendUntil (
            uint32_t timeoutTime,
            uint8_t & result )  [inline]
```

Wait the specified number of TimeTicks for some other task to post to the mailbox.

Parameters

timeoutTime	The value of system time ticks to wait
result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), Pend()

23.52.3.8 Post()

```
uint8_t OS_MBOX::Post (
            void * msg )
```

Post a message to the mailbox.

Parameters

msg	Message that the mailbox should post

Return values

OS_NO_ERR	If successful
OS_MBOX_FULL	If the mailbox is full

See also

NBRTOS Error Codes

The documentation for this struct was generated from the following file:

- nbrtos.h

23.53 OS_Q Struct Reference

A message queue is an object that enables tasks and interrupt service routines to pend and post pointer sized messages. The pointer values typically point to some type of object or structure that contains the actual message or data. A queue functions as a fixed size First In First Out (FIFO) storage for 32-bit void pointers that can be used for communication between tasks.

```
#include <nbrtos.h>
```

Inherits OS_TASK_DLY_OBJ.

Public Member Functions

- OS_Q (void **pQueueStorage, uint8_t size)

  Create and initialize a queue object.
- uint8_t Init (void **pQueueStorage, uint8_t size)

  Set the queue object to its initial state.
- uint8_t Post (void *pItem)

  Post a message to the next available location in the queue.
- uint8_t PostFirst (void *pItem)

  Post a message to the head of the queue.
- uint8_t PostUnique (void *pItem)

  Post the specified message to the next available location in the queue, but only if the message is unique and does not exist anywhere else in the queue.
- uint8_t PostUniqueFirst (void *msg)

  Post the specified message to the first location in the queue, but only if the message is unique and does not exist anywhere else in the queue.
- void * Pend (uint32_t timeoutTicks, uint8_t &result)

  Wait the specified number of time ticks for some other task to post to the queue.

- void * Pend (TickTimeout &t, uint8_t &result)

  Wait the specified number of TickTimeout ticks for some other task to post to the queue.
- void * Pend (uint32_t timeoutTicks=WAIT_FOREVER)

  Wait the specified number of time ticks for some other task to post to the queue.
- void * PendUntil (uint32_t timeoutTime, uint8_t &result)

  Wait the specified TimeTicks value for some other task to post to the queue.
- void * PendNoWait (uint8_t &result)

  Checks if a message is available in the queue and returns immediately.
- void * PendNoWait ()

  Checks if a message is available in the queue and returns immediately.

23.53.1 Detailed Description

A message queue is an object that enables tasks and interrupt service routines to pend and post pointer sized messages. The pointer values typically point to some type of object or structure that contains the actual message or data. A queue functions as a fixed size First In First Out (FIFO) storage for 32-bit void pointers that can be used for communication between tasks.
A queue can be thought of as an array of void pointers, unlike a mailbox which can store only one void pointer. Access is controlled so that the queue can be safely utilized by multiple tasks. Tasks write to the queue using a Post function, and read from the queue using a Pend function.
The void pointer can represent anything, such as a pointer to a variable, object, string, or just used as a 32-bit value. How it is used is defined by the tasks that post and pend to it.

Note

If you need to store more than a simple value as with this OS_Q class, please refer to the OS_FIFO class which can store structures.

23.53.2 Constructor & Destructor Documentation

23.53.2.1 OS_Q()

```
OS_Q::OS_Q (
            void ** pQueueStorage,
            uint8_t size )  [inline]
```

Create and initialize a queue object.
The queue storage must be allocated in your application. A pointer to the storage area is then passed to the queue constructor. For example,

```
#define MY_QUEUE_SIZE (20)          // Number of storage locations (void pointers) in the queue.
                                    //   In this case, 20.
void *myQueueData[MY_QUEUE_SIZE];   // Declare the array of void pointers and a pointer to the array
OS_Q myQ(myQueueData,MY_QUEUE_SIZE); // Create an instance of the queue
```

Parameters

pQueueStorage	A pointer to an array of (void *) pointers to hold queue messages.
size	The number of pointers the queue can store.

23.53.3 Member Function Documentation

23.53.3.1 Init()

```
uint8_t OS_Q::Init (
            void ** pQueueStorage,
```

```
            uint8_t size )
```

Set the queue object to its initial state.

Parameters

pQueueStorage	A pointer to an array of (void *) pointers to hold queue messages.
size	The number of pointers the queue can store.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

23.53.3.2 Pend() [1/3]

```
void * OS_Q::Pend (
            TickTimeout & t,
            uint8_t & result )
```

Wait the specified number of TickTimeout ticks for some other task to post to the queue.

Parameters

t	The number of system ticks to wait. A value of 0 will wait forever.
result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.53.3.3 Pend() [2/3]

```
void * OS_Q::Pend (
            uint32_t timeoutTicks,
            uint8_t & result )  [inline]
```

Wait the specified number of time ticks for some other task to post to the queue.

Parameters

timeoutTicks	The number of system ticks to wait. A value of 0 will wait forever.
result	Reference to store the status of the function call, NBRTOS Error Codes. OS_NO_ERR if successful, OS_TIMEOUT if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.53.3.4 Pend() [3/3]

```
void * OS_Q::Pend (
            uint32_t timeoutTicks = WAIT_FOREVER )  [inline]
```

Wait the specified number of time ticks for some other task to post to the queue.

Parameters

timeoutTicks	The number of system ticks to wait. If no value is specified, it will wait forever.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), PendUntil()

23.53.3.5 PendNoWait() [1/2]

```
void * OS_Q::PendNoWait ( )  [inline]
```

Checks if a message is available in the queue and returns immediately.

Return values

Message	If successful
NULL	If it timed out

See also

Pend(), PendUntil()

23.53.3.6 PendNoWait() [2/2]

```
void * OS_Q::PendNoWait (
            uint8_t & result )
```

Checks if a message is available in the queue and returns immediately.

Parameters

result	Reference to store the status of the function call, NBRTOS Error Codes. `OS_NO_ERR` if successful, `OS_TIMEOUT` if it timed out.

Return values

Message	If successful

Return values

NULL	If it timed out

See also

NBRTOS Error Codes, Pend(), PendUntil()

23.53.3.7 PendUntil()

```
void * OS_Q::PendUntil (
            uint32_t timeoutTime,
            uint8_t & result )  [inline]
```

Wait the specified TimeTicks value for some other task to post to the queue.

Parameters

timeoutTime	The value of system TimeTicks to wait for
result	Reference to store the status of the function call, NBRTOS Error Codes. `OS_NO_ERR` if successful, `OS_TIMEOUT` if it timed out.

Return values

Message	If successful
NULL	If it timed out

See also

NBRTOS Error Codes, PendNoWait(), Pend()

23.53.3.8 Post()

```
uint8_t OS_Q::Post (
            void * pItem )
```

Post a message to the next available location in the queue.

Parameters

pItem	Message (void pointer) to post

Return values

OS_NO_ERR	If successful
OS_Q_FULL	If the queue is full, NBRTOS Error Codes

See also

PostFirst(), PostUnique(), Pend()

23.53.3.9 PostFirst()

```
uint8_t OS_Q::PostFirst (
            void * pItem )
```

Post a message to the head of the queue.

Parameters

pItem	Message (void pointer) to post

Return values

OS_NO_ERR	If successful
OS_Q_FULL	If the queue is full, NBRTOS Error Codes

See also

Post(), PostUniqueFirst(), Pend()

23.53.3.10 PostUnique()

```
uint8_t OS_Q::PostUnique (
            void * pItem )
```

Post the specified message to the next available location in the queue, but only if the message is unique and does not exist anywhere else in the queue.

Parameters

pItem	Message (void pointer) to post

Return values

OS_NO_ERR	If successful
OS_Q_EXISTS	If the message already exists in the queue NBRTOS Error Codes
OS_Q_FULL	If the queue is full NBRTOS Error Codes

See also

Post(), PostFirst(), Pend()

23.53.3.11 PostUniqueFirst()

```
uint8_t OS_Q::PostUniqueFirst (
            void * msg )
```

Post the specified message to the first location in the queue, but only if the message is unique and does not exist anywhere else in the queue.

Parameters

msg	Message (void pointer) to post

Return values

OS_NO_ERR	If successful
OS_Q_EXISTS	If the message already exists in the queue NBRTOS Error Codes
OS_Q_FULL	If the queue is full NBRTOS Error Codes

See also

Post(), PostFirst(), Pend()

The documentation for this struct was generated from the following file:

- nbrtos.h

23.54 OS_SEM Struct Reference

Semaphores are used to control access to shared resources or or to communicate between tasks in a multithreaded system or with interrupt service routines. Semaphores can be 0 or 1, or they can be counting semaphores that increment and decreement based on calls to Pend() and Post() functions.

`#include <nbrtos.h>`

Inherits OS_TASK_DLY_OBJ.

Public Member Functions

- OS_SEM (int32_t cnt=0)

  Create and initialize a semaphore.
- uint8_t Init (int32_t cnt=0)

  Initialize the semaphore object count value.
- uint8_t Post ()

  Posts to the semaphore, increasing it's value by 1.
- uint8_t Pend (uint32_t timeoutTicks=WAIT_FOREVER)

  Wait timeout ticks for the value of the semaphore to be non zero.
- uint8_t PendUntil (uint32_t timeout_time)

  Wait until the specified timeout time for a task to post to the semaphore.
- uint8_t Pend (TickTimeout &t)

  Wait for the specified number of system time ticks for a task to post to the semaphore.
- uint8_t PendNoWait ()

  Pend on a semaphore with no waiting period.
- uint32_t Avail ()

  Returns the number of semaphore counts availble.

23.54.1 Detailed Description

Semaphores are used to control access to shared resources or or to communicate between tasks in a multithreaded system or with interrupt service routines. Semaphores can be 0 or 1, or they can be counting semaphores that increment and decreement based on calls to Pend() and Post() functions.

23.54.2 Constructor & Destructor Documentation

23.54.2.1 OS_SEM()

```
OS_SEM::OS_SEM (
            int32_t cnt = 0 )  [inline]
```

Create and initialize a semaphore.

Parameters

cnt	Starting semaphore count value. If not specified, default is 0.

See also

Init(int32_t cnt = 0)

23.54.3 Member Function Documentation

23.54.3.1 Avail()

```
uint32_t OS_SEM::Avail ( )  [inline]
```

Returns the number of semaphore counts availble.
Difference between the number of counts used and the total count value.

Returns

The number of available semaphore counts

See also

Pend(), Post()

23.54.3.2 Init()

```
uint8_t OS_SEM::Init (
            int32_t cnt = 0 )
```

Initialize the semaphore object count value.

Warning

This must be done before using the semaphore.

Parameters

cnt	Starting semaphore count value. If not specified, default is 0.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

23.54.3.3 Pend() [1/2]

```
uint8_t OS_SEM::Pend (
            TickTimeout & t )
```

Wait for the specified number of system time ticks for a task to post to the semaphore.

Parameters

t	The number of TickTiemout ticks to wait. A value of 0 will wait forever.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

PendNoWait(), PendUntil()

23.54.3.4 Pend() [2/2]

```
uint8_t OS_SEM::Pend (
            uint32_t timeoutTicks = WAIT_FOREVER )  [inline]
```

Wait timeout ticks for the value of the semaphore to be non zero.

Parameters

timeoutTicks	Number of system time ticks to wait. If not specified, will wait forever.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

PendNoWait(), PendUntil()

23.54.3.5 PendNoWait()

```
uint8_t OS_SEM::PendNoWait ( )
```

Pend on a semaphore with no waiting period.
Identical to the Pend() function, but it does not wait.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

Pend(), PendUntil()

23.54.3.6 PendUntil()

```
uint8_t OS_SEM::PendUntil (
            uint32_t timeout_time )  [inline]
```

Wait until the specified timeout time for a task to post to the semaphore.

Parameters

timeout_time	The system TimeTick value to wait until

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

PendNoWait(), Pend()

23.54.3.7 Post()

`uint8_t OS_SEM::Post ( )`

Posts to the semaphore, increasing it's value by 1.
If a higher priority task was pending on the semaphore it will be released.

Returns

OS_NO_ERR on success, otherwise NBRTOS Error Codes

See also

Pend(), PendUntil(), PendNoWait()

The documentation for this struct was generated from the following file:

- nbrtos.h

23.55 OSCriticalSectionObj Class Reference

A simple wrapper class that helps utilize OS_CRIT objects more effectively.
`#include <nbrtos.h>`

Public Member Functions

- OSCriticalSectionObj (OS_CRIT &ocrit)

  Initialize the OSCriticalSectionObj object, and then call Enter() on the OS_CRIT object that is passed in.
- OSCriticalSectionObj (OS_CRIT &ocrit, bool NoWait, TickTimeout &timeout)

  Initialize the OSCriticalSectionObj object, and then call Enter() on the OS_CRIT object that is passed in.
- ~**OSCriticalSectionObj** ()

  Destructs the OSCriticalSectionObj object, and call Leave() on the OS_CRIT object that was passed into the constructor.

23.55.1 Detailed Description

A simple wrapper class that helps utilize OS_CRIT objects more effectively.
When an OSCriticalSectionObj is constructed calls Enter() on the OS_CRIT object that is passed in, and will wait indefinitely to claim the section. When the object is deconstructed, it calls Leave() on the OS_CRIT object.

See also

OS_CRIT::Enter(), OS_CRIT::Leave()

23.55.2 Constructor & Destructor Documentation

23.55.2.1 OSCriticalSectionObj() [1/2]

```
OSCriticalSectionObj::OSCriticalSectionObj (
            OS_CRIT & ocrit )  [inline]
```

Initialize the OSCriticalSectionObj object, and then call Enter() on the OS_CRIT object that is passed in.

Parameters

ocrit	The object that will be used to enter and leave the critical section.

23.55.2.2 OSCriticalSectionObj() [2/2]

```
OSCriticalSectionObj::OSCriticalSectionObj (
            OS_CRIT & ocrit,
            bool NoWait,
            TickTimeout & timeout )  [inline]
```

Initialize the OSCriticalSectionObj object, and then call Enter() on the OS_CRIT object that is passed in.

Parameters

ocrit	The object that will be used to enter and leave the critical section.
NoWait	Set to true to enable the no waiting mode of operation.
timeout	Number of TickTimout time ticks to wait for the critical section.

The documentation for this class was generated from the following file:

- nbrtos.h

23.56 OSLockAndCritObj Class Reference

A simple wrapper class that helps utilize OS_CRIT objects to lock tasks and enter critical sections more effectively.
`#include <nbrtos.h>`

Public Member Functions

- OSLockAndCritObj (OS_CRIT &ocrit)

  Initialize the OSCriticalSectionObj object, and then call LockAndEnter() on the OS_CRIT object that is passed in.
- ~OSLockAndCritObj ()

  Call LeaveAndUnlock() on the OSCriticalSectionObj object, then destruct.

23.56.1 Detailed Description

A simple wrapper class that helps utilize OS_CRIT objects to lock tasks and enter critical sections more effectively. When an OSCriticalSectionObj is constructed it calls LockAndEnter() on the OS_CRIT object that is passed in, and will wait indefinitely to claim the section. When the object is deconstructed, it calls LeaveAndUnlock() on the OS_CRIT object.

See also

OS_CRIT::LockAndEnter(), OS_CRIT::LeaveAndUnlock()

23.56.2 Constructor & Destructor Documentation

23.56.2.1 OSLockAndCritObj()

```
OSLockAndCritObj::OSLockAndCritObj (
            OS_CRIT & ocrit )  [inline]
```

Initialize the OSCriticalSectionObj object, and then call LockAndEnter() on the OS_CRIT object that is passed in.

Parameters

ocrit	The object that will be used to enter and leave the critical section.

23.56.2.2 ~OSLockAndCritObj()

`OSLockAndCritObj::~OSLockAndCritObj ( ) [inline]`
Call LeaveAndUnlock() on the OSCriticalSectionObj object, then destruct.
The documentation for this class was generated from the following file:

- nbrtos.h

23.57 OSLockObj Class Reference

A simple wrapper class that helps use OS locks effectively.
`#include <nbrtos.h>`

Public Member Functions

- **OSLockObj** ()

  Initialize the OSLockObj and calls OSLock().
- ~**OSLockObj** ()

  Destructs the OSLockObj and calls OSUnlock().

23.57.1 Detailed Description

A simple wrapper class that helps use OS locks effectively.
When an OSLockObj is constructed it locks the OS. When it is destructed it unlocks the OS. If you have a function that needs an OS lock and has multiple points of exit, create an OSLockObj at the beginning of the function.

Note

No matter how you leave the function, the destructor will release the lock.

See also

OSLock(), OSUnlock()

The documentation for this class was generated from the following file:

- nbrtos.h

23.58 OSSpinCrit Class Reference

A simple wrapper class that uses an OS_CRIT object to try and claim a critical section, and will continue the attempt until it is able to do so.
`#include <nbrtos.h>`

Public Member Functions

- OSSpinCrit (OS_CRIT &ocrit)

  Initialize the OSSpinCrit object, and then call EnterNoWait() repeatedly on the OS_CRIT object that is passed in until it is able to claim the critical section.
- ~OSSpinCrit ()

  Call Leave() on the OS_CRIT object, and then destruct/.

23.58.1 Detailed Description

A simple wrapper class that uses an OS_CRIT object to try and claim a critical section, and will continue the attempt until it is able to do so.
The way this class is implemented allows the task to continue to be made active by the scheduler. When an OSSpinCrit is constructed, it calls EnterNoWait() repeatedly on the OS_CRIT object passed in until it is able to claim the critical section. When the object is deconstructed, it calls Leave() on the OS_CRIT object.

See also

OS_CRIT::EnterNoWait(), OS_CRIT::Leave()

23.58.2 Constructor & Destructor Documentation

23.58.2.1 OSSpinCrit()

```
OSSpinCrit::OSSpinCrit (
            OS_CRIT & ocrit ) [inline]
```

Initialize the OSSpinCrit object, and then call EnterNoWait() repeatedly on the OS_CRIT object that is passed in until it is able to claim the critical section.

Parameters

&ocrit	The object that will be used to enter and leave the critical section.

23.58.2.2 ~OSSpinCrit()

```
OSSpinCrit::~OSSpinCrit ( ) [inline]
```

Call Leave() on the OS_CRIT object, and then destruct/.
The documentation for this class was generated from the following file:

- nbrtos.h

23.59 ParsedJsonDataSet Class Reference

A class to create, read, and modify a JSON object.
`#include <json_lexer.h>`
Inherits buffer_object.
Inherited by AcmeJsonBuffer, and AcmeServletBuffer.

Public Member Functions

Parsing

Ingesting raw JSON data from some source, or resetting. Enables compatibility with buffer_object and Web↩ Client.

- void EnableLargeStrings (bool b)
  Call to allow allocation of strings larger than 1500 bytes.
- virtual int WriteData (const unsigned char *pCopyFrom, int numBytes)
  Writes the passed in data to the JSON data set.
- virtual int ReadFrom (int fd)
  Reads in data from the specified file descriptor and parses it into the JSON data set.
- bool CopyObject (ParsedJsonDataSet &src_set)
  Copies the provided JSON data set into the current one.
- void ClearObject ()

Clears all data from the object and resets it to its initial state.

Querying the Data Structure

Allows easy retrieval of entities by array index or object key name

Example

```
printf("User %d name %s",
  (int)myJsonRef("users")[0]("id"),
  (const char *)myJsonRef("users")[0]("name")
);
```

- JsonRef operator[ ] (int i)

  Get a JsonRef from an array by numerical index.
- JsonRef operator() (const char *name)

  Get a JsonRef representing any entity from a parent object by key name.
- JsonRef name (const char *name)

  Get a JsonRef representing any entity from a parent object by key name.
- JsonRef object (const char *name)

  Get a JsonRef representing an object from a parent object by key name.

Traversing the Data Structure

Used for scanning through the JSON elements

- JsonRef start ()

  Begin traversal: check for ref validity and reset the position.
- JsonRef next ()

  Traverse: check for ref validity and increment the position.
- json_primitive_type GetFirst ()

  Get the first element of the JSON data set.
- json_primitive_type GetNext ()

  Get the element at the next position of the JSON data set.
- json_primitive_type GetCurrent ()

  Get the element at the current position of the JSON data set.
- json_primitive_type GetRawCurrent ()

  Get the element at the current position of the JSON data set, including non-public types.
- json_primitive_type GetNextName ()

  Iterates the current element of the JSON data set until it finds one of type NAME.
- json_primitive_type GetNextObject ()

  Get the next element of type name that exists in the current OBJECT after the current position.
- json_primitive_type GetNextArray ()

  Iterates the current element of the JSON data set until it finds one of type BEGIN_ARRAY.
- json_primitive_type GetNextNameInCurrentObject ()

  Get the next element of type NAME that exists in the current OBJECT after the current position.
- json_primitive_type GetNextNameInCurrentArray ()

  Get the next element of type NAME that exists in the current ARRAY after the current position.
- json_primitive_type GetNextNumberInCurrentArray ()

  Get the next element of type NUMBER that exists in the current ARRAY after the current position.
- json_primitive_type GetNextStringInCurrentArray ()

  Get the next element of type STRING that exists in the current ARRAY after the current position.
- json_primitive_type GetNextBoolInCurrentArray ()

  Get the next element of type TRUE_EL that exists in the current ARRAY after the current position.
- json_primitive_type GetNextObjectInCurrentArray ()

  Get the next element of type BEGIN_OBJECT that exists in the current ARRAY after the current position.
- json_primitive_type SkipCurrentValue ()

  Skips over the current value, and get the next element. If called inside an ARRAY or OBJECT, it will walk to the end of the ARRAY or OJBECT and return the next element it finds. If it reaches the end of the ARRAY or OBJECT before it finds a value, it will return NOTFOUND.
- void ResetPosition ()

  Resets the parser position to the beginning of the JSON data set.
- JsonRef GetParsePosition ()

Gets the current parse position object for the JSON data set.

- JsonRef SetParsePosition (JsonRef pos)

  Sets the current parse position object for the JSON data set.

Advanced Finding and Returning

Advanced functions for scanning and returning specific objects. All of these can return NOTFOUND.

Example

This function works with dotted string notation to find sub elements:

```
{
    "SUCCESS": true,
    "ACTION": null,
    "JDATA": {
        "foo": 7,
        "bar": {
            "b1": "Sb1",
            "b2": {
                "s1": "Sub1",
                "s2": "Sub2",
                "s3": "Sub3",
                "s4": {
                    "Sub4a": "Test4a",
                    "Sub4b": "Test4b"
                },
                "s5": "Sub5",
                "s6": "Sub6"
            },
            "b3": true,
            "b4": null
        },
        "bogus": "Bstring",
        "nubog": 1234.57,
        "av": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9,
            10
        ],
        "aftaav": true
    }
}
```

You could return the "`Sub4b`" element's "`Test4b`" string with:

```
myParsedJsonDataSet.FindFullName("JDATA.bar.b2.s4.Sub4b");
```

- json_primitive_type FindFullName (const char *name)

  Find the element in the data set with the given name and move the parser to the next element. This searches a full name path, such as ob1.ob2.ob3.ob4.

- json_primitive_type FindFullAtName (const char *name)

  Find the element in the data set with the given name and move the parser to that element. This searches a full name path, such as ob1.ob2.ob3.ob4.

- json_primitive_type FindElementAfterName (const char *name)

  Finds name in current object points at element after name. This only supports simple, single element names. This searches from the current position to the end.

- json_primitive_type FindGlobalElementAfterName (const char *name)

  Finds name in current object points at element after name. This only supports simple, single element names. It starts the search from the parser's current position, and if it doesn't find it, it will start the search over from the beginning.

- json_primitive_type FindElementAfterNameInCurrentObject (const char *name)

  Looks for elements with the current name in the current OBJECT only. Does not search sub objects.

- json_primitive_type FindElementAfterNameInCurrentArray (const char *name)

  Looks for elements with the current name in the current ARRAY only. Does not search sub arrays.

- bool CurrentBool ()

  Returns true if the current element a TRUE_EL primitive type.

- bool PermissiveCurrentBool ()

  Returns true if the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.

- double CurrentNumber ()

Get the number value of the current element.

- const char * CurrentString ()

  Get the string value of the current element.
- const char * CurrentName ()

  Get the name of the current element.
- bool FindFullNameBoolean (const char *name)

  Find the boolean value of a given element.
- bool FindGlobalBoolean (const char *name)

  Find the boolean value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.
- bool FindBoolean (const char *name)

  Find the boolean value of the element after the element with the given name.
- bool FindBooleanInCurentObject (const char *name)

  Find the boolean value of the element within the current object. Does not search sub-objects.
- bool FindFullNamePermissiveBoolean (const char *name)

  Find the permissive boolean value of a given element.
- bool FindGlobalPermissiveBoolean (const char *name)

  Find the permissive boolean value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.
- bool FindPermissiveBoolean (const char *name)

  Find the permissive boolean value of the element after the element with the provided name.
- bool FindPermissiveBooleanInCurentObject (const char *name)

  Find the permissive boolean value of the element with the given name in the current object. Does not search sub-objects.
- const char * FindFullNameString (const char *name)

  Find the string value of the element with the given name in the current object.
- const char * FindGlobalString (const char *name)

  Find the string value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.
- const char * FindString (const char *name)

  Find the string value of the element after the element with the given name.
- const char * FindStringInCurentObject (const char *name)

  Find the string value of the element with the given name in the current object. Does not search sub-objects.
- double FindFullNameNumber (const char *name)

  Find the number value of the element with the given name in the current object.
- double FindGlobalNumber (const char *name)

  Find the number value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.
- double FindNumber (const char *name)

  Find the number value of the element after the element with the given name.
- double FindNumberInCurentObject (const char *name)

  Find the number value of the element with the given name in the current object. Does not search sub-objects.
- bool FindGlobalObject (const char *name)

  Find the OBJECT with the given name. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.
- bool FindObject (const char *name)

  Find the OBJECT of the element after the element with the given name.
- bool FindObjectInCurentObject (const char *name)

  Find the OBJECT of the element with the given name in the current OBJECT. Does not search sub-objects.

Utility

Ingesting raw JSON data from some source, or resetting

- void DumpState ()

  Outputs what's in the parse tree to stdout.
- int PrintObject (bool pretty=false)

  Prints the JSON data set to stdout.
- int PrintObjectToBuffer (char *buffer, int maxlen, bool pretty=false)

  Prints the JSON data set to a provided buffer.
- int PrintObjectToFd (int fd, bool pretty=false)

Prints the JSON data set to a specified file descriptor.

- int PrintChildren (bool pretty=false)

  Output child objects of this JsonRef as JSON text to stdout.
- int PrintChildrenToFd (int fd, bool pretty=false)

  Output child objects of this JsonRef as JSON text to a file descriptor.
- int PrintChildrenToBuffer (char *buffer, int maxlen, bool pretty=false)

  Output child objects of this JsonRef as JSON text to a buffer pointer.
- int GetPrintSize (bool pretty=false)

  Calculates how many characters the JSON data set would take to print.

Building

Programmatically creating or modifying a JSON dataset

- void StartBuilding ()

  Use to start building the JSON data set.
- void AddObjectStart (const char *name)

  Use to start an object in the JSON data set.
- void AddMyMac (const char *name)

  Add the device MAC address to the JSON data set.
- void Add (const char *name, int i)

  Add a name/value pair to the JSON data set where the value is an int.
- void Add (const char *name, short i)

  Add a name/value pair to the JSON data set where the value is a short.
- void Add (const char *name, long i)

  Add a name/value pair to the JSON data set where the value is a long.
- void Add (const char *name, unsigned int i)

  Add a name/value pair to the JSON data set where the value is an unsigned int.
- void Add (const char *name, unsigned short i)

  Add a name/value pair to the JSON data set where the value is a unsigned short.
- void Add (const char *name, unsigned long i)

  Add a name/value pair to the JSON data set where the value is a unsigned long.
- void Add (const char *name, double d)

  Add a name/value pair to the JSON data set where the value is a double.
- void Add (const char *name, const char *str)

  Add a name/value pair to the JSON data set where the value is a string.
- void Add (const char *name, bool b)

  Add a name/value pair to the JSON data set where the value is a bool.
- void Add (const char *name, IPADDR4 i4)

  Add a name/value pair to the JSON data set where the value is an IPv4 address.
- void Add (const char *name, const IPADDR &i)

  Add a name/value pair to the JSON data set where the value is an IP address.
- void Add (const char *name, const MACADR &ma)

  Add a name/value pair to the JSON data set where the value is a MAC address.
- void AddNull (const char *name)

  Add a name/value pair to the JSON data set where the value is null.
- void AddArrayStart (const char *name)

  Add an ARRAY start to the JSON data set.
- void EndArray ()

  Add an ARRAY end to the JSON data set.
- void AddArrayElement (int i)

  Add an integer value to the current array.
- void AddArrayElement (short i)

  Add a short value to the current array.
- void AddArrayElement (long i)

  Add a long value to the current array.
- void AddArrayElement (unsigned int i)

  Add an unsigned int value to the current array.
- void AddArrayElement (unsigned short i)

  Add an unsigned short value to the current array.

- void AddArrayElement (unsigned long i)
  Add an unsigned long value to the current array.
- void AddArrayElement (double d)
  Add a double to the current array.
- void AddArrayElement (const char *str)
  Add a string to the current array.
- void AddArrayElement (bool b)
  Add a bool to the current array.
- void AddArrayElement (const IPADDR &i)
  Add an IP address to the current array.
- void AddArrayElementArray ()
  Add the start of an array element to the current array.
- void AddArrayObjectStart ()
  Add the start of an object element to the current array.
- void AddNullArrayElement ()
  Add a null element to the current array.
- void EndObject ()
  Add an end to the current object.
- void DoneBuilding ()
  Add an end JSON data set and finish building.

Friends

- class **JsonRef**

23.59.1 Detailed Description

A class to create, read, and modify a JSON object.
Many of the methods on ParsedJsonDataSet return a JsonRef which is a pointer to a variable or sub object inside the ParsedJsonDataSet. So, once data is parsed and traversal or querying has begun most of the work will be inside the JsonRef class.

23.59.2 Member Function Documentation

23.59.2.1 Add() [1/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            bool b )
```

Add a name/value pair to the JSON data set where the value is a bool.

Parameters

name	The name of the pair.
b	The value for the pair.

23.59.2.2 Add() [2/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            const char * str )
```

Add a name/value pair to the JSON data set where the value is a string.

Parameters

name	The name of the pair.
str	A pointer to the value for the pair.

23.59.2.3 Add() [3/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            const IPADDR & i )
```

Add a name/value pair to the JSON data set where the value is an IP address.

Parameters

name	The name of the pair.
i	The IP address to add.

23.59.2.4 Add() [4/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            const MACADR & ma )
```

Add a name/value pair to the JSON data set where the value is a MAC address.

Parameters

name	The name of the pair.
ma	The MAC address to add.

23.59.2.5 Add() [5/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            double d )
```

Add a name/value pair to the JSON data set where the value is a double.

Parameters

name	The name of the pair.
d	The value for the pair.

23.59.2.6 Add() [6/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            int i )
```

Add a name/value pair to the JSON data set where the value is an int.

Parameters

name	The name of the pair.
i	The value for the pair.

23.59.2.7 Add() [7/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            IPADDR4 i4 )
```

Add a name/value pair to the JSON data set where the value is an IPv4 address.

Parameters

name	The name of the pair.
i4	The IPv4 address to add.

23.59.2.8 Add() [8/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            long i )
```

Add a name/value pair to the JSON data set where the value is a long.

Parameters

name	The name of the pair.
i	The value for the pair.

23.59.2.9 Add() [9/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            short i )
```

Add a name/value pair to the JSON data set where the value is a short.

Parameters

name	The name of the pair.
i	The value for the pair.

23.59.2.10 Add() [10/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            unsigned int i )
```

Add a name/value pair to the JSON data set where the value is an unsigned int.

Parameters

name	The name of the pair.
i	The value for the pair.

23.59.2.11 Add() [11/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            unsigned long i )
```

Add a name/value pair to the JSON data set where the value is a unsigned long.

Parameters

name	The name of the pair.
i	The value for the pair.

23.59.2.12 Add() [12/12]

```
void ParsedJsonDataSet::Add (
            const char * name,
            unsigned short i )
```

Add a name/value pair to the JSON data set where the value is a unsigned short.

Parameters

name	The name of the pair.
i	The value for the pair.

23.59.2.13 AddArrayElement() [1/10]

```
void ParsedJsonDataSet::AddArrayElement (
            bool b )
```

Add a bool to the current array.

Parameters

b	The bool value to add.

23.59.2.14 AddArrayElement() [2/10]

```
void ParsedJsonDataSet::AddArrayElement (
            const char * str )
```

Add a string to the current array.

Parameters

str	A char* pointing to the string value to add.

23.59.2.15 AddArrayElement() [3/10]

```
void ParsedJsonDataSet::AddArrayElement (
            const IPADDR & i )
```

Add an IP address to the current array.

Parameters

i	The IP address to add.

23.59.2.16 AddArrayElement() [4/10]

```
void ParsedJsonDataSet::AddArrayElement (
            double d )
```

Add a double to the current array.

Parameters

d	The double value to add.

23.59.2.17 AddArrayElement() [5/10]

```
void ParsedJsonDataSet::AddArrayElement (
            int i )
```

Add an integer value to the current array.

Parameters

i	The integer value to add.

23.59.2.18 AddArrayElement() [6/10]

```
void ParsedJsonDataSet::AddArrayElement (
            long i )
```

Add a long value to the current array.

Parameters

i	The long value to add.

23.59.2.19 AddArrayElement() [7/10]

```
void ParsedJsonDataSet::AddArrayElement (
            short i )
```

Add a short value to the current array.

Parameters

i	The short value to add.

23.59.2.20 AddArrayElement() [8/10]

```
void ParsedJsonDataSet::AddArrayElement (
            unsigned int i )
```

Add an unsigned int value to the current array.

Parameters

i	The unsigned int value to add.

23.59.2.21 AddArrayElement() [9/10]

```
void ParsedJsonDataSet::AddArrayElement (
            unsigned long i )
```

Add an unsigned long value to the current array.

Parameters

i	The unsigned long value to add.

23.59.2.22 AddArrayElement() [10/10]

```
void ParsedJsonDataSet::AddArrayElement (
            unsigned short i )
```

Add an unsigned short value to the current array.

Parameters

i	The unsigned short value to add.

23.59.2.23 AddArrayElementArray()

```
void ParsedJsonDataSet::AddArrayElementArray ( )
```

Add the start of an array element to the current array.

23.59.2.24 AddArrayObjectStart()

```
void ParsedJsonDataSet::AddArrayObjectStart ( )
```

Add the start of an object element to the current array.

23.59.2.25 AddArrayStart()

```
void ParsedJsonDataSet::AddArrayStart (
            const char * name )
```

Add an ARRAY start to the JSON data set.

Parameters

name	The name of the array to start.

23.59.2.26 AddMyMac()

```
void ParsedJsonDataSet::AddMyMac (
            const char * name )
```

Add the device MAC address to the JSON data set.

Parameters

name	The name of the field containing the MAC address.

23.59.2.27 AddNull()

```
void ParsedJsonDataSet::AddNull (
            const char * name )
```

Add a name/value pair to the JSON data set where the value is null.

Parameters

name	The name of the pair.

23.59.2.28 AddNullArrayElement()

```
void ParsedJsonDataSet::AddNullArrayElement ( )
```

Add a null element to the current array.

23.59.2.29 AddObjectStart()

```
void ParsedJsonDataSet::AddObjectStart (
            const char * name )
```

Use to start an object in the JSON data set.

Parameters

name	The name of the object to start.

23.59.2.30 ClearObject()

```
void ParsedJsonDataSet::ClearObject ( )
```

Clears all data from the object and resets it to its initial state.

23.59.2.31 CopyObject()

```
bool ParsedJsonDataSet::CopyObject (
            ParsedJsonDataSet & src_set )
```

Copies the provided JSON data set into the current one.

Return values

true	The copy succeeded.

Return values

false	The copy failed.

23.59.2.32 CurrentBool()

```
bool ParsedJsonDataSet::CurrentBool ( )  [inline]
```

Returns true if the current element a TRUE_EL primitive type.

Return values

true	If the current element is a TRUE_EL primitive type.
false	If the current element is not a TRUE_EL primitive type.

23.59.2.33 CurrentName()

```
const char * ParsedJsonDataSet::CurrentName ( )  [inline]
```

Get the name of the current element.

Returns

A pointer to the name of the current element, if it has one. If it does not, it returns a 0.

23.59.2.34 CurrentNumber()

```
double ParsedJsonDataSet::CurrentNumber ( )  [inline]
```

Get the number value of the current element.

Returns

The number value if the current element's primitive type is NUMBER. Otherwise it returns a quite NAN.

23.59.2.35 CurrentString()

```
const char * ParsedJsonDataSet::CurrentString ( )  [inline]
```

Get the string value of the current element.

Returns

The a pointer to the string value of the current element if it is of primitive type STRING or ALLOC_STRING. Otherwise it returns a 0.

23.59.2.36 DoneBuilding()

```
void ParsedJsonDataSet::DoneBuilding ( )
```

Add an end JSON data set and finish building.

23.59.2.37 DumpState()

```
void ParsedJsonDataSet::DumpState ( )
```

Outputs what's in the parse tree to stdout.

23.59.2.38 EnableLargeStrings()

```
void ParsedJsonDataSet::EnableLargeStrings (
            bool b )  [inline]
```

Call to allow allocation of strings larger than 1500 bytes.

23.59.2.39 EndArray()

```
void ParsedJsonDataSet::EndArray ( )
```

Add an ARRAY end to the JSON data set.

23.59.2.40 EndObject()

```
void ParsedJsonDataSet::EndObject ( )
```

Add an end to the current object.

23.59.2.41 FindBoolean()

```
bool ParsedJsonDataSet::FindBoolean (
            const char * name )  [inline]
```

Find the boolean value of the element after the element with the given name.

Parameters

name	The name of the element to find.

Return values

true	If the element is of the TRUE_EL primitive type.
false	If the element is not of the TRUE_EL primitive type.

23.59.2.42 FindBooleanInCurentObject()

```
bool ParsedJsonDataSet::FindBooleanInCurentObject (
            const char * name )  [inline]
```

Find the boolean value of the element within the current object. Does not search sub-objects.

Parameters

name	The name of the element to find.

Return values

true	If the element is of the TRUE_EL primitive type.
false	If the element is not of the TRUE_EL primitive type.

23.59.2.43 FindElementAfterName()

```
json_primitive_type ParsedJsonDataSet::FindElementAfterName (
```

```
            const char * name )  [inline]
```

Finds name in current object points at element after name. This only supports simple, single element names. This searches from the current position to the end.

Parameters

name	The name to look for.

Returns

The type that the parser is now currently set to.

Return values

NOTFOUND	If the an element with the name wasn't found.

23.59.2.44 FindElementAfterNameInCurrentArray()

```
json_primitive_type ParsedJsonDataSet::FindElementAfterNameInCurrentArray (
            const char * name )  [inline]
```

Looks for elements with the current name in the current ARRAY only. Does not search sub arrays.

Parameters

name	The name to look for.

Returns

The type that the parser is now currently set to.

Return values

NOTFOUND	If the an element with the name wasn't found.

23.59.2.45 FindElementAfterNameInCurrentObject()

```
json_primitive_type ParsedJsonDataSet::FindElementAfterNameInCurrentObject (
            const char * name )  [inline]
```

Looks for elements with the current name in the current OBJECT only. Does not search sub objects.

Parameters

name	The name to look for.

Returns

The type that the parser is now currently set to.

Return values

NOTFOUND	If the an element with the name wasn't found.

23.59.2.46 FindFullAtName()

```
json_primitive_type ParsedJsonDataSet::FindFullAtName (
            const char * name )  [inline]
```

Find the element in the data set with the given name and move the parser to that element. This searches a full name path, such as ob1.ob2.ob3.ob4.

Parameters

name	The full name to look for.

Returns

The type that the parser is now currently set to.

Return values

NOTFOUND	If the an element with the name wasn't found.

23.59.2.47 FindFullName()

```
json_primitive_type ParsedJsonDataSet::FindFullName (
            const char * name )  [inline]
```

Find the element in the data set with the given name and move the parser to the next element. This searches a full name path, such as ob1.ob2.ob3.ob4.

Parameters

name	The full name to look for.

Returns

The type that the parser is now currently set to.

Return values

NOTFOUND	If the an element with the name wasn't found, or there is no element after the named element.

23.59.2.48 FindFullNameBoolean()

```
bool ParsedJsonDataSet::FindFullNameBoolean (
            const char * name )  [inline]
```

Find the boolean value of a given element.

Parameters

name	The full name of the element to find.

Return values

true	If the element is of the TRUE_EL primitive type.
false	If the element is not of the TRUE_EL primitive type.

23.59.2.49 FindFullNameNumber()

```
double ParsedJsonDataSet::FindFullNameNumber (
            const char * name )  [inline]
```

Find the number value of the element with the given name in the current object.

Parameters

name	The full name of the element to find. •

Returns

The number value of the element if found, otherwise a quiet nan.

23.59.2.50 FindFullNamePermissiveBoolean()

```
bool ParsedJsonDataSet::FindFullNamePermissiveBoolean (
            const char * name )  [inline]
```

Find the permissive boolean value of a given element.

Parameters

name	The full name of the element to find.

Return values

true	If the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.
false	If the current element is not a TRUE_EL primitive type, "True", "true", or is 0.

23.59.2.51 FindFullNameString()

```
const char * ParsedJsonDataSet::FindFullNameString (
            const char * name )  [inline]
```

Find the string value of the element with the given name in the current object.

Parameters

name	The full name of the element to find. •

Returns

A pointer to the buffer if the string is found, or null otherwise.

23.59.2.52 FindGlobalBoolean()

```
bool ParsedJsonDataSet::FindGlobalBoolean (
            const char * name )  [inline]
```

Find the boolean value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.

Parameters

name	The name of the element to find.

Return values

true	If the element is of the TRUE_EL primitive type.
false	If the element is not of the TRUE_EL primitive type.

23.59.2.53 FindGlobalElementAfterName()

```
json_primitive_type ParsedJsonDataSet::FindGlobalElementAfterName (
            const char * name )  [inline]
```

Finds name in current object points at element after name. This only supports simple, single element names. It starts the search from the parser's current position, and if it doesn't find it, it will start the search over from the beginning.

Parameters

name	The name to look for.

Returns

The type that the parser is now currently set to.

Return values

NOTFOUND	If the an element with the name wasn't found.

23.59.2.54 FindGlobalNumber()

```
double ParsedJsonDataSet::FindGlobalNumber (
            const char * name )  [inline]
```

Find the number value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.

Parameters

name	The name of the element to find. •

Returns

The number value of the element if found, otherwise a quiet nan.

23.59.2.55 FindGlobalObject()

```
bool ParsedJsonDataSet::FindGlobalObject (
            const char * name )  [inline]
```

Find the OBJECT with the given name. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.

Parameters

name	The name of the OBJECT to find. •

Return values

true	If the OBJECT was found. The current position will be set to the start of the OBJECT.
false	If the OBJECT was not found.

23.59.2.56 FindGlobalPermissiveBoolean()

```
bool ParsedJsonDataSet::FindGlobalPermissiveBoolean (
            const char * name )  [inline]
```

Find the permissive boolean value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.

Parameters

name	The name of the element to find.

Return values

true	If the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.
false	If the current element is not a TRUE_EL primitive type, "True", "true", or is 0.

23.59.2.57 FindGlobalString()

```
const char * ParsedJsonDataSet::FindGlobalString (
            const char * name )  [inline]
```

Find the string value of a given element. Starts at the current position and then starts again at the beginning of the data set if the element isn't found. This supports only simple, single element names.

Parameters

name	The name of the element to find.

Returns

A pointer to the buffer if the string is found, or null otherwise.

23.59.2.58 FindNumber()

```
double ParsedJsonDataSet::FindNumber (
            const char * name )  [inline]
```

Find the number value of the element after the element with the given name.

Parameters

name	The name of the element to find.

Returns

The number value of the element if found, otherwise a quiet nan.

23.59.2.59 FindNumberInCurentObject()

```
double ParsedJsonDataSet::FindNumberInCurentObject (
            const char * name )  [inline]
```

Find the number value of the element with the given name in the current object. Does not search sub-objects.

Parameters

name	The name of the element to find. •

Returns

The number value of the element if found, otherwise a quiet nan.

23.59.2.60 FindObject()

```
bool ParsedJsonDataSet::FindObject (
            const char * name )  [inline]
```

Find the OBJECT of the element after the element with the given name.

Parameters

name	The name of the OBJECT to find.

Return values

true	If the OBJECT was found. The current position will be set to the start of the OBJECT.
false	If the OBJECT was not found.

23.59.2.61 FindObjectInCurentObject()

```
bool ParsedJsonDataSet::FindObjectInCurentObject (
            const char * name )  [inline]
```

Find the OBJECT of the element with the given name in the current OBJECT. Does not search sub-objects.

Parameters

name	The name of the OBJECT to find.

Return values

true	If the OBJECT was found. The current position will be set to the start of the OBJECT.
false	If the OBJECT was not found.

23.59.2.62 FindPermissiveBoolean()

```
bool ParsedJsonDataSet::FindPermissiveBoolean (
            const char * name )  [inline]
```

Find the permissive boolean value of the element after the element with the provided name.

Parameters

name	The name of element to search on.

Return values

true	If the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.
false	If the current element is not a TRUE_EL primitive type, "True", "true", or is 0.

23.59.2.63 FindPermissiveBooleanInCurentObject()

```
bool ParsedJsonDataSet::FindPermissiveBooleanInCurentObject (
            const char * name )  [inline]
```

Find the permissive boolean value of the element with the given name in the current object. Does not search sub-objects.

Parameters

name	The name of the element to find. •

Return values

true	If the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.
false	If the current element is not a TRUE_EL primitive type, "True", "true", or is 0.

23.59.2.64 FindString()

```
const char * ParsedJsonDataSet::FindString (
            const char * name )  [inline]
```

Find the string value of the element after the element with the given name.

Parameters

name	The name of the element to find.

Returns

A pointer to the buffer if the string is found, or null otherwise.

23.59.2.65 FindStringInCurentObject()

```
const char * ParsedJsonDataSet::FindStringInCurentObject (
            const char * name )  [inline]
```

Find the string value of the element with the given name in the current object. Does not search sub-objects.

Parameters

name	The name of the element to find. •

Returns

A pointer to the buffer if the string is found, or null otherwise.

23.59.2.66 GetCurrent()

```
json_primitive_type ParsedJsonDataSet::GetCurrent ( )  [inline]
```

Get the element at the current position of the JSON data set.

Returns

The current element of the JSON data set.

Return values

UNDEFINED	If there is an error in the parsing or if the parsing is not complete.
EOF_EL	If it's the last element in the data set.

See also

GetFirst()

GetNext()

GetRawCurrent()

SkipCurrentValue()

ResetPosition()

23.59.2.67 GetFirst()

`json_primitive_type ParsedJsonDataSet::GetFirst ( ) [inline]`

Get the first element of the JSON data set.

Returns

The first element of the JSON data set.

Return values

UNDEFINED	If there is an error in the parsing or if the parsing is not complete.
EOF_EL	If it's the last element in the data set.

See also

GetNext()

GetCurrent()

GetRawCurrent()

ResetPosition()

SkipCurrentValue()

23.59.2.68 GetNext()

`json_primitive_type ParsedJsonDataSet::GetNext ( ) [inline]`

Get the element at the next position of the JSON data set.

Returns

The next element of the JSON data set.

Return values

UNDEFINED	If there is an error in the parsing or if the parsing is not complete.
EOF_EL	If it's the last element in the data set.

See also

GetFirst()

GetCurrent()

GetRawCurrent()

ResetPosition()

SkipCurrentValue()

23.59.2.69 GetNextArray()

`json_primitive_type ParsedJsonDataSet::GetNextArray ( )`
Iterates the current element of the JSON data set until it finds one of type BEGIN_ARRAY.

Returns

The next element with type BEGIN_ARRAY after the current position.

Return values

NOTFOUND	If there is no BEGIN_ARRAY element after the current position.

See also

GetNextNameInCurrentObject()

GetNextNameInCurrentArray()

23.59.2.70 GetNextBoolInCurrentArray()

`json_primitive_type ParsedJsonDataSet::GetNextBoolInCurrentArray ( )`
Get the next element of type TRUE_EL that exists in the current ARRAY after the current position.

Returns

The next element with type TRUE_EL in the current ARRAY after the current position.

Return values

NOTFOUND	If there is no NAME element after the current position.

See also

CurrentBool()

PermissiveCurrentBool()

23.59.2.71 GetNextName()

`json_primitive_type ParsedJsonDataSet::GetNextName ( ) [inline]`
Iterates the current element of the JSON data set until it finds one of type NAME.

Returns

The next element with type NAME after the current position.

Return values

NOTFOUND	If there is no NAME element after the current position.

See also

GetNextNameInCurrentObject()

GetNextNameInCurrentArray()

23.59.2.72 GetNextNameInCurrentArray()

```
json_primitive_type ParsedJsonDataSet::GetNextNameInCurrentArray ( )  [inline]
```

Get the next element of type NAME that exists in the current ARRAY after the current position.

Returns

The next element with type NAME in the current ARRAY after the current position.

Return values

NOTFOUND	If there is no NAME element after the current position.

See also

GetNextName()

GetNextNameInCurrentObject()

23.59.2.73 GetNextNameInCurrentObject()

```
json_primitive_type ParsedJsonDataSet::GetNextNameInCurrentObject ( )  [inline]
```

Get the next element of type NAME that exists in the current OBJECT after the current position.

Returns

The next element with type NAME in the current OBJECT after the current position.

Return values

NOTFOUND	If there is no NAME element after the current position.

See also

GetNextName()

GetNextNameInCurrentArray()

23.59.2.74 GetNextNumberInCurrentArray()

```
json_primitive_type ParsedJsonDataSet::GetNextNumberInCurrentArray ( )
```

Get the next element of type NUMBER that exists in the current ARRAY after the current position.

Returns

The next element with type NUMBER in the current ARRAY after the current position.

Return values

NOTFOUND	If there is no NAME element after the current position.

See also

CurrentNumber()

23.59.2.75 GetNextObject()

`json_primitive_type ParsedJsonDataSet::GetNextObject ( )`
Get the next element of type name that exists in the current OBJECT after the current position.

Returns

The next element with type BEGIN_OBJECT after the current position.

Return values

NOTFOUND	If there is no BEGIN_OBJECT element after the current position.

See also

GetNextNameInCurrentObject()

GetNextNameInCurrentArray()

23.59.2.76 GetNextObjectInCurrentArray()

`json_primitive_type ParsedJsonDataSet::GetNextObjectInCurrentArray ( )`
Get the next element of type BEGIN_OBJECT that exists in the current ARRAY after the current position.

Returns

The next element with type BEGIN_OBJECT in the current ARRAY after the current position.

Return values

NOTFOUND	If there is no BEGIN_OBJECT element after the current position.

See also

GetNextName()

GetNextNameInCurrentObject()

23.59.2.77 GetNextStringInCurrentArray()

`json_primitive_type ParsedJsonDataSet::GetNextStringInCurrentArray ( )`
Get the next element of type STRING that exists in the current ARRAY after the current position.

Returns

The next element with type STRING in the current ARRAY after the current position.

Return values

NOTFOUND	If there is no NAME element after the current position.

See also

CurrentString()

23.59.2.78 GetParsePosition()

```
JsonRef ParsedJsonDataSet::GetParsePosition ( )
```

Gets the current parse position object for the JSON data set.

Returns

The current parse position object.

23.59.2.79 GetPrintSize()

```
int ParsedJsonDataSet::GetPrintSize (
            bool pretty = false )  [inline]
```

Calculates how many characters the JSON data set would take to print.

Parameters

pretty	Whether indentation and new lines should be provided between elements.

Returns

The number of characters needed to print the JSON data set.

23.59.2.80 GetRawCurrent()

```
json_primitive_type ParsedJsonDataSet::GetRawCurrent ( )  [inline]
```

Get the element at the current position of the JSON data set, including non-public types.

Returns

The current element of the JSON data set.

Return values

UNDEFINED	If there is an error in the parsing or if the parsing is not complete.
EOF_EL	If it's the last element in the data set.

See also

GetFirst()

GetNext()

GetCurrent()

ResetPosition()

SkipCurrentValue()

23.59.2.81 name()

```
JsonRef ParsedJsonDataSet::name (
            const char * name )  [inline]
```

Get a JsonRef representing any entity from a parent object by key name.
Useful if more than one JSON object/element has the same name.

Example

```
JsonRef userThree = myJsonRef.name("users")[3];
```

Parameters

∗name	Pointer to name of the entity to find

Returns

The reference to the entity of type JsonRef

See also

ParsedJsonDataSet::operator()(), ParsedJsonDataSet::object()

23.59.2.82 next()

```
JsonRef ParsedJsonDataSet::next ( )  [inline]
```

Traverse: check for ref validity and increment the position.

23.59.2.83 object()

```
JsonRef ParsedJsonDataSet::object (
            const char * name )  [inline]
```

Get a JsonRef representing an object from a parent object by key name.

Example

```
JsonRef john = myJsonRef.object("users").object("john");
```

Parameters

∗name	Pointer to name of the object to find

Returns

The reference to the object of type JsonRef

23.59.2.84 operator()()

```
JsonRef ParsedJsonDataSet::operator() (
            const char * name )  [inline]
```

Get a JsonRef representing any entity from a parent object by key name.
Shortcut for JsonRef::name()

Example

```
JsonRef allUsers = myJsonRef("users");
```

Parameters

∗name	Pointer to name of the entity to find

Returns

The reference to the entity of type JsonRef

See also

ParsedJsonDataSet::name(), ParsedJsonDataSet::object()

23.59.2.85 operator[]()

```
JsonRef ParsedJsonDataSet::operator[] (
            int i )  [inline]
```

Get a JsonRef from an array by numerical index.

Example

```
JsonRef userThree = myJsonRef("users")[3];
```

Returns

A JsonRef if array element is found

23.59.2.86 PermissiveCurrentBool()

```
bool ParsedJsonDataSet::PermissiveCurrentBool ( )  [inline]
```

Returns true if the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.

Return values

true	If the current element a TRUE_EL primitive type, is "True", "true", or is a non-zero number.
false	If the current element does not meet the true condition..

23.59.2.87 PrintChildren()

```
int ParsedJsonDataSet::PrintChildren (
            bool pretty = false )  [inline]
```

Output child objects of this JsonRef as JSON text to stdout.

23.59.2.88 PrintChildrenToBuffer()

```
int ParsedJsonDataSet::PrintChildrenToBuffer (
            char * buffer,
            int maxlen,
            bool pretty = false )  [inline]
```

Output child objects of this JsonRef as JSON text to a buffer pointer.

23.59.2.89 PrintChildrenToFd()

```
int ParsedJsonDataSet::PrintChildrenToFd (
            int fd,
            bool pretty = false )  [inline]
```

Output child objects of this JsonRef as JSON text to a file descriptor.

23.59.2.90 PrintObject()

```
int ParsedJsonDataSet::PrintObject (
            bool pretty = false )  [inline]
```

Prints the JSON data set to stdout.

Parameters

pretty	Whether to show indentation and new lines between JSON elements.

23.59.2.91 PrintObjectToBuffer()

```
int ParsedJsonDataSet::PrintObjectToBuffer (
            char * buffer,
            int maxlen,
            bool pretty = false )  [inline]
```

Prints the JSON data set to a provided buffer.

Parameters

buffer	The buffer to print to.
maxlen	The max length of the buffer.
pretty	Whether indentation and new lines should be provided between elements.

Returns

The number of characters printed.

23.59.2.92 PrintObjectToFd()

```
int ParsedJsonDataSet::PrintObjectToFd (
            int fd,
            bool pretty = false )  [inline]
```

Prints the JSON data set to a specified file descriptor.

Parameters

fd	The file descriptor to print to.
pretty	Whether indentation and new lines should be provided between elements.

23.59.2.93 ReadFrom()

```
virtual int ParsedJsonDataSet::ReadFrom (
            int fd )  [virtual]
```

Reads in data from the specified file descriptor and parses it into the JSON data set.

Parameters

fd	The file descriptor to read data from.

Returns

The number of bytes read

23.59.2.94 ResetPosition()

```
void ParsedJsonDataSet::ResetPosition ( )  [inline]
```

Resets the parser position to the beginning of the JSON data set.

See also

GetFirst()

GetNext()

GetCurrent()

SkipCurrentValue()

23.59.2.95 SetParsePosition()

```
JsonRef ParsedJsonDataSet::SetParsePosition (
            JsonRef pos )
```

Sets the current parse position object for the JSON data set.

Parameters

pos	The JsonRef that you want to set the current position to.

Returns

The current parse ref position for the JSON data set.

23.59.2.96 SkipCurrentValue()

```
json_primitive_type ParsedJsonDataSet::SkipCurrentValue ( )  [inline]
```

Skips over the current value, and get the next element. If called inside an ARRAY or OBJECT, it will walk to the end of the ARRAY or OJBECT and return the next element it finds. If it reaches the end of the ARRAY or OBJECT before it finds a value, it will return NOTFOUND.

Returns

The next element.

Return values

NOTFOUND	If there are no other elements after the current one.

See also

GetFirst()

GetCurrent()

GetNext()

ResetPosition()

SkipCurrentValue()

23.59.2.97 start()

```
JsonRef ParsedJsonDataSet::start ( )  [inline]
```
Begin traversal: check for ref validity and reset the position.

23.59.2.98 StartBuilding()

```
void ParsedJsonDataSet::StartBuilding ( )
```
Use to start building the JSON data set.

23.59.2.99 WriteData()

```
virtual int ParsedJsonDataSet::WriteData (
            const unsigned char * pCopyFrom,
            int numBytes )  [virtual]
```
Writes the passed in data to the JSON data set.

Parameters

pCopyFrom	What data to write.
numBytes	How many bytes to copy.

Returns

The number of bytes written

The documentation for this class was generated from the following file:

- json_lexer.h

23.60 ParsedURI Class Reference

Parsed Uniform Resource Identifier Class (URI)
`#include <http_funcs.h>`

Public Member Functions

- void NewUri (const char *uri, uint16_t timeout=20 *TICKS_PER_SECOND, bool skipLookup=false)
  Replace the existing (URI) object with new information.
- **ParsedURI** ()
  Constructor to create a new blank URI object. For rare cases.
- ParsedURI (const char *uri, uint16_t timeout=20 *TICKS_PER_SECOND, bool skipLookup=false)
  Constructor to create a new URI object.
- bool valid () const
  Check to see if URI is valid.
- const char * GetPath ()
  Get the cached URI path.
- const char * GetHost ()
  Get the cached URI Host name.
- IPADDR GetAddr ()
  Get the resolved host Address.
- uint16_t GetPort ()
  Get the cached URI network port number.
- bool IsSecure ()

Check the security state of the cached URI object. Requires WEB_CLIENT_SSL_SUPPORT be defined in predef.h.

- void **Invalidate** ()

  Set the state of the cached URI object to invalid.

23.60.1 Detailed Description

Parsed Uniform Resource Identifier Class (URI)
Web Client HTTP functions can pass a URL, which has to be parsed and use DNS to resolve if necessary. This class provides an object with the parsed and DNS results that can be used for future transactions. The intent of using this class is to cache the parsed results and dns lookup for communication to the same resource.
Enabling WEB_CLIENT_SSL_SUPPORT in predef.h adds TLS support.
The maximum size of the cached URI, path and hose are 256 bytes each.

23.60.2 Constructor & Destructor Documentation

23.60.2.1 ParsedURI()

```
ParsedURI::ParsedURI (
            const char * uri,
            uint16_t timeout = 20 * TICKS_PER_SECOND,
            bool skipLookup = false )  [inline]
```

Constructor to create a new URI object.

Parameters

uri	Pointer to a URI.
timeout	Number of sysem time tick to wait.
skipLookup	Skip URI lookup. Default is false.

23.60.3 Member Function Documentation

23.60.3.1 GetAddr()

```
IPADDR ParsedURI::GetAddr ( )  [inline]
```

Get the resolved host Address.

Returns

The resolved host address.

23.60.3.2 GetHost()

```
const char * ParsedURI::GetHost ( )  [inline]
```

Get the cached URI Host name.

Returns

Pointer to the cached host name if the cached URI is valid, otherwise NULL.

23.60.3.3 GetPath()

```
const char * ParsedURI::GetPath ( )  [inline]
```
Get the cached URI path.

Returns

Pointer to the cached path string if the cached URI is valid, otherwise NULL.

23.60.3.4 GetPort()

```
uint16_t ParsedURI::GetPort ( )  [inline]
```
Get the cached URI network port number.

Returns

The cached URI port number if the cached URI is valid, otherwise 0.

23.60.3.5 IsSecure()

```
bool ParsedURI::IsSecure ( )  [inline]
```
Check the security state of the cached URI object. Requires WEB_CLIENT_SSL_SUPPORT be defined in predef.h.

Return values

true	Secure.
false	URI object is invalid.

23.60.3.6 NewUri()

```
void ParsedURI::NewUri (
            const char * uri,
            uint16_t timeout = 20 *TICKS_PER_SECOND,
            bool skipLookup = false )
```
Replace the existing (URI) object with new information.
Update the existing URI with a new value.

Parameters

uri	Pointer to a URI string.
timeout	Number of sysem time ticks to wait.
skipLookup	Skip URI lookup. Default is false.

23.60.3.7 valid()

```
bool ParsedURI::valid ( ) const  [inline]
```
Check to see if URI is valid.

Return values

true	if valid
false	if not valid

The documentation for this class was generated from the following file:

- http_funcs.h

23.61 PinIO Class Reference

GPIO Pin Class.
`#include <cpu_pins.h>`

Public Types

- enum pin_fn_t { }
- enum pin_fn_t { }

  Pin Function modes to configure the managed pin(s).

Public Member Functions

- **PinIO** ()

  Construct an empty PinIO. Exists for bootstrap compatibility; not intended for general use.
- constexpr PinIO (uint32_t port, uint32_t pin)

  Construct a PinIO for a specific cpu pin.
- constexpr PinIO (const PinIO &rhs)

  Construct a copy of another PinIO.
- void setFn (pin_fn_t fn) const

  Set the pin function for the managed pin(s).
- int8_t getFn ()

  Get the pin function for the managed pin(s).
- void function (pin_fn_t fn) const

  Set the pin function for the managed pin(s).
- void **hiz** () const

  Configure the pin(s) to Input.
- void **drive** () const

  Configure the pin(s) to Output, without modifying the driven value.
- void **set** () const

  Drive the pin(s) High.
- void **clr** () const

  Drive the pin(s) Low.
- bool tgl () const

  Toggle the driven value for the pin(s).
- bool toggle () const

  Toggle the driven value for the pin(s).
- bool readBack () const

  Read the state of the pin(s) line state without changing the pin function or direction.
- bool read () const

  Configure the pin as an input then return the line state.
- bool operator= (bool val)

  Assign a driven value to the pin(s).
- PinIO & operator= (const PinIO &rhs)

  Assign a driven value to the pin(s) based on the line state of another pin(s).
- operator bool () const

  Read the line state of the pin(s).
- bool operator! () const

Return the opposite of the driven value of the pin(s).

- void multidrv (bool enable) const

  Configure the multidrive/open-drain driver for the pin(s).

- void setHighStrength (bool bHighDrive)

  Configure the drive strength of the output driver for the pin(s).

- void PullUp (bool enable) const

  Configure the pad Pull Up resistor for the pin(s).

- void PullDown (bool enable) const

  Configure the pad Pull Down resistor for the pin(s).

- uint16_t analogRead () const

  Read an analog voltage on the given Pin. Only available for pins connected to the ADC.

Public Attributes

- volatile Pio & pio
- uint32_t mask

23.61.1 Detailed Description

GPIO Pin Class.

23.61.2 Member Enumeration Documentation

23.61.2.1 pin_fn_t [1/2]

`enum PinIO::pin_fn_t`

Enumerator

PIN_FN_IN	Input.
PIN_FN_OUT	Output.
PIN_FN_IN	Input.
PIN_FN_OUT	Output.
PIN_FN_A	Peripheral A.
PIN_FN_B	Peripheral B.
PIN_FN_C	Peripheral C.
PIN_FN_D	Peripheral D.

23.61.2.2 pin_fn_t [2/2]

`enum PinIO::pin_fn_t`

Pin Function modes to configure the managed pin(s).

Enumerator

PIN_FN_IN	Input.
PIN_FN_OUT	Output.
PIN_FN_IN	Input.
PIN_FN_OUT	Output.
PIN_FN_A	Peripheral A.
PIN_FN_B	Peripheral B.

Enumerator

PIN_FN_C	Peripheral C.
PIN_FN_D	Peripheral D.

23.61.3 Constructor & Destructor Documentation

23.61.3.1 PinIO() [1/2]

```
constexpr PinIO::PinIO (
            uint32_t port,
            uint32_t pin )  [inline], [constexpr]
```

Construct a PinIO for a specific cpu pin.

Parameters

port	The zero-indexed numeric instance number of the desired Pio module. (PIOA=0, PIOB=1, PIOC=2, etc.)
pin	The zero-indexed pin number within the requested port that the new PinIO is to manage.

Example usage for pin 'PIOD_24'

```
PinIO LED1(3, 24);
```

23.61.3.2 PinIO() [2/2]

```
constexpr PinIO::PinIO (
            const PinIO & rhs )  [inline], [constexpr]
```

Construct a copy of another PinIO.

Parameters

rhs	The PinIO to construct a copy of.

23.61.4 Member Function Documentation

23.61.4.1 analogRead()

```
uint16_t PinIO::analogRead ( ) const
```

Read an analog voltage on the given Pin. Only available for pins connected to the ADC.

23.61.4.2 function()

```
void PinIO::function (
            pin_fn_t fn ) const  [inline]
```

Set the pin function for the managed pin(s).

Parameters

fn	The pin function to be assigned to the managed pin(s).

23.61.4.3 getFn()

```
int8_t PinIO::getFn ( )  [inline]
```

Get the pin function for the managed pin(s).

Returns

The pin function as a pin_fn_t of the managed pin(s). Returns -1 if pin is not an I/O (ex: GND, VCC)

23.61.4.4 multidrv()

```
void PinIO::multidrv (
            bool enable ) const  [inline]
```

Configure the multidrive/open-drain driver for the pin(s).

Parameters

enable	True: configure the output driver as Open Drain. False: configure the output driver as Push-Pull.

23.61.4.5 operator bool()

```
PinIO::operator bool ( ) const  [inline]
```

Read the line state of the pin(s).

Returns

The line state of the pin(s): true = High, false = Low

23.61.4.6 operator"!()

```
bool PinIO::operator!  ( ) const  [inline]
```

Return the opposite of the *driven* value of the pin(s).

Returns

The opposite of the *driven* value of the pin(s).

23.61.4.7 operator=() [1/2]

```
bool PinIO::operator= (
            bool val )  [inline]
```

Assign a driven value to the pin(s).

Parameters

val	The value to be driven on the pin(s): true = High, false = Low

Returns

The driven value for the pin(s).

23.61.4.8 operator=() [2/2]

```
PinIO & PinIO::operator= (
            const PinIO & rhs )  [inline]
```

Assign a driven value to the pin(s) based on the line state of another pin(s).

Parameters

rhs	The pin(s) who's line state will be assigned to this pin.

Returns

A copy of the PinIO being assigned to.

23.61.4.9 PullDown()

```
void PinIO::PullDown (
            bool enable ) const  [inline]
```

Configure the pad Pull Down resistor for the pin(s).

Parameters

enable	True: enable the pad Pull Down. False: disable the pad Pull Down.

23.61.4.10 PullUp()

```
void PinIO::PullUp (
            bool enable ) const  [inline]
```

Configure the pad Pull Up resistor for the pin(s).

Parameters

enable	True: enable the pad Pull Up. False: disable the pad Pull Up.

23.61.4.11 read()

```
bool PinIO::read ( ) const  [inline]
```

Configure the pin as an input then return the line state.

Returns

The line state of the pin(s).

23.61.4.12 readBack()

```
bool PinIO::readBack ( ) const  [inline]
```

Read the state of the pin(s) line state *without* changing the pin function or direction.

Returns

The line state of the pin(s).

23.61.4.13 setFn()

```
void PinIO::setFn (
            pin_fn_t fn ) const  [inline]
```

Set the pin function for the managed pin(s).

Parameters

fn	The pin function to be assigned to the managed pin(s).

23.61.4.14 setHighStrength()

```
void PinIO::setHighStrength (
            bool bHighDrive )  [inline]
```

Configure the drive strength of the output driver for the pin(s).

Parameters

bHighDrive	True: configure the output driver for High Drive strength. False: configure the output driver for Low Drive strength.

23.61.4.15 tgl()

```
bool PinIO::tgl ( ) const  [inline]
```

Toggle the driven value for the pin(s).

Returns

The previously driven value for the pin(s).

23.61.4.16 toggle()

```
bool PinIO::toggle ( ) const  [inline]
```

Toggle the driven value for the pin(s).

Returns

The previously driven value for the pin(s).

The documentation for this class was generated from the following files:

- coldfire/cpu/MCF5441X/include/cpu_pins.h
- cortex-m7/cpu/SAME70/include/cpu_pins.h

23.62 PinIOArray2 Class Reference

```
#include <pins.h>
```

23.62.1 Detailed Description

The following class is created to assist in processor pin configuration and operation for GPIO and special functions. Each pin is defined by the system in the "\nburn\<Platform>\system" and "\nburn\<Platform>\include" directories for the specific platform you are using.

Examples: P1[5] = 1; Set pin P1-5 high P2[30] = 0; Set pin P2-30 low P1[5].hiz(); Set pin P1-5 to be a high impedance input BOOL bpinstate = P2[30]; Set the pin to be an input and read if (!P2[30]) iprintf("The pin is low");
The documentation for this class was generated from the following file:

- MODM7AE70/include/pins.h

23.63 PinIOArrayJ1 Class Reference

`#include <pins.h>`

23.63.1 Detailed Description

The following class is created to assist in processor pin configuration and operation for GPIO and special functions. Each pin is defined by the system in the "\nburn\<Platform>\system" and "\nburn\<Platform>\include" directories for the specific platform you are using.
Examples: P1[5] = 1; Set pin P1-5 high P2[30] = 0; Set pin P2-30 low P1[5].hiz(); Set pin P1-5 to be a high impedance input BOOL bpinstate = P2[30]; Set the pin to be an input and read if (!P2[30]) iprintf("The pin is low");
The documentation for this class was generated from the following file:

- SBE70LC/include/pins.h

23.64 PinIOJ1Array Class Reference

`#include <pins.h>`

23.64.1 Detailed Description

The following class is created to assist in processor pin configuration and operation for GPIO and special functions. Each pin is defined by the system in the `\Nburn\<Platform>\system` and `\Nburn\<Platform>\include` directories for the specific platform you are using.
Examples: J1[5] = 1; Set pin J1-5 high J2[30] = 0; Set pin J2-30 low J1[5].hiz(); Set pin J1-5 to be a high impedance input BOOL bpinstate = J2[30]; Set the pin to be an input and read if (!J2[30]) iprintf("The pin is low");
The documentation for this class was generated from the following file:

- MOD5441X/include/pins.h

23.65 PinVector< n > Class Template Reference

GPIO Pin Vector Class PinVector is a template instantiation of the _PinVector class, allowing for minimal storage requirements for potentially large vectors, without heavy code duplication due to template copies.
`#include <cpu_pins.h>`
Inherits _PinVector.

Public Member Functions

- **PinVector** ()

  Bare constructor for a PinVector, where the bit configurations will be made later.
- PinVector (PinIO *initpins, uint32_t pinCount)

  PinVector Constructor, where pin configurations will be made at the time of construction.
- uint32_t operator= (uint32_t val)

  Assign a value to the PinVector Bus.

Public Member Functions inherited from _PinVector

- uint32_t operator= (uint32_t val)

  Assign a value to the _PinVector Bus.
- PinIO operator[ ] (int idx)

  Access the PinIO for a specific bit position in the _PinVector.
- void config (uint32_t idx, PinIO cfg)

  Set the PinIO that will be used for a given bit position in the _PinVector.
- void config (PinIO *pinCfgs, uint32_t count)

  Configure the _PinVector based on an array of PinIOs. The index of the PinIO in the configuration array will determine the bit position within the _PinVector that that PinIO represents.
- operator uint32_t () const

  Read the line state of the _PinVector bus.

23.65.1 Detailed Description

template<uint8_t n>
class PinVector< n >

GPIO Pin Vector Class PinVector is a template instantiation of the _PinVector class, allowing for minimal storage requirements for potentially large vectors, without heavy code duplication due to template copies.

Parameters

PinVector::pinStore	pinStore is the actual storage allocation for large PinVector's PinIO configurations.

23.65.2 Constructor & Destructor Documentation

23.65.2.1 PinVector()

```
template<uint8_t n>
PinVector< n >::PinVector (
            PinIO * initpins,
            uint32_t pinCount )  [inline]
```

PinVector Constructor, where pin configurations will be made at the time of construction.

Parameters

initpins	The array of PinIOs to be used to configure the PinVector.
pinCount	The number of PinIOs in the configuration array to be used.

23.65.3 Member Function Documentation

23.65.3.1 operator=()

```
template<uint8_t n>
uint32_t PinVector< n >::operator= (
            uint32_t val )  [inline]
```

Assign a value to the PinVector Bus.

Parameters

val	The value to assign to the bus.

Returns

The value driven on the bus.

The documentation for this class was generated from the following file:

- cortex-m7/cpu/SAME70/include/cpu_pins.h

23.66 QSPI_Record Struct Reference

This struct contains the major variables/configurations used for a QSPI transfer.
`#include <qspi.h>`

Public Attributes

- volatile uint8_t * **pQSPIRxbuf**

  This pointer is used to track the locations in memory where data will be read from the peripheral.
- volatile uint8_t * **pQSPITxbuf**

  This pointer is used to track the locations in memory where data will be written to the peripheral.
- uint8_t **BitsPerQueue**

  This is the number if bits per transfer, (value = 8 - 32).
- uint32_t **QSPI_SizeLeft**

  This is the number if bytes left in the transfer.
- uint16_t **Command_Mask**

  This is a partial configuration for the queue's command reg.
- OS_SEM * **QSPI_Sem**

  This is a pointer to an external semaphore provided by QSPIStart().

23.66.1 Detailed Description

This struct contains the major variables/configurations used for a QSPI transfer.
The documentation for this struct was generated from the following file:

- qspi.h

23.67 SerialRecord Struct Reference

`#include <serialrecord.h>`

Public Member Functions

- void AssignUartNumber (void)
- void ProcessTCPReadSerialData (void)
- void ProcessSpecialFrameTCPReadSerialData (void)
- void ProcessSpecialFrameWriteNetworkData (void)
- void ProcessSpecialFrameWriteTimeout (void)
- void MakeTcpConnection (void)
- void CloseListenPort (void)
- void MakeUdpConnection (void)
- bool OkToListen (void)
- void ProcessTimeouts (void)

- void ProcessAccept (void)
- void OpenSerialPort (void)
- void ProcessReadNetworkData (void)
- void OpenListenPort (void)
- void ProcessWriteNetworkData (void)
- void ProcessWriteSerialData (void)
- void ProcessSerialError (void)
- void ProcessListenError (void)
- void ProcessNetworkError (void)
- void GetCurrentChannelStatus (char *buffer)

23.67.1 Detailed Description

Class Definition (struct is class with all members public)

23.67.2 Member Function Documentation

23.67.2.1 AssignUartNumber()

```
void SerialRecord::AssignUartNumber (
            void  )
```

Methods
Assign physical UART number to PortRecord array index member

23.67.2.2 CloseListenPort()

```
void SerialRecord::CloseListenPort (
            void  )
```

Close the listening port

23.67.2.3 GetCurrentChannelStatus()

```
void SerialRecord::GetCurrentChannelStatus (
            char * buffer )
```

" Connected to IP xx.xx.xx.xx " Listening on port xx " UDP mode with learned send-to IP Address : xx.xx.xx.xx " UDP mode send to Address : xx.xx.xx.xx " Idle

23.67.2.4 MakeTcpConnection()

```
void SerialRecord::MakeTcpConnection (
            void  )
```

Initiate a TCP connection

23.67.2.5 MakeUdpConnection()

```
void SerialRecord::MakeUdpConnection (
            void  )
```

Initiate a UDP connection

23.67.2.6 OkToListen()

```
bool SerialRecord::OkToListen (
            void  )
```

Check if conditions are right to establish a listening port

23.67.2.7 OpenListenPort()

```
void SerialRecord::OpenListenPort (
            void  )
```

Open the listening port

23.67.2.8 OpenSerialPort()

```
void SerialRecord::OpenSerialPort (
            void  )
```

Open a serial port

23.67.2.9 ProcessAccept()

```
void SerialRecord::ProcessAccept (
            void  )
```

Called when connections are waiting to be accepted

23.67.2.10 ProcessListenError()

```
void SerialRecord::ProcessListenError (
            void  )
```

Process listening errors

23.67.2.11 ProcessNetworkError()

```
void SerialRecord::ProcessNetworkError (
            void  )
```

Process network errors

23.67.2.12 ProcessReadNetworkData()

```
void SerialRecord::ProcessReadNetworkData (
            void  )
```

Called when data is received on the network port side

23.67.2.13 ProcessSerialError()

```
void SerialRecord::ProcessSerialError (
            void  )
```

Process serial errors

23.67.2.14 ProcessSpecialFrameTCPReadSerialData()

```
void SerialRecord::ProcessSpecialFrameTCPReadSerialData (
            void  )
```

Called to initiate an outgoing TCP connection when data is received on the serial data port side (this version for TCP packet customization logic)

23.67.2.15 ProcessSpecialFrameWriteNetworkData()

```
void SerialRecord::ProcessSpecialFrameWriteNetworkData (
            void  )
```

Called when write space has just opened up on network port (this version for TCP packet customization logic)

23.67.2.16 ProcessSpecialFrameWriteTimeout()

```
void SerialRecord::ProcessSpecialFrameWriteTimeout (
            void  )
```

Time-out function used with TCP packet customization logic - waits for a given time to accumulate serial characters before flushing them out the network side

23.67.2.17 ProcessTCPReadSerialData()

```
void SerialRecord::ProcessTCPReadSerialData (
            void  )
```

Called to initiate an outgoing TCP connection when data is received on the serial data port side

23.67.2.18 ProcessTimeouts()

```
void SerialRecord::ProcessTimeouts (
            void  )
```

Test all of the communication time-outs Periodically check for live connection at a user-specified interval by sending an empty packet to the target device.
Wait at least two seconds for ACK to return after sending keep-alive packet. Target is lost if ACK was not the last packet received.

23.67.2.19 ProcessWriteNetworkData()

```
void SerialRecord::ProcessWriteNetworkData (
            void  )
```

Called when write space has just opened up on network port

23.67.2.20 ProcessWriteSerialData()

```
void SerialRecord::ProcessWriteSerialData (
            void  )
```

Called when write space has just opened up serial port
The documentation for this struct was generated from the following files:

- serialrecord.h
- serialrecord.cpp

23.68 SPI_QSPI Class Reference

The Single-Bit SPI mode QSPI Peripheral Class.
`#include <quadspi.h>`
Inherits SPIModule.

Public Member Functions

- SPI_QSPI (uint8_t QuadSPIModule=DEFAULT_QUADSPI_MODULE)

  Create a SPI object.
- SPI_QSPI (uint8_t QuadSPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Create a SPI object and Initialize with parameters.
- virtual uint8_t Init (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0) override

  Initialize an existing SPI object.
- virtual uint32_t SetBusSpeed (uint32_t maxSpeed) override

  Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.
- virtual uint8_t Start (uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST) override

  Start a SPI transfer.
- virtual uint8_t Tx (uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST) override

Convenience function for unidirectional transmit.

- uint8_t Rx (uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional receive.

- virtual bool SetCS (uint8_t CS) override

  Set the chip select configuration for the SPI object's bus transactions.

Public Member Functions inherited from SPIModule

- SPIModule (uint8_t SPIModule)

  Create a SPI object.

- SPIModule (uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Create a SPI object and Initialize with parameters.

- virtual uint8_t Init (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize an existing SPI object.

- virtual uint32_t SetBusSpeed (uint32_t maxSpeed)

  Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.

- virtual uint8_t Start (uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Start a SPI transfer.

- virtual uint8_t Tx (uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional transmit.

- virtual uint8_t Rx (uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional receive.

- virtual bool RegisterSem (OS_SEM *finishedSem)

  Register a semaphore for the SPI module.

- virtual bool ClrSem ()

  Clear a semaphore registration.

- virtual OS_SEM * GetSem ()

  Obtain a pointer to the SPI finished semaphore.

- virtual bool Done ()

  Function to check SPI status.

- virtual uint32_t GetActualBaudrate ()

  Returns the active baud rate.

- virtual bool SetCS (uint8_t CS)

  Set the chip select configuration for the SPI object's bus transactions.

Additional Inherited Members

Static Public Member Functions inherited from SPIModule

- static BOOL Done (uint8_t SPIModule)

  Static function to check SPI status.

23.68.1 Detailed Description

The Single-Bit SPI mode QSPI Peripheral Class.
Class definition for SPI module objects

23.68.2 Constructor & Destructor Documentation

23.68.2.1 SPI_QSPI() [1/2]

```
SPI_QSPI::SPI_QSPI (
            uint8_t QuadSPIModule = DEFAULT_QUADSPI_MODULE )
```

Create a SPI object.
This constructor will create the object, but will *not* initialize the SPI module. You must call the Init() member function to initialize.

Parameters

QuadSPIModule	QSPI module number (0). The SAME70 only has one QuadSPI module.

23.68.2.2 SPI_QSPI() [2/2]

```
SPI_QSPI::SPI_QSPI (
            uint8_t QuadSPIModule,
            uint32_t baudRateInBps,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )
```

Create a SPI object and Initialize with parameters.
Creates a SPI object and initializes the SPI module. You do not need to call the Init() member function.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for $>$ than 8 bits per transfer then the data must be uint16_t aligned

Parameters

QuadSPIModule	QSPI module number (0). The SAME70 has only one QuadSPI module
baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8-bit up to 16-bit transfer sizes are valid
peripheralChipSelects	SPI chip selects to use for transfer. Only one chip select is available to the QuadSPI peripheral on pin P2[48].
chipSelectPolarity	0 = inactive logic level low, 1 = high
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	Delay from chip select to valid clock (default is 0). When 0, the delay is half the clock period.
delayAfterTransfer	When 0, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

23.68.3 Member Function Documentation

23.68.3.1 Init()

```
virtual uint8_t SPI_QSPI::Init (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )  [override], [virtual]
```

Initialize an existing SPI object.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned

Parameters

baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8-bit up to 16-bit transfer sizes are valid
peripheralChipSelects	SPI chip selects to use for transfer. Only one chip select is available to the QuadSPI peripheral on pin P2[48].
chipSelectPolarity	0 = inactive logic level low, 1 = high
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	Delay from chip select to valid clock (default is 0). When 0, the delay is half the clock period.
delayAfterTransfer	When 0, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.

23.68.3.2 Rx()

```
uint8_t SPI_QSPI::Rx (
            uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline], [virtual]
```

Convenience function for unidirectional receive.

Parameters

receiveBufferPtr	Pointer to the buffer to write the received data
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.

23.68.3.3 SetBusSpeed()

```
virtual uint32_t SPI_QSPI::SetBusSpeed (
            uint32_t maxSpeed )  [override], [virtual]
```

Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.

Parameters

maxSpeed	The maximum desired bus speed

Returns

The actual bus speed

Reimplemented from SPIModule.

23.68.3.4 SetCS()

```
virtual bool SPI_QSPI::SetCS (
            uint8_t CS )  [inline], [override], [virtual]
```

Set the chip select configuration for the SPI object's bus transactions.
Single-bit SPI mode for the QuadSPI peripheral only supports one chip select. This function is defined for compatibility between SPI drivers.

Parameters

CS	The QuadSPI module has only one chip select.

Returns

true if successful, false if SPI is currently active

Reimplemented from SPIModule.

23.68.3.5 Start()

```
virtual uint8_t SPI_QSPI::Start (
            uint8_t * transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [override], [virtual]
```

Start a SPI transfer.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
receiveBufferPtr	Pointer to the buffer to store the received data
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state dspiChipSelectMode

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.

23.68.3.6 Tx()

```
virtual uint8_t SPI_QSPI::Tx (
            uint8_t * transmitBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline], [override], [virtual]
```

Convenience function for unidirectional transmit.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.
The documentation for this class was generated from the following file:

- quadspi.h

23.69 SPI_USART Class Reference

USART in SPI mode Peripheral Module Class.
`#include <usart.h>`
Inherits SPIModule.

Public Member Functions

- SPI_USART (uint8_t USARTModule)

  Create a SPI object.
- SPI_USART (uint8_t USARTModule, uint32_t baudRateInBps, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Create a SPI object and Initialize with parameters.
- virtual uint8_t Init (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0) override

  Initialize an existing SPI object.
- virtual uint32_t SetBusSpeed (uint32_t maxSpeed) override

Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.

- virtual uint8_t Start (uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST) override

  Start a SPI transfer.
- virtual uint8_t Tx (uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST) override

  Convenience function for unidirectional transmit.
- uint8_t Rx (uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional receive.
- virtual bool SetCS (uint8_t CS) override

  Set the chip select configuration for the SPI object's bus transactions.

Public Member Functions inherited from SPIModule

- SPIModule (uint8_t SPIModule)

  Create a SPI object.
- SPIModule (uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Create a SPI object and Initialize with parameters.
- virtual uint8_t Init (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize an existing SPI object.
- virtual uint32_t SetBusSpeed (uint32_t maxSpeed)

  Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.
- virtual uint8_t Start (uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Start a SPI transfer.
- virtual uint8_t Tx (uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional transmit.
- virtual uint8_t Rx (uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional receive.
- virtual bool RegisterSem (OS_SEM *finishedSem)

  Register a semaphore for the SPI module.
- virtual bool ClrSem ()

  Clear a semaphore registration.
- virtual OS_SEM * GetSem ()

  Obtain a pointer to the SPI finished semaphore.
- virtual bool Done ()

  Function to check SPI status.
- virtual uint32_t GetActualBaudrate ()

  Returns the active baud rate.
- virtual bool SetCS (uint8_t CS)

  Set the chip select configuration for the SPI object's bus transactions.

Additional Inherited Members

Static Public Member Functions inherited from SPIModule

- static BOOL Done (uint8_t SPIModule)

  Static function to check SPI status.

23.69.1 Detailed Description

USART in SPI mode Peripheral Module Class.
Class definition for SPI module objects

23.69.2 Constructor & Destructor Documentation

23.69.2.1 SPI_USART() [1/2]

```
SPI_USART::SPI_USART (
            uint8_t USARTModule )
```

Create a SPI object.
This constructor will create the object, but will *not* initialize the SPI module. You must call the Init() member function to initialize.

Parameters

USARTModule	SPI module number (0 or 1)

23.69.2.2 SPI_USART() [2/2]

```
SPI_USART::SPI_USART (
            uint8_t USARTModule,
            uint32_t baudRateInBps,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )
```

Create a SPI object and Initialize with parameters.
Creaets a SPI object and initializes the SPI module. You do not need to call the Init() member function.

Parameters

USARTModule	USART module number (0 or 1)
baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8, 16 or 32
peripheralChipSelects	SPI chip selects to use for transfer
chipSelectPolarity	0 = inactive logic level low, 1 = high
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	This option is not supported by the USART peripheral in SPI mode.
delayAfterTransfer	This option is not supported by the USART peripheral in SPI mode.

23.69.3 Member Function Documentation

23.69.3.1 Init()

```
virtual uint8_t SPI_USART::Init (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )  [override], [virtual]
```

Initialize an existing SPI object.

Parameters

baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8, 16 or 32
peripheralChipSelects	SPI chip selects to use for transfer
chipSelectPolarity	0 = inactive logic level low, 1 = high
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	This option is not supported by the USART peripheral in SPI mode.
delayAfterTransfer	This option is not supported by the USART peripheral in SPI mode.

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.

23.69.3.2 Rx()

```
uint8_t SPI_USART::Rx (
            uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline], [virtual]
```

Convenience function for unidirectional receive.

Parameters

receiveBufferPtr	Pointer to the buffer to write the received data
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.

23.69.3.3 SetBusSpeed()

```
virtual uint32_t SPI_USART::SetBusSpeed (
```

```
            uint32_t maxSpeed )  [override], [virtual]
```

Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.

Parameters

maxSpeed	The maximum desired bus speed

Returns

The actual bus speed

Reimplemented from SPIModule.

23.69.3.4 SetCS()

```
virtual bool SPI_USART::SetCS (
            uint8_t CS )  [inline], [override], [virtual]
```

Set the chip select configuration for the SPI object's bus transactions.

Parameters

CS	Chip select to set, 0 - 3.

Returns

true if successful, false if SPI is currently active

Reimplemented from SPIModule.

23.69.3.5 Start()

```
virtual uint8_t SPI_USART::Start (
            uint8_t * transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [override], [virtual]
```

Start a SPI transfer.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
receiveBufferPtr	Pointer to the buffer to store the received data
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.

23.69.3.6 Tx()

```
virtual uint8_t SPI_USART::Tx (
            uint8_t * transmitBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline], [override], [virtual]
```

Convenience function for unidirectional transmit.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented from SPIModule.
The documentation for this class was generated from the following file:

- usart.h

23.70 SPIModule Class Reference

SPI Peripheral Module Class.
`#include <dspi.h>`
Inherited by SPI_QSPI, and SPI_USART.

Public Member Functions

- SPIModule (uint8_t SPIModule)

  Create a SPI object.
- SPIModule (uint8_t SPIModule, uint32_t baudRateInBps, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Create a SPI object and Initialize with parameters.
- virtual uint8_t Init (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x00, uint8_t chipSelectPolarity=0x0F, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize an existing SPI object.
- virtual uint32_t SetBusSpeed (uint32_t maxSpeed)

  Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.
- virtual uint8_t Start (uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Start a SPI transfer.
- virtual uint8_t Tx (uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Convenience function for unidirectional transmit.
- virtual uint8_t Rx (uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive=DEASSERT_AFTER_LAST)

Convenience function for unidirectional receive.

- virtual bool RegisterSem (OS_SEM *finishedSem)

  Register a semaphore for the SPI module.

- virtual bool ClrSem ()

  Clear a semaphore registration.

- virtual OS_SEM * GetSem ()

  Obtain a pointer to the SPI finished semaphore.

- virtual bool Done ()

  Function to check SPI status.

- virtual uint32_t GetActualBaudrate ()

  Returns the active baud rate.

- virtual bool SetCS (uint8_t CS)

  Set the chip select configuration for the SPI object's bus transactions.

Static Public Member Functions

- static BOOL Done (uint8_t SPIModule)

  Static function to check SPI status.

23.70.1 Detailed Description

SPI Peripheral Module Class.
Class definition for SPI module objects

23.70.2 Constructor & Destructor Documentation

23.70.2.1 SPIModule() [1/2]

```
SPIModule::SPIModule (
            uint8_t SPIModule )
```

Create a SPI object.
This constructor will create the object, but will *not* initialize the SPI module. You must call the Init() member function to initialize.

Parameters

SPIModule	SPI module number (0 or 1)

23.70.2.2 SPIModule() [2/2]

```
SPIModule::SPIModule (
            uint8_t SPIModule,
            uint32_t baudRateInBps,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )
```

Create a SPI object and Initialize with parameters.

Creaets a SPI object and initializes the SPI module. You do not need to call the Init() member function.

Parameters

SPIModule	SPI module number (0 or 1)
baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8, 16 or 32
peripheralChipSelects	SPI chip selects to use for transfer spiChipSelect
chipSelectPolarity	Parameter not supported on the MODM7AE70. Chip select asserts with logic level low. spiChipSelectPolarity
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	Delay from chip select to valid clock (default is 0)
delayAfterTransfer	Chip select mode dspiChipSelectMode

23.70.3 Member Function Documentation

23.70.3.1 ClrSem()

```
virtual bool SPIModule::ClrSem ( )  [inline], [virtual]
```

Clear a semaphore registration.

Returns

true if the clear was successful, false if a SPI transaction is in progress

23.70.3.2 Done() [1/2]

```
virtual bool SPIModule::Done ( )  [inline], [virtual]
```

Function to check SPI status.
Called as a class method on a specific SPI object. For example: MySpi.Done()

Returns

true if SPI is finished, false if active

23.70.3.3 Done() [2/2]

```
static BOOL SPIModule::Done (
            uint8_t SPIModule )  [static]
```

Static function to check SPI status.
Is called as a class wide static method. For example: SPIModule::Done(0)

Parameters

SPIModule	The SPI module to check, 0, 1 or 2

Returns

true if SPI is finished, false if active

23.70.3.4 GetActualBaudrate()

```
virtual uint32_t SPIModule::GetActualBaudrate ( )  [inline], [virtual]
```

Returns the active baud rate.
The baud rate will be set to the value specified when the SPI module is initialized. If that value is not possible, the next lowest baud rate will be chosen.

Returns

The actual SPI module baud rate

23.70.3.5 GetSem()

```
virtual OS_SEM * SPIModule::GetSem ( )  [inline], [virtual]
```

Obtain a pointer to the SPI finished semaphore.

Returns

A pointer to the semaphore

23.70.3.6 Init()

```
virtual uint8_t SPIModule::Init (
            uint32_t baudRateInBps = 2000000,
            uint8_t transferSizeInBits = 8,
            uint8_t peripheralChipSelects = 0x00,
            uint8_t chipSelectPolarity = 0x0F,
            uint8_t clockPolarity = 0,
            uint8_t clockPhase = 1,
            BOOL doutHiz = TRUE,
            uint8_t csToClockDelay = 0,
            uint8_t delayAfterTransfer = 0 )  [virtual]
```

Initialize an existing SPI object.

Parameters

baudRateInBps	Maximum baud rate requested
transferSizeInBits	Number of bits per transfer: 8, 16 or 32
peripheralChipSelects	SPI chip selects to use for transfer spiChipSelect
chipSelectPolarity	Parameter not supported on the MODM7AE70. Chip select asserts with logic level low. spiChipSelectPolarity
clockPolarity	0 = inactive logic level low, 1 = high
clockPhase	0 = data captured leading edge clock, changed following edge. 1 = data changed leading edge clock, captured following edge.
doutHiz	Data output high impedance between transfers
csToClockDelay	Delay from chip select to valid clock (default is 0)
delayAfterTransfer	Chip select mode dspiChipSelectMode

Returns

The current state of the SPI bus dspiState

Reimplemented in SPI_QSPI, and SPI_USART.

23.70.3.7 RegisterSem()

```
virtual bool SPIModule::RegisterSem (
            OS_SEM * finishedSem )  [virtual]
```

Register a semaphore for the SPI module.
The SPI module will post to this semaphore when a transaction is complete

Parameters

finishedSem	Pointer to the semaphore

Returns

true if the registration was successful, false if a SPI transaction is in progress

23.70.3.8 Rx()

```
virtual uint8_t SPIModule::Rx (
            uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline], [virtual]
```

Convenience function for unidirectional receive.

Parameters

∗receiveBufferPtr	Pointer to the buffer to store incomeing data
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented in SPI_QSPI, and SPI_USART.

23.70.3.9 SetBusSpeed()

```
virtual uint32_t SPIModule::SetBusSpeed (
            uint32_t maxSpeed )  [virtual]
```

Set the SPI bus speed Will attempt to set the desired bus speed. It may be different based on the available system characteristics.

Parameters

maxSpeed	The maximum desired bus speed

Returns

The actual bus speed

Reimplemented in SPI_QSPI, and SPI_USART.

23.70.3.10 SetCS()

```
virtual bool SPIModule::SetCS (
            uint8_t CS )  [inline], [virtual]
```

Set the chip select configuration for the SPI object's bus transactions.

Parameters

CS	Chip select to set, 0 - 3.

Returns

true if successful, false if SPI is currently active

Reimplemented in SPI_QSPI, and SPI_USART.

23.70.3.11 Start()

```
virtual uint8_t SPIModule::Start (
            uint8_t * transmitBufferPtr,
            volatile uint8_t * receiveBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [virtual]
```

Start a SPI transfer.

- If configured for 8 bits per transfer then the data must be uint8_t aligned
- If configured for > than 8 bits per transfer then the data must be uint16_t aligned
- If configured for > than 16 bits per transfer then the data must be uint32_t aligned
- If either RX or TX pointer is assigned 'null' then that communication direction will not occur.
- If DSPI_Finished points to a semaphore, then the DSPI will POST to it when the transfer is complete.
- The semaphore is optional, but it can increase efficiency.

Parameters

transmitBufferPtr	Pointer to the buffer containing the data to transmit
receiveBufferPtr	Pointer to the buffer to store the received data
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented in SPI_QSPI, and SPI_USART.

23.70.3.12 Tx()

```
virtual uint8_t SPIModule::Tx (
            uint8_t * transmitBufferPtr,
            uint32_t byteCount,
            int csReturnToInactive = DEASSERT_AFTER_LAST )  [inline], [virtual]
```

Convenience function for unidirectional transmit.

Parameters

∗transmitBufferPtr	Pointer to the buffer containing the data to transmit
byteCount	Number of bytes to transmit
csReturnToInactive	Chip select state

Returns

The current state of the SPI bus dspiState

Reimplemented in SPI_QSPI, and SPI_USART.
The documentation for this class was generated from the following file:

- cortex-m7/cpu/SAME70/include/dspi.h

23.71 SSC_cfg_t Struct Reference

Configuration structure for the SSC driver. Passed to the initialize function to configure the hardware.
`#include <ssc_i2s.h>`

Public Attributes

- uint16_t clkDiv {150}
- SSC_rxtx_cfg_t rx
- SSC_rxtx_cfg_t tx

23.71.1 Detailed Description

Configuration structure for the SSC driver. Passed to the initialize function to configure the hardware.

23.71.2 Member Data Documentation

23.71.2.1 clkDiv

`SSC_cfg_t::clkDiv {150}`
The divider value for the peripheral clock used to generate the bit clock when CLK_SRC_MCK is selected.

23.71.2.2 rx

`SSC_cfg_t::rx`
The receive configuration.

23.71.2.3 tx

`SSC_cfg_t::tx`
The transmit configuration.
The documentation for this struct was generated from the following file:

- ssc_i2s.h

23.72 SSC_rxtx_cfg_t Struct Reference

Configuration structure for a given direction (rx or tx) of the SSC module. Passed to the initialize function to configure the hardware.
`#include <ssc_i2s.h>`

Public Attributes

- bool enable {false}
- uint8_t period {64}
- uint8_t startDly {1}
- startCond_t startCond {START_FRAME_EDGE}
- clkGate_t clkGate {CLK_GATE_CONTINUOUS}
- dataValid_t dataValid {DATA_VALID_RISING}
- clkOut_t clkOut {CLK_OUT_INPUT}
- clkSrc_t clkSrc {CLK_SRC_RK}
- uint8_t syncLen {0}
- frameEdge_t syncEdge {FRAME_SYNC_FALLING}
- bool syncDataEnabled {false}
- frameSyncOut_t syncOut {FRAME_SYNC_INPUT}
- uint8_t wordsPerFrame {1}
- bitOrder_t bitOrder {MOST_SIG_FIRST}
- bool lineIdleState {1}
- uint8_t bitsPerWord {24}
- bufferDepletionBehavior_t depletionBehavior {DEPLETED_PAUSE}

23.72.1 Detailed Description

Configuration structure for a given direction (rx or tx) of the SSC module. Passed to the initialize function to configure the hardware.

23.72.2 Member Data Documentation

23.72.2.1 bitOrder

`SSC_rxtx_cfg_t::bitOrder {MOST_SIG_FIRST}`
The bit order of transitions.

23.72.2.2 bitsPerWord

`SSC_rxtx_cfg_t::bitsPerWord {24}`
The bit width of words to be transferred.

23.72.2.3 clkGate

`SSC_rxtx_cfg_t::clkGate {CLK_GATE_CONTINUOUS}`
The clock gating condition for this channel.

23.72.2.4 clkOut

`SSC_rxtx_cfg_t::clkOut {CLK_OUT_INPUT}`
The clock out condition for this channel.

23.72.2.5 clkSrc

`SSC_rxtx_cfg_t::clkSrc {CLK_SRC_RK}`
The clock source for this channel.

23.72.2.6 dataValid

`SSC_rxtx_cfg_t::dataValid {DATA_VALID_RISING}`
The data valid condition for this channel.

23.72.2.7 depletionBehavior

`SSC_rxtx_cfg_t::depletionBehavior {DEPLETED_PAUSE}`
The depletion behavior that should be used for driver DMA transfers.

23.72.2.8 enable

`SSC_rxtx_cfg_t::enable {false}`
Whether this direction should be enabled.

23.72.2.9 lineIdleState

`SSC_rxtx_cfg_t::lineIdleState {1}`
The state the line should be driven to when not transmitting a word.

23.72.2.10 period

`SSC_rxtx_cfg_t::period {64}`
The period of each frame, in terms of the bit clock.

23.72.2.11 startCond

`SSC_rxtx_cfg_t::startCond {START_FRAME_EDGE}`
The starting condition for this channel.

23.72.2.12 startDly

`SSC_rxtx_cfg_t::startDly {1}`
How many bit periods should data be delayed by at the start of the frame.

23.72.2.13 syncDataEnabled

`SSC_rxtx_cfg_t::syncDataEnabled {false}`
Whether synchronizing data is used.

23.72.2.14 syncEdge

`SSC_rxtx_cfg_t::syncEdge {FRAME_SYNC_FALLING}`
The frame sync edge defining frame start.

23.72.2.15 syncLen

`SSC_rxtx_cfg_t::syncLen {0}`
The bit length of the synchronization data (when used).

23.72.2.16 syncOut

`SSC_rxtx_cfg_t::syncOut {FRAME_SYNC_INPUT}`
The frame sync output configuration.

23.72.2.17 wordsPerFrame

`SSC_rxtx_cfg_t::wordsPerFrame {1}`
Number of words contained in a frame.
The documentation for this struct was generated from the following file:

- ssc_i2s.h

23.73 SSCCtx_t Class Reference

`#include <ssc_i2s.h>`

Public Member Functions

- int Init (const SSC_cfg_t &cfg)

  Initializes the SSC hardware and driver context.
- void **Shutdown** ()

  Shuts down the SSC hardware and driver.
- int getCurrentConfig (SSC_cfg_t &cfg)

  Fills in the config object with the current active configuration.
- ctxState_t getState ()

  Returns the current driver state.
- int TransmitBuffer (void *buffer, uint32_t bufferLen, bool waitIfNeeded)

  Hands off a buffer to be transmitted by the SSC driver.
- int ReadyReceiveBuffer (void *buffer, uint32_t bufferLen, bool waitIfNeeded)

  Hands off a buffer to be written to by the SSC driver.
- void **RegisterTxBufferDoneCB** (SSC_BufferDoneFn_t cb)

  Registers a callback for when a transmit buffer is finished.
- void **RegisterRxBufferDoneCB** (SSC_BufferDoneFn_t cb)

  Registers a callback for when a receive buffer is finished.

23.73.1 Detailed Description

SSCtcx_t is a driver for the Synchronous Serial Controller. It operates on a buffer level, where buffers are handed to the system and either transmitted or filled with received data as appropriate.

23.73.2 Member Function Documentation

23.73.2.1 getCurrentConfig()

```
int SSCCtx_t::getCurrentConfig (
            SSC_cfg_t & cfg )
```

Fills in the config object with the current active configuration.

Parameters

out	*cfg*	The configuration object to populate.

Returns

Negative on failure.

23.73.2.2 getState()

```
ctxState_t SSCCtx_t::getState ( )
```

Returns the current driver state.

Returns

The current driver state.

23.73.2.3 Init()

```
int SSCCtx_t::Init (
            const SSC_cfg_t & cfg )
```

Initializes the SSC hardware and driver context.

Parameters

in	*cfg*	The configuration to use.

Returns

Negative on failure.

23.73.2.4 ReadyReceiveBuffer()

```
int SSCCtx_t::ReadyReceiveBuffer (
            void * buffer,
            uint32_t bufferLen,
            bool waitIfNeeded )
```

Hands off a buffer to be written to by the SSC driver.

Parameters

buffer	A pointer to the buffer to be written to.
bufferLen	The length of the buffer to be written. (Must be multiples of 1, 2, or 4 bytes depending on word bit width)
waitIfNeeded	Whether the driver should wait for space to receive or fail immediately upon exhausting the queue depth.

Returns

Negative on failure.

23.73.2.5 TransmitBuffer()

```
int SSCCtx_t::TransmitBuffer (
            void * buffer,
            uint32_t bufferLen,
            bool waitIfNeeded )
```

Hands off a buffer to be transmitted by the SSC driver.

Parameters

buffer	A pointer to the buffer to be transmit.
bufferLen	The length of the buffer to be transmit. (Must be multiples of 1, 2, or 4 bytes depending on word bit width)
waitIfNeeded	Whether the driver should wait for space to transmit or fail immediately upon exhausting the queue depth.

Returns

Negative on failure.

The documentation for this class was generated from the following files:

- ssc_i2s.h
- ssc_i2s.cpp

23.74 StopWatch Class Reference

Stopwatch for timing events.
`#include <stopwatch.h>`

Public Member Functions

- StopWatch (int timer_number=FIRST_UNUSED_TIMER)
  Create a stopwatch object.
- void Start ()
  Start the stopwatch.
- void Clear ()
  Clear the stopwatch count.
- unsigned long long Stop ()
  Stop the StopWatch.
- unsigned long long GetTime ()
  Get the StopWatch time.
- double CountResolution ()
  Returns the value of one StopWatch tick time.
- double Convert (unsigned long long)
  Convert a time or interval from an unsigned long long to double floatl.

23.74.1 Detailed Description

Stopwatch for timing events.

23.74.2 Constructor & Destructor Documentation

23.74.2.1 StopWatch()

```
StopWatch::StopWatch (
            int timer_number = FIRST_UNUSED_TIMER )
```

Create a stopwatch object.
The timer number can be ignored on platforms that have a permanently allocated timer for the RTOS and CPU frequency.

Parameters

timer_number	Optional. Specifies the timer to use. If no timer is specified, the first unused timer will be selected.

23.74.3 Member Function Documentation

23.74.3.1 Clear()

```
void StopWatch::Clear ( )
```

Clear the stopwatch count.
Clear the StopWatch count

23.74.3.2 Convert()

```
double StopWatch::Convert (
            unsigned long long  )
```

Convert a time or interval from an unsigned long long to double floatl.

Return values

Elapsed	time as a floating point number

23.74.3.3 CountResolution()

```
double StopWatch::CountResolution ( )
```

Returns the value of one StopWatch tick time.

Return values

The	value of one stopwatch tick

23.74.3.4 GetTime()

```
unsigned long long StopWatch::GetTime ( )
```

Get the StopWatch time.

Return values

Elapsed	time. Will work even if stopwatch is running.

23.74.3.5 Start()

```
void StopWatch::Start ( )
```

Start the stopwatch.
Start the StopWatch timer

23.74.3.6 Stop()

```
unsigned long long StopWatch::Stop ( )
```

Stop the StopWatch.

Return values

Elapsed	time

The documentation for this class was generated from the following file:

- stopwatch.h

23.75 TickTimeout Class Reference

TickTimeout objects are used to facilitate sequential function calls with timeout parameters that need to indexed from an initial start time, and to prevent TimeTick rollover errors.
`#include <nbrtos.h>`

Public Member Functions

- TickTimeout (uint32_t timeout)

  Create and initialize the Timeout.
- uint32_t val () const

  Get the timeout duration to be passed to a function utilizing timeout ticks.
- bool expired () const

  Determine whether the timeout duration has elapsed.
- operator bool () const

  Use the Timeout as a boolean (such as in a branch condition).
- void SetUntil (uint32_t when)

  Set the TimeTick value to expire.

Friends

- void OSTimeWaitUntil (uint32_t systemTickValue)

  Delay the task until the specified value of the system timer tick. The number of system ticks per second is defined by the constant: `TICKS_PER_SECOND` in `<nburn_install>/nbrtos/include/constants.h`. The default value is 20 ticks per second.

23.75.1 Detailed Description

TickTimeout objects are used to facilitate sequential function calls with timeout parameters that need to indexed from an initial start time, and to prevent TimeTick rollover errors.

23.75.2 Constructor & Destructor Documentation

23.75.2.1 TickTimeout()

```
TickTimeout::TickTimeout (
            uint32_t timeout )  [inline]
```

Create and initialize the Timeout.

Parameters

timeout	The maximum allowed time duration in ticks. **Note:** The maximum delay that can be used is INT32_MAX

23.75.3 Member Function Documentation

23.75.3.1 expired()

```
bool TickTimeout::expired ( ) const  [inline]
```

Determine whether the timeout duration has elapsed.

Return values

true	The timeout has expired
false	The timeout has not expired

23.75.3.2 operator bool()

```
TickTimeout::operator bool ( ) const  [inline]
```

Use the Timeout as a boolean (such as in a branch condition).
Since the intention is to be used for while() loop or in a precondition check, the boolean value represents whether the duration is still valid and has not expired. **As a result this is the opposite behavior of the `expired()` method**.

Return values

true	The Timeout duration is still valid
false	The Timeout duration is expired

23.75.3.3 SetUntil()

```
void TickTimeout::SetUntil (
            uint32_t when )  [inline]
```

Set the TimeTick value to expire.

Parameters

when	TimeTick value in which to expire

23.75.3.4 val()

```
uint32_t TickTimeout::val ( ) const  [inline]
```

Get the timeout duration to be passed to a function utilizing timeout ticks.

Returns

The timeout duration in TimeTicks

23.75.4 Friends And Related Function Documentation

23.75.4.1 OSTimeWaitUntil

```
void OSTimeWaitUntil (
            uint32_t systemTickValue )  [friend]
```

Delay the task until the specified value of the system timer tick. The number of system ticks per second is defined by the constant: TICKS_PER_SECOND in <nburn_install>/nbrtos/include/constants.h. The default value is 20 ticks per second.

Parameters

systemTickValue	The system time tick value to wait for

```
uint32_t Now = TimeTick;
while(1)
{
   Now += 60 * TICKS_PER_SECOND;
   OSTimeWaitUntil(Now);
   // Do whatever every 60 seconds, no mater how long this takes to run the interval will have a consistant
      spacing
}
```

See also

OSChangeTaskDly(), OSTimeDly()

The documentation for this class was generated from the following file:

- nbrtos.h

23.76 UDPPacket Class Reference

UDP Packet Class.
#include <udp.h>

Public Member Functions

- UDPPacket (OS_FIFO *pFifo, TickTimeout timeout)

  Constructor to create a UDP Packet object from a UDP FIFO entry.
- UDPPacket (int sock)

  Constructor to create a UDP Packet object from an open UDP socket.
- UDPPacket (PoolPtr p)

  Constructor to create a UDP packet from a system pool buffer.
- UDPPacket (UDPPacket &pkt)

  Constructor to create a new UDP packet from an existing UDP packet.
- ~UDPPacket ()

  UDP packet object destructor. Frees any associated memory.
- void SetSourcePort (uint16_t port)

  Set the source port number of a UDP Packet object.
- uint16_t GetSourcePort (void) const

  Get the source port number of a UDP Packet object.
- MACADR GetMacSource ()

  Get the source MAC address a UDP Packet object.
- IPADDR GetSourceAddress (void)

  Get the source IP address a UDP Packet object.
- IPADDR GetDestinationAddress (void)

  Get the destination IP address a UDP Packet object.
- bool bIsIPV6 ()

  Check if the IPADDR holds an IPv6 IP address.
- void SetDestinationPort (uint16_t port)

  Set the destination port number of a UDP Packet object.
- uint16_t GetDestinationPort (void) const

  Get the destination port number of a UDP Packet object.

- uint16_t GetPacketId (void)

  Get UDP packet ID.
- puint8_t GetDataBuffer (bool bReAllocateIfNeeded=false)

  Get a pointer to the UDP Packet object's data buffer.
- void SetDataSize (uint16_t numBytes)

  Set the UDP Packet data size.
- uint16_t GetDataSize (void) const

  Get the UDP Packet object data size.
- void AddData (puint8_t pData, uint16_t len)

  Add a number of data bytes to a UDP Packet object.
- void AddData (PCSTR pData)

  Add data to a UDP Packet object as a NULL terminated ASCII string.
- void AddDataWord (uint16_t w)

  Add a 16-bit unsigned integer to a UDP Packet object.
- void AddDataByte (uint8_t b)

  Add an 8-bit unsigned integer to a UDP Packet object.
- BOOL Validate (void)

  Verify a received UDP packet is valid.
- void ResetData (void)

  Set the data size of a UDP Packet object to 0.
- void SendAndKeep (const IPADDR &to, uint8_t ttl=0)

  Make a copy of a UDP Packet and send it. The original packet will remain intact.
- void Send (const IPADDR &to, uint8_t ttl=0)

  Send the UDP Packet and free the pool buffer.
- void SendAndKeepViaInterfaceNum (const IPADDR &to, int interface, uint8_t ttl=0)

  Make a copy of a UDP Packet and send it using the specified network interface. The original packet will remain intact.
- void SendViaInterfaceNum (const IPADDR &to, int interface, uint8_t ttl=0)

  Send the UDP Packet using the specified network interface and free the pool buffer.
- void SendAndKeepViaIfAddr (const IPADDR &to, const IPADDR &from_ip, uint8_t ttl=0)

  Make a copy of a UDP Packet and send it through the network interface specified by the from_ip IP address parameter. If more than one interface has the same IP address, the lower interface number will be used. The original packet will remain intact.
- void SendViaIfAddr (const IPADDR &to, const IPADDR &from_ip, uint8_t ttl=0)

  Send a UDP packet through the network interface specified by the from_ip IP address parameter. If more than one interface has the same IP address, the lower interface number will be used. The UDP pool buffer will be freed.

23.76.1 Detailed Description

UDP Packet Class.
This class holds PoolBuffers and treats them as UDP Packets. The process for sending a UDP packet is as follows:

```
UDPPacket myUdpPacket;                  // Create a UDPPacket object

myUdpPacket.SetSourcePort(123);         // Set the source port number
myUdpPacket.SetDestinatioPort(456);     // Set destination port number
```

Data can be put in the packet two ways:

1. Obtain a pointer to the object's data buffer and copy the data in. Be certain to set the data length.
   ```
   uint16_t len = siprintf( myUdpPacket.GetDataBuffer(), "Using sprintf() to copy data at time = %ld",
   TickCount );
   myUdpPacket.SetDataSize(len);
   ```

2. Copy the data directly to the packet buffer:
   ```
   AddData("This is the data to add");        // Add a constant ASCII null terminated string
   AddData(pData, len);                       // Add with a pointer to the data and the data length
   ```

When the UDP packet has been configured it can be sent two ways:

1. Send and keep the packet and keep the constructed pool buffer: `SendAndKeep(IPADDR destination↩IP);`. You must free the buffer manually.

2. Send and automatically free the pool buffer (recommended and more efficient): `Send(IPADDR4 destinationIP);`

There are many UDP examples, and we recommend reviewing them to determine the best method to use for your application.

23.76.2 Constructor & Destructor Documentation

23.76.2.1 UDPPacket() [1/4]

```
UDPPacket::UDPPacket (
            OS_FIFO * pFifo,
            TickTimeout timeout )
```

Constructor to create a UDP Packet object from a UDP FIFO entry.
UDP packets are received in a OS_FIFO. This constructor create a UDP packet from the next entry in the FIFO, which also removes the entry from the FIFO. If there is not a packet in the FIFO, this constructor will block until one is available, or the specified timeout occurs. If a timeout occurs, an invalid UDP packet will be created. The UDP Validate() function must always be called after this constructor to verify a valid UDP packet has been created.

Parameters

pFifo	Pointer to an OS_FIFO object
timeout	Timeout in system Time Ticks

See also

UDPPacket(int sock), UDPPacket(PoolPtr p)

23.76.2.2 UDPPacket() [2/4]

```
UDPPacket::UDPPacket (
            int sock )
```

Constructor to create a UDP Packet object from an open UDP socket.
This constructor will block until a UDP packet is received. It is useful for situations such as a select() call that will block on a UDP file descriptor until a packet is received. After the select() returns, this constructor can be used to create the packet.

Parameters

sock	Open UDP socket

See also

UDPPacket(OS_FIFO *pFifo, uint32_t timeout), UDPPacket(PoolPtr p)

23.76.2.3 UDPPacket() [3/4]

```
UDPPacket::UDPPacket (
            PoolPtr p )
```

Constructor to create a UDP packet from a system pool buffer.
This constructor can be used in unique situations in which an application is operating directly on the system pool buffers. This is a rare situation.

Parameters

p	Pointer to a pool buffer

See also

UDPPacket(OS_FIFO *pFifo, uint32_t timeout), UDPPacket(int sock)

23.76.2.4 UDPPacket() [4/4]

```
UDPPacket::UDPPacket (
            UDPPacket & pkt )
```

Constructor to create a new UDP packet from an existing UDP packet.
The UDP packet passed in the argument will be destroyed.

Parameters

pkt	UDP packet object

See also

UDPPacket(OS_FIFO *pFifo, uint32_t timeout), UDPPacket(int sock)

23.76.2.5 ~UDPPacket()

```
UDPPacket::~UDPPacket ( )
```

UDP packet object destructor. Frees any associated memory.

See also

UDPPacket(OS_FIFO *pFifo, uint32_t timeout), UDPPacket(int sock), UDPPacket(PoolPtr p)

23.76.3 Member Function Documentation

23.76.3.1 AddData() [1/2]

```
void UDPPacket::AddData (
            PCSTR pData )
```

Add data to a UDP Packet object as a NULL terminated ASCII string.

Parameters

pData	Pointer to the null terminated string

See also

AddDataByte(), AddDataWord()

23.76.3.2 AddData() [2/2]

```
void UDPPacket::AddData (
            puint8_t pData,
```

```
            uint16_t len )
```

Add a number of data bytes to a UDP Packet object.

Parameters

pData	Pointer to the data to add
len	Number of bytes

See also

AddDataByte(), AddDataWord()

23.76.3.3 AddDataByte()

```
void UDPPacket::AddDataByte (
            uint8_t b )
```

Add an 8-bit unsigned integer to a UDP Packet object.

Parameters

b	8-bit unsigned integer

See also

AddDataWord(), AddData()

23.76.3.4 AddDataWord()

```
void UDPPacket::AddDataWord (
            uint16_t w )
```

Add a 16-bit unsigned integer to a UDP Packet object.

Parameters

w	16 bit unsigned integer

See also

AddDataByte(), AddData()

23.76.3.5 bIsIPV6()

```
bool UDPPacket::bIsIPV6 ( )  [inline]
```

Check if the IPADDR holds an IPv6 IP address.

Returns

true if the IP address is an IPv6 address

23.76.3.6 GetDataBuffer()

```
puint8_t UDPPacket::GetDataBuffer (
            bool bReAllocateIfNeeded = false )
```

Get a pointer to the UDP Packet object's data buffer.

Returns

Pointer to the data buffer

23.76.3.7 GetDataSize()

```
uint16_t UDPPacket::GetDataSize (
            void  ) const
```

Get the UDP Packet object data size.

Returns

The data size as number of bytes

See also

SetDataSize()

23.76.3.8 GetDestinationAddress()

```
IPADDR UDPPacket::GetDestinationAddress (
            void  )  [inline]
```

Get the destination IP address a UDP Packet object.

Returns

The destination IP address

23.76.3.9 GetDestinationPort()

```
uint16_t UDPPacket::GetDestinationPort (
            void  ) const
```

Get the destination port number of a UDP Packet object.

Returns

The destination port number

See also

SetDestinationPort()

23.76.3.10 GetMacSource()

```
MACADR UDPPacket::GetMacSource ( )
```

Get the source MAC address a UDP Packet object.

Returns

The source MAC address

23.76.3.11 GetPacketId()

```
uint16_t UDPPacket::GetPacketId (
            void  )
```

Get UDP packet ID.

Returns

The UDP packet ID

23.76.3.12 GetSourceAddress()

```
IPADDR UDPPacket::GetSourceAddress (
            void  )  [inline]
```

Get the source IP address a UDP Packet object.

Returns

The source IP address

23.76.3.13 GetSourcePort()

```
uint16_t UDPPacket::GetSourcePort (
            void  ) const
```

Get the source port number of a UDP Packet object.

Returns

The source port number

See also

SetSourcePort()

23.76.3.14 ResetData()

```
void UDPPacket::ResetData (
            void  )
```

Set the data size of a UDP Packet object to 0.

23.76.3.15 Send()

```
void UDPPacket::Send (
            const IPADDR & to,
            uint8_t ttl = 0 )  [inline]
```

Send the UDP Packet and free the pool buffer.

Parameters

to	Destination IP address
ttl	Optional. If not specified the system default will be used

See also

SendAndKeep(), SendViaInterfaceNum(), SendViaIfAddr()

23.76.3.16 SendAndKeep()

```
void UDPPacket::SendAndKeep (
            const IPADDR & to,
            uint8_t ttl = 0 )  [inline]
```

Make a copy of a UDP Packet and send it. The original packet will remain intact.

Parameters

to	Destination IP address
ttl	Optional. If not specified the system default will be used

See also

Send(), SendViaInterfaceNum(), SendAndKeepViaIfAddr()

23.76.3.17 SendAndKeepViaIfAddr()

```
void UDPPacket::SendAndKeepViaIfAddr (
            const IPADDR & to,
            const IPADDR & from_ip,
            uint8_t ttl = 0 )  [inline]
```

Make a copy of a UDP Packet and send it through the network interface specified by the from_ip IP address parameter. If more than one interface has the same IP address, the lower interface number will be used. The original packet will remain intact.

Parameters

to	Destination IP address
from_ip	IP address to identify the local network interface as the source
ttl	Optional. If not specified the system default will be used

See also

Send(), SendViaInterfaceNum()

23.76.3.18 SendAndKeepViaInterfaceNum()

```
void UDPPacket::SendAndKeepViaInterfaceNum (
            const IPADDR & to,
            int interface,
            uint8_t ttl = 0 )  [inline]
```

Make a copy of a UDP Packet and send it using the specified network interface. The original packet will remain intact.

Parameters

to	Destination IP address
interface	Interface number

Parameters

ttl	Optional. If not specified the system default will be used

See also

Send(), SendAndKeep(), SendViaInterfaceNum(), SendAndKeepViaIfAddr()

23.76.3.19 SendViaIfAddr()

```
void UDPPacket::SendViaIfAddr (
            const IPADDR & to,
            const IPADDR & from_ip,
            uint8_t ttl = 0 )  [inline]
```

Send a UDP packet through the network interface specified by the from_ip IP address parameter. If more than one interface has the same IP address, the lower interface number will be used. The UDP pool buffer will be freed.

Parameters

to	Destination IP address
from_ip	IP address to identify the local network interface as the source
ttl	Optional. If not specified the system default will be used

See also

Send(), SendAndKeep(), SendAndKeepViaIfAddr()

23.76.3.20 SendViaInterfaceNum()

```
void UDPPacket::SendViaInterfaceNum (
            const IPADDR & to,
            int interface,
            uint8_t ttl = 0 )  [inline]
```

Send the UDP Packet using the specified network interface and free the pool buffer.

Parameters

to	Destination IP address
interface	Interface number
ttl	Optional. If not specified the system default will be used

See also

Send(), SendAndKeep(), SendAndKeepViaIfAddr()

23.76.3.21 SetDataSize()

```
void UDPPacket::SetDataSize (
            uint16_t numBytes )
```

Set the UDP Packet data size.

Parameters

numBytes	Data size as number of bytes

See also

GetDataSize()

23.76.3.22 SetDestinationPort()

```
void UDPPacket::SetDestinationPort (
            uint16_t port )
```

Set the destination port number of a UDP Packet object.

Parameters

port	Destination port number

See also

GetDestinationPort()

23.76.3.23 SetSourcePort()

```
void UDPPacket::SetSourcePort (
            uint16_t port )
```

Set the source port number of a UDP Packet object.

Parameters

port	Source port number

See also

GetSourcePort()

23.76.3.24 Validate()

```
BOOL UDPPacket::Validate (
            void  )
```

Verify a received UDP packet is valid.
Verifies a received UDP packet has data and validates the checksum. This function should be called anytime a UDP packet is received.

Returns

true if packet is valid, otherwise false

The documentation for this class was generated from the following file:

- udp.h

23.77 UniqueIdentifier Class Reference

Get the 128-bit Unique Identifier.

```
#include <bsp.h>
```

Public Member Functions

- const uint8_t * GetUniqueIdentifier ()

  Will attempt to read the Unique Identifier from flash memory.
- **UniqueIdentifier** ()

  Create a UniqueIdentifier object and read the Unique Identifier from flash memory.
- const uint8_t * GetBuffer ()

  Get a pointer to the buffer containing the uniqueID.
- void **Print** ()

  Print the 128-bit UniqueID in hex to serial.
- const uint8_t * GetUniqueIdentifier ()

  Will attempt to read the Unique Identifier from flash memory.
- **UniqueIdentifier** ()

  Create a UniqueIdentifier object and read the Unique Identifier from flash memory.
- const uint8_t * GetBuffer ()

  Get a pointer to the buffer containing the uniqueID.
- void **Print** ()

  Print the 128-bit UniqueID in hex to serial.

23.77.1 Detailed Description

Get the 128-bit Unique Identifier.
Each MODM7AE70 is programmed with a 128-bit unique identifier in flash memory. This class exposes functions to get the 128-bit unique ID in a 16-byte array of type uint8_t.
Each SBE70LC is programmed with a 128-bit unique identifier in flash memory. This class exposes functions to get the 128-bit unique ID in a 16-byte array of type uint8_t.

23.77.2 Member Function Documentation

23.77.2.1 GetBuffer() [1/2]

```
const uint8_t * UniqueIdentifier::GetBuffer ( )  [inline]
```

Get a pointer to the buffer containing the uniqueID.

Returns

A pointer to the buffer containing the uniqueID.

23.77.2.2 GetBuffer() [2/2]

```
const uint8_t * UniqueIdentifier::GetBuffer ( )  [inline]
```

Get a pointer to the buffer containing the uniqueID.

Returns

A pointer to the buffer containing the uniqueID.

23.77.2.3 GetUniqueIdentifier() [1/2]

```
const uint8_t * UniqueIdentifier::GetUniqueIdentifier ( )
```
Will attempt to read the Unique Identifier from flash memory.

Returns

If successful, returns a pointer to the buffer containing the uniqueID. Otherwise, returns a nullptr.

23.77.2.4 GetUniqueIdentifier() [2/2]

```
const uint8_t * UniqueIdentifier::GetUniqueIdentifier ( )
```
Will attempt to read the Unique Identifier from flash memory.

Returns

If successful, returns a pointer to the buffer containing the uniqueID. Otherwise, returns a nullptr.

The documentation for this class was generated from the following files:

- MODM7AE70/include/bsp.h
- SBE70LC/include/bsp.h

23.78 UserAuthManager Class Reference

The user authorization manager class allows application developers the ability to manage user authorization records. The can be loaded and saved to any storage space, including the config system or UserParams. Authorization values are hashed before being saved. Validation compares both the hash as well as the authorization type. Adding, updating, and removing records will automatically call the user devined save functions. For usage, please see the example found in `examples/SSH/sshServerUserAuth`.
`#include <UserAuthManager.h>`

Public Member Functions

- **UserAuthManager** ()

  Default constructor.
- ~**UserAuthManager** ()

  Default destructor.
- bool Init (SaveAuthRecordsFn svRcFn, LoadAuthRecordsFn ldRcFn)

  Initialization function. Must be called before use.
- bool UserExists (const NBString &userName)

  Determines if a user record exists.
- AuthResponse AddUserAuth (const NBString &userName, const NBString &auth, AuthType authType)

  Attempts to add a user authorization record. This will automatically call the save record function if the user is added.
- AuthResponse CheckUserAuth (const NBString &userName, const NBString &auth, AuthType authType)

  Checks the for a user and compares the authorization value to what is stored.
- AuthResponse CheckUserAuth (const NBString &userName, byte *auth, AuthType authType)

  Checks the for a user and compares the authorization value to what is stored.
- AuthResponse UpdateUserAuth (const NBString &userName, const NBString &newAuth, AuthType auth↩ Type)

  Updates a user authorization record with the information provided. This function will automatically save all user records if the changes are successful.
- AuthResponse RemoveUserAuth (const NBString &userName)

  Remove a user authorization record. This function will automatically save all user records if the removal is successful.
- AuthResponse CheckUserAuthLevel (const NBString &userName, uint32_t authLevel, bool hasAll=true)

  Checks the user against the specific authLevel.

- AuthResponse SetUserAuthLevel (const NBString &userName, uint32_t authLevel)

  This adds the authorization levels passed in to the user's current authorization level.
- AuthResponse ClrUserAuthLevel (const NBString &userName, uint32_t authLevel)

  Clears the authorization for the specified user.
- void **ListUsers** ()

  Lists the users currently in the User Authorization Record system, along with their saved authorization type and level.
- int **GetMaxAuthRecords** ()

  Gets the maximum number of authorization records available to the system. This can be changed with the macro MAX_AUTH_RECORDS.

23.78.1 Detailed Description

The user authorization manager class allows application developers the ability to manage user authorization records. The can be loaded and saved to any storage space, including the config system or UserParams. Authorization values are hashed before being saved. Validation compares both the hash as well as the authorization type. Adding, updating, and removing records will automatically call the user devined save functions. For usage, please see the example found in `examples/SSH/sshServerUserAuth`.

23.78.2 Member Function Documentation

23.78.2.1 AddUserAuth()

```
AuthResponse UserAuthManager::AddUserAuth (
            const NBString & userName,
            const NBString & auth,
            AuthType authType )
```

Attempts to add a user authorization record. This will automatically call the save record function if the user is added.

Parameters

userName	The username to add.
auth	The authorization value to hash and store.
authType	The authorization type of the authorization value.

Return values

eAuthSuccess	If the user was successfully added.
!eAuthSuccess	If the user was not added successfully. The AuthResponse error code returned will indicate why.

23.78.2.2 CheckUserAuth() [1/2]

```
AuthResponse UserAuthManager::CheckUserAuth (
            const NBString & userName,
            byte * auth,
            AuthType authType )
```

Checks the for a user and compares the authorization value to what is stored.

Parameters

userName	The username to check against.
auth	The authorization value to compare as hashed value. This will be compared directly to what is stored in the record.

Parameters

authType	The authorization type of the authorization value.

Return values

eAuthSuccess	If the user exists and the authorization value matches what was stored
!eAuthSuccess	If the user does not exist or the authorization comparison fails. The AuthResponse error code returned will indicate why.

23.78.2.3 CheckUserAuth() [2/2]

```
AuthResponse UserAuthManager::CheckUserAuth (
            const NBString & userName,
            const NBString & auth,
            AuthType authType )
```

Checks the for a user and compares the authorization value to what is stored.

Parameters

userName	The username to check against.
auth	The authorization value to compare as a plain string. This value will be hashed before getting compared.
authType	The authorization type of the authorization value.

Return values

eAuthSuccess	If the user exists and the authorization value matches what was stored
!eAuthSuccess	If the user does not exist or the authorization comparison fails. The AuthResponse error code returned will indicate why.

23.78.2.4 CheckUserAuthLevel()

```
AuthResponse UserAuthManager::CheckUserAuthLevel (
            const NBString & userName,
            uint32_t authLevel,
            bool hasAll = true )
```

Checks the user against the specific authLevel.

Parameters

userName	The username to check.
authLevel	The authorization level to check against. This check compares the bits passed in against the bits set on the user's authLevel.
hasAll	Used to establish the conditions for the check to pass. If hasAll is true, then every bit passed in must be set on the user's authLevel for the check to pass. If hasAll is false, then only one bit needs to match for the check to pass.

Return values

eAuthSuccess	If the check passed and the user has the proper authorization level.
!eAuthSuccess	If the user does not have the proper authorization level.

23.78.2.5 ClrUserAuthLevel()

```
AuthResponse UserAuthManager::ClrUserAuthLevel (
            const NBString & userName,
            uint32_t authLevel )
```

Clears the authorization for the specified user.

Parameters

userName	The username to check.
authLevel	The authorization level to clear.

Return values

eAuthSuccess	If the specified authorization level was successfully cleared and saved.
!eAuthSuccess	If the authorization level could not be cleared and saved. The AuthResponse error code returned will indicate why.

23.78.2.6 Init()

```
bool UserAuthManager::Init (
            SaveAuthRecordsFn svRcFn,
            LoadAuthRecordsFn ldRcFn )
```

Initialization function. Must be called before use.

Parameters

svRcFn	The user defined function that will save authorization records.
ldRcFn	The user defined function that will load the authorization records from storage. Called inside init().

Return values

true	If successfully initialzed, and the users are properly loaded.
!0	If there was an error initializing the object, or the users were not successfully loaded. Error code will

23.78.2.7 RemoveUserAuth()

```
AuthResponse UserAuthManager::RemoveUserAuth (
            const NBString & userName )
```

Remove a user authorization record. This function will automatically save all user records if the removal is successful.

Parameters

userName	The username to remove.

Return values

eAuthSuccess	If the user record was successfully removed.
!eAuthSuccess	If the user record could not be removed. The AuthResponse error code returned will indicate why.

23.78.2.8 SetUserAuthLevel()

```
AuthResponse UserAuthManager::SetUserAuthLevel (
            const NBString & userName,
            uint32_t authLevel )
```

This adds the authorization levels passed in to the user's current authorization level.

Parameters

userName	The username to check.
authLevel	The authorization level to check against. This check compares the bits passed in against the bits set on the user's authLevel.

Return values

eAuthSuccess	If the the user authorization level was properly set and saved.
!eAuthSuccess	If the user's authorization level could not be set and saved. The AuthResponse error code returned will indicate why.

23.78.2.9 UpdateUserAuth()

```
AuthResponse UserAuthManager::UpdateUserAuth (
            const NBString & userName,
            const NBString & newAuth,
            AuthType authType )
```

Updates a user authorization record with the information provided. This function will automatically save all user records if the changes are successful.

Parameters

userName	The username to check against.
newAuth	The new authorization value to assign to the user.
authType	The authorization type of the authorization value.

Return values

eAuthSuccess	If the user record was successfully updated.
!eAuthSuccess	If the user record could not be updated. The AuthResponse error code returned will indicate why.

23.78.2.10 UserExists()

```
bool UserAuthManager::UserExists (
            const NBString & userName )
```

Determines if a user record exists.

Parameters

userName	The username to check.

Return values

true	If the user exists.
false	If the user does not exist.

The documentation for this class was generated from the following file:

- UserAuthManager.h

23.79 UserAuthRecord Struct Reference

A stored record of a user's authorization credentials. The value is hashed when saved so it can't be read directly. User's can currently only have one authorization record per user name.
`#include <UserAuthManager.h>`

Public Attributes

- NBString **m_userName**

  The username for the record. This value must be unique across all records.
- uint8_t **m_authHash** [WC_SHA256_DIGEST_SIZE]

  The hashed representation of the authentication value.
- AuthType **m_authType**

  The type of authentication hashes. Currently supported are keys and passwords.
- uint32_t m_authLevel

23.79.1 Detailed Description

A stored record of a user's authorization credentials. The value is hashed when saved so it can't be read directly. User's can currently only have one authorization record per user name.

23.79.2 Member Data Documentation

23.79.2.1 m_authLevel

`uint32_t UserAuthRecord::m_authLevel`
The authorization level granted to this user. This is treated as a bit field, and authorization checks look for specific bits being set.
The documentation for this struct was generated from the following file:

- UserAuthManager.h

23.80 USERCritObj Class Reference

User critial section object class.
`#include <nbrtos.h>`

23.80.1 Detailed Description

User critial section object class.
The documentation for this class was generated from the following file:

- nbrtos.h

23.81 wifi_init Struct Reference

`#include <wifiDriver.h>`

23.81.1 Detailed Description

This structure is used as a WiFi initialization structure for defining pins, IRQ, and type of WiFi module being used with a specific bus type. the GetDriver function pointer must be defined to one of the supported WiFi module option's static driver function. The options are GetNewWicedSPIDriver(), GetNewWicedSerialDriver(), or GetNew↩ WilcDriver(). Default WiFiInit structures are defined using macros for each of the NBWIFI development kit options. If the user is using pin/IRQ settings that differ from the default values, the user must either define their own wifiInit structure in their project or pass in the custom values to the WiFi initialization functions as parameters.
The documentation for this struct was generated from the following file:

- wifiDriver.h

23.82 WireIntf Class Reference

Wire Interface Class for I2C.
`#include <i2c.h>`

Public Member Functions

- void begin ()

  Initialize the WireIntf driver object as a master with a bus speed of 100 kHz. This normally only needs to be called once.
- uint32_t requestFrom (uint8_t addr, uint32_t len, bool stop=true)

  Request a number of bytes from a slave device. The bytes can then be retrieved from the slave device with the read() and available() functions.
- void beginTransmission (uint8_t addr)

  Begin a transmission to a I2C slave device at the provided address. Bytes can be queued for transmission with the write() functions, and are transmitted by calling endTransmission().
- void endTransmission (bool stop=true)

  Transmits the bytes of data queued by using the write() functions, and ends a transmission to a slave device that was started using beginTransmission().
- uint32_t write (uint8_t dat)

  Queues bytes of data to be transmitted to the slave device. This function can be called after a call to beginTransmission(). The data will not be transmitted on the I2C bus until a call to endTransmission() or flush() is performed.
- uint32_t write (char ∗str)

  Queues bytes of data to be transmitted to the slave device. This function can be called after a call to beginTransmission(). The data will not be transmitted on the I2C bus until a call to endTransmission() or flush() is performed.
- uint32_t write (uint8_t ∗buf, uint32_t len)

  Queues bytes of data to be transmitted to the slave device. This function can be called after a call to beginTransmission(). The data will not be transmitted on the I2C bus until a call to endTransmission() or flush() is performed.
- uint32_t available ()

  Get the number of bytes available to be read from the slave device with read(). This function can be called after a call to requestFrom().

- uint8_t read ()

  Reads a byte of data that was transmitted from a slave I2C device after a call to requestFrom().
- void flush (bool bIssueStop=false)

  Force the data that were queued to be transmitted using the write() functions to be transmitted on the I2C bus.

23.82.1 Detailed Description

Wire Interface Class for I2C.
For I2C communication, you can choose the WireIntf class, or the I2C class. The WireIntf class is simpler to use, while the I2C class provides more low level control.
Note that the system automatically instantiates I2C and WireIntf objects for each of the I2C peripheral modules. The WireIntf objects can be accessed with: Wire, Wire1 and Wire2. The I2C objects can be accesses with I2C[0], I2C[1] and I2C[2].

23.82.2 Member Function Documentation

23.82.2.1 available()

```
uint32_t WireIntf::available ( )
```

Get the number of bytes available to be read from the slave device with read(). This function can be called after a call to requestFrom().

Returns

The number of bytes available for reading from the slave device.

23.82.2.2 begin()

```
void WireIntf::begin ( )
```

Initialize the WireIntf driver object as a master with a bus speed of 100 kHz. This normally only needs to be called once.

23.82.2.3 beginTransmission()

```
void WireIntf::beginTransmission (
            uint8_t addr )
```

Begin a transmission to a I2C slave device at the provided address. Bytes can be queued for transmission with the write() functions, and are transmitted by calling endTransmission().

Parameters

addr	the address of the device to transmit data to.

23.82.2.4 endTransmission()

```
void WireIntf::endTransmission (
            bool stop = true )
```

Transmits the bytes of data queued by using the write() functions, and ends a transmission to a slave device that was started using beginTransmission().

Parameters

stop	If true, sends a stop message after the request, releasing the I2C bus. If false, the driver will send a restart request, keeping the connection alive. This prevents other I2C master devices from transmitting between messages. This allows one master to transmit multiple transmissions uninterrupted.

23.82.2.5 flush()

```
void WireIntf::flush (
            bool bIssueStop = false )
```

Force the data that were queued to be transmitted using the write() functions to be transmitted on the I2C bus.
If you need to know exactly when a transaction will occur, call flush(). If accessing a register, be sure to write all address bytes before the call.

Parameters

bIssueStop	If true, sends a stop message after the request, releasing the I2C bus. If false, the driver will send a restart request, keeping the connection alive. This prevents other I2C master devices from transmitting between messages. This allows one master to transmit multiple transmissions uninterrupted.

23.82.2.6 read()

```
uint8_t WireIntf::read ( )
```

Reads a byte of data that was transmitted from a slave I2C device after a call to requestFrom().

Returns

The byte of data received from the slave I2C device.

23.82.2.7 requestFrom()

```
uint32_t WireIntf::requestFrom (
            uint8_t addr,
            uint32_t len,
            bool stop = true )
```

Request a number of bytes from a slave device. The bytes can then be retrieved from the slave device with the read() and available() functions.

Parameters

addr	the address of the slave to request data from.
len	the number of bytes to request from the slave.
stop	If true, sends a stop message after the request, releasing the I2C bus. If false, the driver will send a restart request, keeping the connection alive. This prevents other I2C master devices from transmitting between messages. This allows one master to transmit multiple transmissions uninterrupted.

Returns

The number of bytes returned from the slave I2C device.

23.82.2.8 write() [1/3]

```
uint32_t WireIntf::write (
            char * str )
```

Queues bytes of data to be transmitted to the slave device. This function can be called after a call to beginTransmission(). The data will not be transmitted on the I2C bus until a call to endTransmission() or flush() is performed.

Parameters

str	A string to send as a series of bytes to the I2C slave device.

Returns

The number of of bytes written to the transmit queue buffer.

23.82.2.9 write() [2/3]

```
uint32_t WireIntf::write (
            uint8_t * buf,
            uint32_t len )
```

Queues bytes of data to be transmitted to the slave device. This function can be called after a call to beginTransmission(). The data will not be transmitted on the I2C bus until a call to endTransmission() or flush() is performed.

Parameters

buf	An array of data to send as bytes to the I2C slave device.
len	The number of bytes to be written from the buf parameter.

Returns

The number of of bytes written to the transmit queue buffer.

23.82.2.10 write() [3/3]

```
uint32_t WireIntf::write (
            uint8_t dat )
```

Queues bytes of data to be transmitted to the slave device. This function can be called after a call to beginTransmission(). The data will not be transmitted on the I2C bus until a call to endTransmission() or flush() is performed.

Parameters

dat	A byte of data to be transmitted to the I2C slave device.

Returns

The number of of bytes written to the transmit queue buffer.

The documentation for this class was generated from the following file:

- i2c.h

23.83 WM8904 Class Reference

`#include <wm8904.h>`

Public Member Functions

- WM8904 (I2C &module)

  The constructor for the WM8904 context.
- void Init (const cfg_t &cfg, const SSC_cfg_t &ssc_cfg)

  Configures and initializes both the driver and the codec.
- void **Shutdown** ()

  Shuts down the WM8904 codec driver.
- void WriteReg (Reg::RegAddr_t reg, uint16_t dat)

  Write a register on the WM8904.
- uint16_t ReadReg (Reg::RegAddr_t reg)

  Read a register on the WM8904.
- void SendCmd (Reg::cmd_t cmd)

  Execute codec Command. A command is: a register to write, a value to write, and a delay of time required for command execution.
- void SendCmdList (Reg::cmd_t *cmds, uint32_t len)

  Execute an array of codec Commands. A command is: a register to write, a value to write, and a delay of time required for command execution.
- void UpdateCmd (Reg::cmd_t cmd, uint16_t updateMask)

  Execute a codec Command to update a register. A command is: a register to write, a value to write, and a delay of time required for command execution.
- void SetVolume (AudioOutSelect_t out, AudioChSelect_t channel, uint8_t volume)

  Set the volume of the selected audio output and channel.
- uint8_t GetVolume (AudioOutSelect_t out, AudioChSelect_t channel)

  Get the volume of the selected audio output and channel.
- void Mute (AudioOutSelect_t out, AudioChSelect_t channel, bool mute)

  Mute or unmute the selected audio output and channel.
- void SetMicGain (AudioChSelect_t channel, uint8_t gain)

  Set the microphone gain of the selected input channel.
- uint8_t GetMicGain (AudioChSelect_t channel)

  Get the microphone gain of the selected input channel.
- void MuteMic (AudioChSelect_t channel, bool mute)

  Mute or unmute the selected input channel.
- int TransmitBuffer (void *buffer, uint32_t bufferLen, bool waitIfNeeded)

  Hands off a buffer to be transmitted to the codec.
- int ReadyReceiveBuffer (void *buffer, uint32_t bufferLen, bool waitIfNeeded)

  Hands off a buffer to be written to by the codec.
- void **RegisterTxBufferDoneCB** (SSC_BufferDoneFn_t cb)

  Registers a callback for when a transmit buffer is finished.
- void **RegisterRxBufferDoneCB** (SSC_BufferDoneFn_t cb)

  Registers a callback for when a receive buffer is finished.

23.83.1 Detailed Description

WM8904 is a driver for the WM8904 Audio Codec.

23.83.2 Constructor & Destructor Documentation

23.83.2.1 WM8904()

```
WM8904::WM8904 (
            I2C & module )
```

The constructor for the WM8904 context.

Parameters

module	A reference to the I2C module to be used for configuring the codec.

23.83.3 Member Function Documentation

23.83.3.1 GetMicGain()

```
uint8_t WM8904::GetMicGain (
            AudioChSelect_t channel )
```

Get the microphone gain of the selected input channel.

Parameters

channel	Input Channel to select.

Returns

Gain the channel is set to.

23.83.3.2 GetVolume()

```
uint8_t WM8904::GetVolume (
            AudioOutSelect_t out,
            AudioChSelect_t channel )
```

Get the volume of the selected audio output and channel.

Parameters

out	Audio Output to select.
channel	Output Channel to select.

Returns

Volume the channel is set to.

23.83.3.3 Init()

```
void WM8904::Init (
            const cfg_t & cfg,
            const SSC_cfg_t & ssc_cfg )
```

Configures and initializes both the driver and the codec.

Parameters

cfg	The driver configuration to use.
ssc_cfg	The SSC driver configuration to use.

23.83.3.4 Mute()

```
void WM8904::Mute (
            AudioOutSelect_t out,
            AudioChSelect_t channel,
            bool mute )
```

Mute or unmute the selected audio output and channel.

Parameters

out	Audio Output to select.
channel	Output Channel to select.
mute	Whether the channel is to be muted.

23.83.3.5 MuteMic()

```
void WM8904::MuteMic (
            AudioChSelect_t channel,
            bool mute )
```

Mute or unmute the selected input channel.

Parameters

channel	Input Channel to select.
mute	Whether the channel is to be muted.

23.83.3.6 ReadReg()

```
uint16_t WM8904::ReadReg (
            Reg::RegAddr_t reg )
```

Read a register on the WM8904.

Parameters

reg	Register to write.

Returns

Data read.

23.83.3.7 ReadyReceiveBuffer()

```
int WM8904::ReadyReceiveBuffer (
            void * buffer,
            uint32_t bufferLen,
            bool waitIfNeeded )
```

Hands off a buffer to be written to by the codec.

Parameters

buffer	A pointer to the buffer to be written to.
bufferLen	The length of the buffer to be written. (Must be multiples of 1, 2, or 4 bytes depending on word bit width)
waitIfNeeded	Whether the driver should wait for space to receive or fail immediately upon exhausting the queue depth.

Returns

Negative on failure.

23.83.3.8 SendCmd()

```
void WM8904::SendCmd (
            Reg::cmd_t cmd )
```

Execute codec Command. A command is: a register to write, a value to write, and a delay of time required for command execution.

Parameters

cmd	Command to execute

23.83.3.9 SendCmdList()

```
void WM8904::SendCmdList (
            Reg::cmd_t * cmds,
            uint32_t len )
```

Execute an array of codec Commands. A command is: a register to write, a value to write, and a delay of time required for command execution.

Parameters

cmds	Commands to execute
len	Number of commands to execute.

23.83.3.10 SetMicGain()

```
void WM8904::SetMicGain (
            WM8904::AudioChSelect_t channel,
            uint8_t gain )
```

Set the microphone gain of the selected input channel.

Parameters

channel	Input Channel to select.
gain	Gain to set channel to.

23.83.3.11 SetVolume()

```
void WM8904::SetVolume (
            WM8904::AudioOutSelect_t out,
            WM8904::AudioChSelect_t channel,
            uint8_t volume )
```

Set the volume of the selected audio output and channel.

Parameters

out	Audio Output to select.
channel	Output Channel to select.
volume	Volume to set channel to.

23.83.3.12 TransmitBuffer()

```
int WM8904::TransmitBuffer (
            void * buffer,
            uint32_t bufferLen,
            bool waitIfNeeded )
```

Hands off a buffer to be transmitted to the codec.

Parameters

buffer	A pointer to the buffer to be transmit.
bufferLen	The length of the buffer to be transmit. (Must be multiples of 1, 2, or 4 bytes depending on word bit width)
waitIfNeeded	Whether the driver should wait for space to transmit or fail immediately upon exhausting the queue depth.

Returns

Negative on failure.

23.83.3.13 UpdateCmd()

```
void WM8904::UpdateCmd (
            Reg::cmd_t cmd,
            uint16_t updateMask )
```

Execute a codec Command to update a register. A command is: a register to write, a value to write, and a delay of time required for command execution.

Parameters

cmd	Command to execute
updateMask	A Positive bit mask of bits to update in the target register

23.83.3.14 WriteReg()

```
void WM8904::WriteReg (
            Reg::RegAddr_t reg,
            uint16_t dat )
```

Write a register on the WM8904.

Parameters

reg	Register to write.
dat	Data to write.

The documentation for this class was generated from the following files:

- wm8904.h
- wm8904.cpp

Chapter 24

File Documentation

24.1 canif.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _CANIF_H
00006 #define _CANIF_H
00007
00008 // This driver has been superseeded by the compatible multican.h driver
00009 #include <multican.h>
00010
00011 #endif
```

24.2 coldfire/cpu/MCF5441X/include/cpu.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __NB_CPU_H
00006 #define __NB_CPU_H
00007
00008 #define NB_BIG_ENDIAN (1)
00009
00010 #endif /* ----- #ifndef __NB_CPU_H  ----- */
```

24.3 cortex-m7/cpu/SAME70/include/cpu.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005
00006 #ifndef  __NB_CPU_H
00007 #define  __NB_CPU_H
00008
00009 #define NB_LITTLE_ENDIAN (1)
00010
00011 #endif   /* ----- #ifndef __NB_CPU_H  ----- */
```

24.4 coldfire/cpu/MCF5441X/include/cpu_pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _CPU_PINS_H_
00006 #define _CPU_PINS_H_
00007
00008 #include <basictypes.h>
00009
00010 #define CPU_PINS
00011 //namespace CPU_PINS
00012 //{
00013 class PinIO
```

```
00014 {
00015    public:
00016     uint8_t port, pinnum;
00017    private:
00018
00019     void funcA(int ft);
00020     void funcB(int ft);
00021     void funcC(int ft);
00022     void funcD(int ft);
00023     void funcE(int ft);
00024     void funcF(int ft);
00025     void funcG(int ft);
00026     void funcH(int ft);
00027     void funcI(int ft);
00028     void funcJ(int ft);
00029     void funcK(int ft);
00030
00031     int getFuncA();
00032     int getFuncB();
00033     int getFuncC();
00034     int getFuncD();
00035     int getFuncE();
00036     int getFuncF();
00037     int getFuncG();
00038     int getFuncH();
00039     int getFuncI();
00040     int getFuncJ();
00041     int getFuncK();
00042
00043
00044    public:
00045     typedef enum {
00046         PIN_FN_0 = 0,
00047         PIN_FN_1 = 1,
00048         PIN_FN_2 = 2,
00049         PIN_FN_3 = 3,
00050         PIN_FN_IN = 4,
00051         PIN_FN_OUT = 6,
00052     } pin_fn_t;
00053
00054
00055     typedef enum {
00056         PULL_DOWN_WEAK = 0,
00057         PULL_UP_WEAK = 1,
00058         PULL_UP_MED = 2,
00059         PULL_UP_STRONG = 3,
00060         PULL_KEEP = 4,
00061         PULL_OFF = 5,
00062     } pull_t;
00063
00064
00065     PinIO(int j, int n)
00066     {
00067         port = j;
00068         pinnum = n;
00069     };
00070     PinIO(const PinIO &rhs) : port(rhs.port), pinnum(rhs.pinnum) {}
00071     PinIO() : port(0), pinnum(0){};
00072
00073     void set(BOOL = TRUE);         // Set output high
00074     BOOL toggle();                 // Toggle the pin state
00075     void clr() { set(FALSE); };    // Set output low
00076     BOOL read();                   // Read pin hi/low state
00077     void hiz() { read(); };        // Set output to tristate
00078     void drive();                  // Turn output on (opposite of tristate)
00079
00080            void setPull(pull_t pull);
00081     inline void PullUp()     { setPull(PULL_UP_MED); }
00082     inline void PullDown()   { setPull(PULL_DOWN_WEAK); }
00083          pull_t getPull();
00084
00085     void    setSlew(uint8_t slew);
00086     inline void setSpeed(uint8_t speed)          { return; }
00087     inline void setStrength(uint8_t strength)    { return; }
00088     void setOpenDrain(bool enable);
00089     inline void multidrv(bool enable) { return setOpenDrain(enable); }
00090     inline void setHyst(bool enable) { return; }
00091
00092     uint8_t getSlew();
00093     inline uint8_t getSpeed() { return 0; }
00094     uint8_t getStrength()     { return 0; }
00095     bool    getOpenDrain()    { return (port & 0x80) != 0; }
00096     bool    getHyst();
00097
00098     inline void setSchmidt(bool enable) { setHyst(enable); }
00099     inline bool getSchmidt()            { return getHyst(); }
00100     inline void HystOn()     { setHyst(true); }
```

```
00101     inline void HystOff()   { setHyst(false); }
00102     inline void SchmidtOn() { setHyst(true); }
00103     inline void SchmidtOff(){ setHyst(false); }
00104
00105     void function(int ft);        // Set pin to special function
00106     void setFn(int ft)            // Set pin to special function
00107         { function(ft); }
00108     int getFunction();            // Get the special function the pin is set to
00109     PinIO &operator=(BOOL b)
00110     {
00111         set(b);
00112         return *this;
00113     };
00114     PinIO &operator=(int i)
00115     {
00116         set(i);
00117         return *this;
00118     };
00119     PinIO &operator=(uint32_t i)
00120     {
00121         set(i);
00122         return *this;
00123     };
00124     operator int() { return read(); };           // Read and return int value
00125     operator BOOL() { return read(); };          // Read and return BOOL value
00126     operator bool() { return (read() != 0); };   // Read and return boolean value
00127     friend class PinIOPortAArray;
00128     friend class PinIOPortBArray;
00129     friend class PinIOPortCArray;
00130     friend class PinIOPortDArray;
00131     friend class PinIOPortEArray;
00132     friend class PinIOPortFArray;
00133     friend class PinIOPortGArray;
00134     friend class PinIOPortHArray;
00135     friend class PinIOPortIArray;
00136     friend class PinIOPortJArray;
00137     friend class PinIOPortKArray;
00138 };
00139
00140 class PinIOPortAArray
00141 {
00142    public:
00143     PinIO operator[](int n) { return PinIO(0, n); };
00144 };
00145 class PinIOPortBArray
00146 {
00147    public:
00148     PinIO operator[](int n) { return PinIO(1, n); };
00149 };
00150 class PinIOPortCArray
00151 {
00152    public:
00153     PinIO operator[](int n) { return PinIO(2, n); };
00154 };
00155 class PinIOPortDArray
00156 {
00157    public:
00158     PinIO operator[](int n) { return PinIO(3, n); };
00159 };
00160 class PinIOPortEArray
00161 {
00162    public:
00163     PinIO operator[](int n) { return PinIO(4, n); };
00164 };
00165 class PinIOPortFArray
00166 {
00167    public:
00168     PinIO operator[](int n) { return PinIO(5, n); };
00169 };
00170 class PinIOPortGArray
00171 {
00172    public:
00173     PinIO operator[](int n) { return PinIO(6, n); };
00174 };
00175 class PinIOPortHArray
00176 {
00177    public:
00178     PinIO operator[](int n) { return PinIO(7, n); };
00179 };
00180 class PinIOPortIArray
00181 {
00182    public:
00183     PinIO operator[](int n) { return PinIO(8, n); };
00184 };
00185 class PinIOPortJArray
00186 {
00187    public:
```

```
00188     PinIO operator[](int n) { return PinIO(9, n); };
00189 };
00190 class PinIOPortKArray
00191 {
00192    public:
00193     PinIO operator[](int n) { return PinIO(10, n); };
00194 };
00195 //}   // namespace CPU_PINS
00196
00197 extern CPU_PINS::PinIOPortAArray PortA;
00198 extern CPU_PINS::PinIOPortBArray PortB;
00199 extern CPU_PINS::PinIOPortCArray PortC;
00200 extern CPU_PINS::PinIOPortDArray PortD;
00201 extern CPU_PINS::PinIOPortEArray PortE;
00202 extern CPU_PINS::PinIOPortFArray PortF;
00203 extern CPU_PINS::PinIOPortGArray PortG;
00204 extern CPU_PINS::PinIOPortHArray PortH;
00205 extern CPU_PINS::PinIOPortIArray PortI;
00206 extern CPU_PINS::PinIOPortJArray PortJ;
00207 extern CPU_PINS::PinIOPortKArray PortK;
00208
00209 #endif
```

24.5 cpu_pins.h File Reference

GPIO pin driver class for the ARM SAME70 (MODM7AE70)

```
#include <sim.h>
#include <basictypes.h>
```

Classes

- class PinIO

  GPIO Pin Class.
- class _PinVector

  GPIO Pin Vector Base Class.
- class PinVector< n >

  GPIO Pin Vector Class PinVector is a template instantiation of the _PinVector class, allowing for minimal storage requirements for potentially large vectors, without heavy code duplication due to template copies.

24.5.1 Detailed Description

GPIO pin driver class for the ARM SAME70 (MODM7AE70)

24.6 cortex-m7/cpu/SAME70/include/cpu_pins.h

Go to the documentation of this file.

```
00001 #ifndef  __CPU_PINS_H
00002 #define  __CPU_PINS_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00017 #include <sim.h>
00018 #include <basictypes.h>
00019
00020
00032 class PinIO {
00033 public:
00034     volatile Pio &pio;
00035     uint32_t mask;
00036
00040     typedef enum {
00041         PIN_FN_IN = 0,
00042         PIN_FN_OUT,
00043         PIN_FN_A,
00044         PIN_FN_B,
00045         PIN_FN_C,
00046         PIN_FN_D
00047     } pin_fn_t;
00048
```

```
00053     PinIO() : pio(*((volatile Pio*)PIOA)), mask(0) {}
00065     constexpr PinIO(uint32_t port, uint32_t pin) : pio(*((volatile Pio*)PIOA+port)), mask(1 « pin) {}
00070     constexpr PinIO(const PinIO &rhs): pio(rhs.pio), mask(rhs.mask) {}
00071
00076     void setFn(pin_fn_t fn) const {
00077         if (fn > PIN_FN_OUT) {
00078             uint8_t bits = ((uint8_t)fn)-2;
00079             if (bits & 0x1) { pio.PIO_ABCDSR[0] |= mask; }
00080             else            { pio.PIO_ABCDSR[0] &= ~mask; }
00081             if (bits & 0x2) { pio.PIO_ABCDSR[1] |= mask; }
00082             else            { pio.PIO_ABCDSR[1] &= ~mask; }
00083
00084             pio.PIO_PDR = mask;
00085         }
00086         else {
00087             if (fn) { pio.PIO_OER = mask; }
00088             else    { pio.PIO_ODR = mask; }
00089             pio.PIO_PER = mask;
00090         }
00091     }
00092
00097     int8_t getFn()
00098     {
00099         // TODO: check if PIO_LOCKSR is set. if it's set, a peripheral has locked the I/O line
00100         // The only way to unlock it is to apply a hardware reset to the PIO controller
00101         if(mask == 0) { return -1; }  // pin is not valid
00102
00103         if(pio.PIO_PSR & mask) // if PIO_PSR bit is set, PIO is active on corresponding I/O line
    (peripheral is inactive)
00104         {
00105             // Determine if I/O line is configured for an input or output
00106             // Read output status register (PIO_OSR). If 0, I/O is an input. If 1, I/O is an output
00107             return ((pio.PIO_OSR & mask) ? PIN_FN_OUT : PIN_FN_IN);
00108         }
00109         else                    // if PIO_PSR bit is clear, PIO is inactive on corresponding I/O line
    (peripheral is active)
00110         {
00111             // Determine which peripheral function is set
00112             uint32_t abcdsr0 = (pio.PIO_ABCDSR[0] & mask);
00113             uint32_t abcdsr1 = (pio.PIO_ABCDSR[1] & mask);
00114
00115             if (abcdsr0 & mask)
00116             {
00117                 if (abcdsr1 & mask) { return PIN_FN_D; }
00118                 else                { return PIN_FN_B; }
00119             }
00120             else
00121             {
00122                 if (abcdsr1 & mask) { return PIN_FN_C; }
00123                 else                { return PIN_FN_A; }
00124             }
00125         }
00126     }
00127
00132     void function(pin_fn_t fn) const { setFn(fn); }
00136     void hiz() const { setFn(PIN_FN_IN); }
00140     void drive() const { setFn(PIN_FN_OUT); }
00144     inline void set() const { pio.PIO_SODR = mask; }
00148     inline void clr() const { pio.PIO_CODR = mask; }
00153     inline bool tgl() const {bool val = pio.PIO_ODSR & mask;(&(pio.PIO_SODR))[val] = mask;return val;}
00158     inline bool toggle() const { return tgl(); }
00164     inline bool readBack() const { return (pio.PIO_PDSR & mask); }
00169     inline bool read() const { hiz(); return readBack(); }
00175     inline bool operator=( bool val ) { (&(pio.PIO_SODR))[!val] = mask; return val;}
00182    inline PinIO& operator=(const PinIO& rhs)
00183     {
00184         bool val = rhs;
00185         (&(pio.PIO_SODR))[!val] = mask; return *this;
00186     }
00187
00192     inline operator bool() const { return (pio.PIO_PDSR & mask); }
00193
00198     inline bool operator!() const { return ((pio.PIO_ODR & mask) == 0); }
00199
00206     void multidrv(bool enable) const { (&(pio.PIO_MDER))[!enable] = mask; }
00207
00214     void setHighStrength( bool bHighDrive )
00215     {
00216         if (bHighDrive) { pio.PIO_DRIVER |= mask; }
00217         else            { pio.PIO_DRIVER &= ~mask;}
00218     }
00219
00226     void PullUp(bool enable) const {if (enable) {pio.PIO_PUER =mask;} else {pio.PIO_PUDR =mask;}};
00233     void PullDown(bool enable) const {if (enable) {pio.PIO_PPDER =mask;} else {pio.PIO_PPDDR =mask;}};
00234
00240     uint16_t analogRead() const;
00241
```

```
00242     //friend class PinIOArray2;
00243
00244 };
00245
00246
00259 class _PinVector {
00260 protected:
00261     struct pinCfg {
00262         uint8_t port : 3;
00263         uint8_t num : 5;
00264     };
00265     const uint8_t len;
00266     union {
00267         pinCfg _pins[4];
00268         pinCfg *const pinArr;
00269     };
00270     // _PinVector constructors are protected to prevent the construction of the
00271     //  base class, as the base class has no allocated storage for the pin
00272     //  confuguration settings.
00273     _PinVector(uint8_t _len, pinCfg *arr);
00274     _PinVector(uint8_t len, pinCfg *arr,
00275             PinIO *initpins, uint32_t pinCount);
00276 public:
00282     uint32_t operator=(uint32_t val);
00288     PinIO operator[](int idx);
00289
00296     void config(uint32_t idx, PinIO cfg);
00304     void config(PinIO *pinCfgs, uint32_t count);
00305
00310     operator uint32_t() const;
00311 };
00312
00313
00323 template<uint8_t n> class PinVector : public _PinVector {
00324     _PinVector::pinCfg pinStore[n];
00325 public:
00330     inline PinVector(): _PinVector(n, pinStore) {}
00331
00338     inline PinVector(PinIO *initpins, uint32_t pinCount)
00339         : _PinVector(n, pinStore, initpins, pinCount)
00340     { }
00346     inline uint32_t operator=(uint32_t val) {return (*(_PinVector*)this) = val; }
00347     inline uint32_t operator=(int val) {return (*(_PinVector*)this) = (uint32_t)val; }
00348 };
00349
00350 // TinyPinVectors are a template specialization made for vector lengths where
00351 //  the entire configuration list can fit within the memory space of the
00352 //  storage pointer, thereby reducing the storage requirements to an absolute
00353 //  minimum, while preventing the duplication of code.
00354 #define TinyPinVector(n) \
00355 template<> class PinVector<n> : public _PinVector { \
00356 public: \
00357     inline PinVector(): _PinVector((n), nullptr) {} \
00358  \
00359     inline PinVector(PinIO *initpins, uint32_t pinCount) \
00360         : _PinVector((n), nullptr, initpins, pinCount) \
00361     { } \
00362     inline uint32_t operator=(uint32_t val) {return (*(_PinVector*)this) = val; } \
00363     inline uint32_t operator=(int val) {return (*(_PinVector*)this) = (uint32_t)val; } \
00364 };
00365
00366 TinyPinVector(1);
00367 TinyPinVector(2);
00368 TinyPinVector(3);
00369 TinyPinVector(4);
00370
00371
00372
00373  // end of groupGPIO
00375 #endif   /* ----- #ifndef __CPU_PINS_H  ----- */
```

24.7 dspi.h File Reference

NetBurner MCF5441x DSPI API.

```
#include <nbrtos.h>
#include <basictypes.h>
#include <sim5441x.h>
```

Classes

- class DSPIModule

  DSPIModule is a SPI communications driver. It is an object based driver, which allows for low overhead multiplexing between peripherals with different bus configurations.

Enumerations

- enum csReturnType {
  DEASSERT_NEVER = 0 , DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 ,
  DEASSERT_NEVER = 0 ,
  DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 }

  Chip select return types.
- enum spiChipSelect {
  CHIP_SELECT_0 = 0xFE , CHIP_SELECT_1 = 0xFD , CHIP_SELECT_2 = 0xFB , CHIP_SELECT_3 = 0xF7
  ,
  CHIP_SELECT_DISABLED = 0xFF , CHIP_SELECT_0 = 0 , CHIP_SELECT_1 = 1 , CHIP_SELECT_2 = 2 ,
  CHIP_SELECT_3 = 3 , CHIP_SELECT_DISABLED = 0xFF }

  Chip select number.
- enum spiChipSelectPolarity { CS_ASSERT_LOW = 0x00 , CS_ASSERT_HIGH = 0xFF , CS_ASSERT_LOW = 0 , CS_ASSERT_HIGH = 1 }

  Chip select polarity.

Functions

- uint8_t DSPIInit (uint8_t SPIModule=DEFAULT_DSPI_MODULE, uint32_t Baudrate=2000000, uint8_t QueueBitSize=8, uint8_t CS=0x00, uint8_t CSPol=0x0F, uint8_t ClkPolarity=0, uint8_t ClkPhase=1, BOOL DoutHiz=TRUE, uint8_t QCD=0, uint8_t DTL=0)

  Initialize a DSPI module.
- uint8_t QSPIInit (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x0F, uint8_t chipSelectPolarity=1, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize Queued Serial Peripheral Interface (QSPI)
- uint8_t QSPIStart (puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL)

  Start QSPI Data Transfer.
- BOOL QSPIdone ()

  Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.

24.7.1 Detailed Description

NetBurner MCF5441x DSPI API.

24.8 coldfire/cpu/MCF5441X/include/dspi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00054 #ifndef _DMA_SPI_H_INC
00055 #define _DMA_SPI_H_INC
00056 #include <nbrtos.h>
00057 #include <basictypes.h>
00058 #include <sim5441x.h>
00059
00060 #ifndef __cplusplus
00061 #error DSPI driver is a C++ only library
00062 #endif
```

```
00063
00064 /*
00065    DSPI state
00066 */
00071 #define DSPI_OK (0)
00072 #define DSPI_BUSY (1)
00073 #define DSPI_ERROR (2)
00076 // MCR Register masks
00077 #define MCR_MASTER_INIT 0x80000C01
00078 #define MCR_HALT_BIT 0x00000001
00079 #define MCR_DIS_TXF 0x00002000
00080 #define MCR_DIS_RXF 0x00001000
00081 #define MCR_CLR_FIFOS 0x00000C00
00082
00083 // CTAR Register masks
00084 #define CTAR_CLOCK_POLARITY 0x04000000
00085 #define CTAR_CLOCK_PHASE 0x02000000
00086 #define CTAR_FRAME_16BIT 0x78000000
00087 #define CTAR_FRAME_4BIT 0x18000000
00088
00089 // SR Register masks
00090 #define SR_EOQF_MASK 0x10000000
00091 #define SR_CLR_FLAGS 0x8A0A0000
00092
00093 // RSER (Interrupt/DMA enable) Register masks
00094 #define RSER_EOQF_IRQ_ONLY ~0x8B0B0000
00095 #define RSER_DMA_IRQ_ONLY 0x03030000
00096
00097 // PUSHR (command) Register masks
00098 #define PUSHR_CONT_BIT 0x80000000
00099 #define PUSHR_EOQ_BIT 0x08000000
00100
00102 //
00103 // DMA CHANNEL NUMBERS
00104 #define DMA_CH_DSPI_0_RX 0x0C   // (12)
00105 #define DMA_CH_DSPI_0_TX 0x0D   // (13)
00106 #define DMA_CH_DSPI_1_RX 0x0E   // (14)
00107 #define DMA_CH_DSPI_1_TX 0x0F   // (15)
00108 #define DMA_CH_DSPI_2_RX 0x1C   // (28)
00109 #define DMA_CH_DSPI_2_TX 0x1D   // (29)
00110 #define DMA_CH_DSPI_3_RX 0x2C   // (44)
00111 #define DMA_CH_DSPI_3_TX 0x2D   // (45)
00112
00113 // EDMA CR Register masks
00114 #define CR_CLEAR_NON_GRP_PRIO ~0x000300FF
00115 #define CR_SET_RR_CH_ARB 0x00000004
00116
00118 // EDMA TCD masks
00119 #define TCD_ATTR_8BIT_TRANS 0x0000
00120 #define TCD_ATTR_16BIT_TRANS 0x0101
00121
00122 #define TCD_XOFF_0uint8_t 0x0000
00123 #define TCD_XOFF_1uint8_t 0x0001
00124 #define TCD_XOFF_2uint8_t 0x0002
00125
00126 #define TCD_XITER_CNT_MASK 0x7FFF
00127
00128 #define TCD_CSR_DONE_BIT 0x0080
00129 #define TCD_CSR_DISABLE_REQ 0xC008
00130 #define TCD_CSR_DREQ_INT_MAJOR 0xC00A
00131 // end EDMA TCD masks
00133
00134 #define DEFAULT_DSPI_MODULE 1
00135 #define DSPI_MODULE_COUNT 4
00136
00144 enum csReturnType
00145 {
00146     DEASSERT_NEVER = 0,
00147     DEASSERT_AFTER_LAST = 1,
00149     DEASSERT_EVERY_TRANSFER = 2,
00150 };
00151
00159 enum spiChipSelect
00160 {
00161     CHIP_SELECT_0 = 0xFE,
00162     CHIP_SELECT_1 = 0xFD,
00163     CHIP_SELECT_2 = 0xFB,
00164     CHIP_SELECT_3 = 0xF7,
00165     CHIP_SELECT_DISABLED = 0xFF,
00166 };
00167
00177 enum spiChipSelectPolarity
00178 {
00179     CS_ASSERT_LOW = 0x00,
00180     CS_ASSERT_HIGH = 0xFF,
00181 };
00182
```

```
00183 /*
00184  *********************************************************************************-
00185  *
00186  *  dspiDMAStruct
00187  *
00188  *  bool enabled - whether or not the driver is using DMA for the transfer
00189  *
00190  *  uint8_t byteCount - the bytes transfer per dma call
00191  *
00192  *  uint32_t savedMCR - the actual MCR value, saved when doing DMA based transfer
00193  *                      as the CS inactive state is inverted on the active CS
00194  *
00195  *********************************************************************************-
00196  */
00197 typedef struct
00198 {
00199     bool enabled;
00200     uint8_t byteCount;
00201     bool rxPresent;
00202     bool txPresent;
00203     dspistruct savedDSPI;
00204     bool csState;
00205 } dspiDMAStruct;
00206
00207 /*
00208  *********************************************************************************-
00209  *
00210  *   dspiDriverStruct
00211  *
00212  *   This struct contains the major variables/configurations used for a DSPI transfer
00213  *
00214  *   volatile uint8_t* pDSPIRxbuf/pDSPITxbuf -
00215  *   These pointers are used to track the locations in memory where data will be
00216  *   read or written to the peripheral
00217  *
00218  *   uint8_t BitsPerQueue - This is the number if bits per transfer, (value = 8 - 32)
00219  *
00220  *   uint32_t DSPI_SizeLeft - This is the number if bytes left in the transfer
00221  *
00222  *   uint16_t Command_Mask - This is a partial configuration for the queue's command reg
00223  *
00224  *   OS_SEM* DSPI_Sem - This is a pointer to an external semaphore provided by DSPIStart()
00225  *
00226  *   uint8_t DSPI_INT_STATUS - Status of the spi device
00227  *
00228  *   uint32_t WordsToWrite - Number of words to write to the command/TX Queues and Read from RX
00229  *
00230  *   uint32_t LastWordsToWrite - the amount from the previous ISR
00231  *
00232  *********************************************************************************-
00233  */
00234 typedef struct
00235 {
00236     volatile uint8_t *pDSPIRxbuf;
00237     volatile uint8_t *pDSPITxbuf;
00238     uint8_t BitsPerQueue;
00239     uint32_t DSPI_SizeLeft;
00240     uint16_t Command_Mask;
00241     csReturnType csReturnToInactive;
00242     OS_SEM *DSPI_Sem;
00243     volatile uint8_t DSPI_INT_STATUS;
00244     uint32_t WordsToWrite;
00245     uint32_t LastWordsToWrite;
00246     volatile BOOL DSPIfinished;
00247     dspiDMAStruct dma;
00248 } dspiDriverStruct;
00249
00250 // uint8_t DSPIInit( uint8_t SPIModule, uint32_t Baudrate, uint8_t QueueBitSize, uint8_t CS,
00251 //                 uint8_t CSPol, uint8_t ClkPolarity, uint8_t ClkPhase, BOOL DoutHiz, uint8_t QCD,
      uint8_t DTL );
00252
00259 class DSPIModule
00260 {
00261     uint32_t m_moduleNum;
00262     uint32_t m_mcr;
00263     uint32_t m_ctar0;
00264     uint32_t m_ctar1;
00265     bool m_enableDMA;
00266     uint16_t m_CommandMask;
00267     uint8_t m_BitsPerQueue;
00268     OS_SEM *m_finishedSem;
00269     uint32_t m_actualBaudrate;
00270
00271    public:
00272     bool m_inProgress;
00273
00274     static DSPIModule *lastCxts[DSPI_MODULE_COUNT];
```

```
00275     static dspiDriverStruct driverCxt[DSPI_MODULE_COUNT];
00276
00289     DSPIModule(uint8_t SPIModule);
00290
00321     DSPIModule(uint8_t SPIModule,
00322                uint32_t baudRateInBps,
00323                uint8_t transferSizeInBits = 8,
00324                uint8_t peripheralChipSelects = 0x00,
00325                uint8_t chipSelectPolarity = 0x0F,
00326                uint8_t clockPolarity = 0,
00327                uint8_t clockPhase = 1,
00328                BOOL doutHiz = TRUE,
00329                uint8_t csToClockDelay = 0,
00330                uint8_t delayAfterTransfer = 0);
00331
00360     uint8_t Init(uint32_t baudRateInBps = 2000000,
00361                  uint8_t transferSizeInBits = 8,
00362                  uint8_t peripheralChipSelects = 0x00,
00363                  uint8_t chipSelectPolarity = 0x0F,
00364                  uint8_t clockPolarity = 0,
00365                  uint8_t clockPhase = 1,
00366                  BOOL doutHiz = TRUE,
00367                  uint8_t csToClockDelay = 0,
00368                  uint8_t delayAfterTransfer = 0);
00369
00386     uint8_t Start(uint8_t *transmitBufferPtr,
00387                   volatile uint8_t *receiveBufferPtr,
00388                   uint32_t byteCount,
00389                   int csReturnToInactive = DEASSERT_AFTER_LAST);
00390
00406     inline uint8_t Tx(uint8_t *transmitBufferPtr, uint32_t byteCount, int csReturnToInactive =
      DEASSERT_AFTER_LAST)
00407     {
00408         return Start(transmitBufferPtr, NULL, byteCount, csReturnToInactive);
00409     }
00410
00426     inline uint8_t Rx(uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive =
      DEASSERT_AFTER_LAST)
00427     {
00428         return Start(NULL, receiveBufferPtr, byteCount, csReturnToInactive);
00429     }
00430
00437     bool EnableDMA(bool enableDMA = true);
00442     inline bool DisableDMA() { return EnableDMA(false); }
00443
00450     bool RegisterSem(OS_SEM *finishedSem);
00455     inline bool ClrSem() { return RegisterSem(NULL); }
00460     inline OS_SEM *GetSem() { return m_finishedSem; }
00461
00467     inline bool Done() { return !m_inProgress; }
00468
00476     static BOOL Done(uint8_t SPIModule);
00477
00483     inline uint32_t GetActualBaudrate() { return m_actualBaudrate; }
00484
00493     inline bool SetCS(uint8_t CS)
00494     {
00495         OSLockObj lock;
00496         if (m_inProgress) { return false; }
00497         m_CommandMask = (((m_mcr >> 16) ^ CS) & 0xFF);
00498         return true;
00499     }
00500
00536     friend uint8_t DSPIInit(uint8_t SPIModule,
00537                             uint32_t Baudrate,
00538                             uint8_t QueueBitSize,
00539                             uint8_t CS,
00540                             uint8_t CSPol,
00541                             uint8_t ClkPolarity,
00542                             uint8_t ClkPhase,
00543                             BOOL DoutHiz,
00544                             uint8_t QCD,
00545                             uint8_t DTL);
00546 };
00547
00548 /*-------------------------------------------------------------------------------
00549  *
00550  *  Initialize DMA Serial Peripheral Interface (DSPI)
00551  *
00552  *      Parameters:
00553  *          baudRateInBps          - Maximum speed requested. (1) (2)
00554  *          transferSizeInBits     - 8 thru 32 valid (3)
00555  *          SPIModule              - the dspi module to target on the processor
00556  *          peripheralChipSelects  - DSPI_CS[7:0] (4)
00557  *          chipSelectPolarity     - CS[7:0] 0 inactive logic level low, 1 high
00558  *          clockPolarity          - 0 inactive logic level low, 1 high
00559  *          clockPhase             - 0 (5) or 1 (6)
```

```
00560  *          doutHiz               - Data output high impedance between transfers
00561  *          csToClockDelay        - Delay form chip select to valid clock (7)
00562  *          delayAfterTransfer    - Delay between data transfers (8)
00563  *
00564  *      Return:
00565  *          Current DSPI state    - DSPI_OK success or DSPI_BUSY busy
00566  *
00567  *      Notes:
00568  *          (1) Maximum BAUD rate is CPU_CLOCK / 3
00569  *          (2) Will select highest BAUD rate available that does not exceed.
00570  *          (3) Transfers > 8 must be word aligned
00571  *          (4) Select based on chipSelectPolarity
00572  *          (5) 0 data captured leading edge of DSPI_CLK, changed following edge.
00573  *          (6) 1 data changed leading edge of DSPI_CLK, captured following edge.
00574  *          (7) 0 default is as close to 1/2 DSPI_CLK without going under,
00575  *              keeping with the interface to QSPI
00576  *          (8) 0 default is 17/(system clock / 2), in keepin with interface to QSPI
00577  *
00578  ------------------------------------------------------------------------------------*/
00579
00616 uint8_t DSPIInit(uint8_t SPIModule = DEFAULT_DSPI_MODULE,
00617                  uint32_t Baudrate = 2000000,
00618                  uint8_t QueueBitSize = 8,
00619                  uint8_t CS = 0x00,
00620                  uint8_t CSPol = 0x0F,
00621                  uint8_t ClkPolarity = 0,
00622                  uint8_t ClkPhase = 1,
00623                  BOOL DoutHiz = TRUE,
00624                  uint8_t QCD = 0,
00625                  uint8_t DTL = 0);
00626
00627 /*
00628  ************************************************************************-
00629  *
00630  *  Start DSPI Data Transfer
00631  *
00632  *      Parameters:
00633  *          transmitBufferPtr   - Buffer with data to send, NULL for receiving
00634  *          receiveBufferPtr    - Buffer to receive data, NULL for sending
00635  *          byteCount           - Count of bytes to send or receive
00636  *          SPIModule           - Selects the dspi module to use
00637  *          finishedSem         - Optional semaphore to set when completed
00638  *
00639  *      Return:
00640  *          Current DSPI state  - DSPI_OK success or DSPI_BUSY busy
00641  *
00642  *      Notes:
00643  *          None
00644  *
00645  ************************************************************************-
00646  */
00647
00648 uint8_t DSPIStart(uint8_t SPIModule,
00649                   puint8_t transmitBufferPtr,
00650                   volatile uint8_t *receiveBufferPtr,
00651                   uint32_t byteCount,
00652                   OS_SEM *finishedSem = NULL,
00653                   uint8_t enableDMA = TRUE,
00654                   int csReturnToInactive = DEASSERT_AFTER_LAST);
00655
00656 /*-------------------------------------------------------------------------
00657  *
00658  * Check current DSPI Data Transfer
00659  *
00660  *    Parameters:
00661  *       None
00662  *
00663  *    Return:
00664  *       Progress of transfer    - TRUE is finished, FALSE is active
00665  *
00666  *    Notes:
00667  *       None
00668  *
00669  -------------------------------------------------------------------------*/
00670 BOOL DSPIdone(uint8_t SPIModule = DEFAULT_DSPI_MODULE);
00671
00672 // QSPI to DSPITranslation macros
00673 //  Translates QSPI calls to DSPI calls
00674
00675 inline uint8_t QSPIInit(uint32_t baudRateInBps = 2000000,
00676                         uint8_t transferSizeInBits = 8,
00677                         uint8_t peripheralChipSelects = 0x0F,
00678                         uint8_t chipSelectPolarity = 1,
00679                         uint8_t clockPolarity = 0,
00680                         uint8_t clockPhase = 1,
00681                         BOOL doutHiz = TRUE,
00682                         uint8_t csToClockDelay = 0,
```

```
00683                         uint8_t delayAfterTransfer = 0)
00684 {
00685     return DSPIInit(DEFAULT_DSPI_MODULE, baudRateInBps, transferSizeInBits, peripheralChipSelects,
      chipSelectPolarity, clockPolarity,
00686                     clockPhase, doutHiz, csToClockDelay, delayAfterTransfer);
00687 }
00688
00689 inline uint8_t QSPIStart(puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t
      byteCount, OS_SEM *finishedSem = NULL)
00690 {
00691     return DSPIStart(DEFAULT_DSPI_MODULE, transmitBufferPtr, receiveBufferPtr, byteCount,
      finishedSem);
00692 }
00693
00694 inline BOOL QSPIdone()
00695 {
00696     return DSPIdone();
00697 }
00698
00699 #endif /* ----- #ifndef _DMA_SPI_H_INC  ----- */
00700
```

24.9 dspi.h File Reference

NetBurner DMA SPI (DSPI) API for ARM SAME70 (MODM7AE70, SBE70LC)

```
#include <nbrtos.h>
#include <basictypes.h>
#include <xdmac.h>
```

Classes

- class SPIModule

  SPI Peripheral Module Class.

Macros

- #define **DSPI_OK** (0)

  DSPI OK.
- #define **DSPI_BUSY** (1)

  DSPI Busy.
- #define **DSPI_ERROR** (2)

  DSPI Error.
- #define **DEFAULT_DSPI_MODULE** 0

  Default DSPI module.
- #define **DSPI_MODULE_COUNT** 1

  Number of modules: 0, 1.

Enumerations

- enum csReturnType {
  DEASSERT_NEVER = 0 , DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 ,
  DEASSERT_NEVER = 0 ,
  DEASSERT_AFTER_LAST = 1 , DEASSERT_EVERY_TRANSFER = 2 }

  Chip select deassertion modes. Used to determine when the driver should deassert chip selects during SPI transfer.
- enum spiChipSelect {
  CHIP_SELECT_0 = 0xFE , CHIP_SELECT_1 = 0xFD , CHIP_SELECT_2 = 0xFB , CHIP_SELECT_3 = 0xF7
  ,
  CHIP_SELECT_DISABLED = 0xFF , CHIP_SELECT_0 = 0 , CHIP_SELECT_1 = 1 , CHIP_SELECT_2 = 2 ,
  CHIP_SELECT_3 = 3 , CHIP_SELECT_DISABLED = 0xFF }

  Chip select number.

- enum spiChipSelectPolarity { CS_ASSERT_LOW = 0x00 , CS_ASSERT_HIGH = 0xFF , CS_ASSERT_LOW = 0 , CS_ASSERT_HIGH = 1 }

  Chip select polarity.

Functions

- uint8_t DSPIInit (uint8_t SPIModule=DEFAULT_DSPI_MODULE, uint32_t Baudrate=2000000, uint8_t QueueBitSize=8, uint8_t CS=0x00, uint8_t CSPol=0x0F, uint8_t ClkPolarity=0, uint8_t ClkPhase=1, BOOL DoutHiz=TRUE, uint8_t QCD=0, uint8_t DTL=0)

  Initialize a DSPI module.
- uint8_t DSPIStart (uint8_t SPIModule, puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL, uint8_t enableDMA=TRUE, int csReturnToInactive=DEASSERT_AFTER_LAST)

  Start a DSPI transfer.
- BOOL DSPIdone (uint8_t SPIModule=DEFAULT_DSPI_MODULE)

  Check SPI status.
- uint8_t QSPIInit (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChipSelects=0x0F, uint8_t chipSelectPolarity=1, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL doutHiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Compatibility function for previous drivers. Initialize SPI module.
- uint8_t QSPIStart (puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL)

  Compatibility function for previous drivers. Start a SPI transfer.
- BOOL QSPIdone ()

  Compatibility function for previous drivers. Check SPI status.

24.9.1 Detailed Description

NetBurner DMA SPI (DSPI) API for ARM SAME70 (MODM7AE70, SBE70LC)

24.10 cortex-m7/cpu/SAME70/include/dspi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00024 #ifndef  _DMA_SPI_H_INC
00025 #define  _DMA_SPI_H_INC
00026
00027 #include <nbrtos.h>
00028 #include <basictypes.h>
00029 #include <xdmac.h>
00030
00031 #ifndef __cplusplus
00032 #error DSPI driver is a C++ only library
00033 #endif
00034
00035 //#define __DEBUG_DSPI 1
00036
00043 #define DSPI_OK      ( 0 )
00044 #define DSPI_BUSY    ( 1 )
00045 #define DSPI_ERROR   ( 2 )
00053 #define DEFAULT_DSPI_MODULE 0
00054 #define DSPI_MODULE_COUNT   1
00064 enum csReturnType {
00065     DEASSERT_NEVER          = 0,
00066     DEASSERT_AFTER_LAST     = 1,
00067     DEASSERT_EVERY_TRANSFER = 2
00068 };
00069
00077 enum spiChipSelect {
00078     CHIP_SELECT_0        = 0,
00079     CHIP_SELECT_1        = 1,
00080     CHIP_SELECT_2        = 2,
00081     CHIP_SELECT_3        = 3,
00082     CHIP_SELECT_DISABLED = 0xFF
```

```
00083 };
00084
00094 enum spiChipSelectPolarity {
00095     CS_ASSERT_LOW          = 0,
00096     CS_ASSERT_HIGH         = 1,
00097 };
00098
00099 // *   spiDriverStruct
00100 //*
00101 // *   This structure contains the major variables/configurations used for a DSPI transfer
00102 // *
00103 // *   volatile uint8_t* pDSPIRxbuf/pDSPITxbuf -
00104 // *   These pointers are used to track the locations in memory where data will be
00105 // *   read or written to the peripheral
00106 // *
00107 // *   uint8_t  BitsPerQueue     This is the number if bits per transfer, (value = 8 - 32)
00108 // *   uint32_t DSPI_SizeLeft    This is the number if bytes left in the transfer
00109 // *   uint16_t Command_Mask     This is a partial configuration for the queue's command reg
00110 // *   OS_SEM*  DSPI_Sem         This is a pointer to an external semaphore provided by DSPIStart()
00111 // *   uint8_t  DSPI_INT_STATUS  Status of the SPI device
00112 // *   uint32_t WordsToWrite     Number of words to write to the command/TX Queues and Read from RX
00113 // *   uint32_t LastWordsToWrite the amount from the previous ISR
00114
00115 typedef struct
00116 {
00117     csReturnType csReturnToInactive;
00118     volatile uint8_t SPI_INT_STATUS;
00119     volatile BOOL    SPIfinished;
00120     XdmaCh_t         *txCh;
00121     XdmaCh_t         *rxCh;
00122     dma_descview_1   last_TxDesc;     // descriptor setting the 'lastxfer' transfer
00123     dma_descview_1   last16_TxDesc;   // descriptor setting the 'lastxfer' transfer - 16bit
00124     uint32_t         last_cr;         // buffer to allow DMA to set the 'lastxfer' at completion
00125     uint32_t         lastTxISR;
00126     uint32_t         lastRxISR;
00127     volatile uint8_t *rxBuf;
00128     uint32_t         rxLen;
00129 } spiDriverStruct;
00130
00131 //uint8_t DSPIInit( uint8_t SPIModule, uint32_t Baudrate, uint8_t QueueBitSize, uint8_t CS,
00132 //                  uint8_t CSPol, uint8_t ClkPolarity, uint8_t ClkPhase, BOOL DoutHiz, uint8_t QCD,
    uint8_t DTL );
00133
00140 class SPIModule
00141 {
00142 public:
00143     // static  SPIModule *lastCtxs[DSPI_MODULE_COUNT];
00144     // static  spiDriverStruct driverCtx[DSPI_MODULE_COUNT];
00145
00146 protected:
00147     static  SPIModule *lastCtxs[DSPI_MODULE_COUNT];
00148     static  spiDriverStruct driverCtx[DSPI_MODULE_COUNT];
00149
00150     uint32_t    m_moduleNum;
00151     uint32_t    m_regMR;
00152     uint32_t    m_regCSR;
00153     OS_SEM      *m_finishedSem;
00154     uint32_t    m_busSpeed;
00155     uint8_t     m_CSNum;
00156     volatile bool m_inProgress;
00157
00158     virtual void ReadyHW();
00159     virtual spiDriverStruct *getCtx() { return driverCtx + m_moduleNum; }
00160     inline Spi   * spi() { return SPI0 + m_moduleNum; }
00161
00162 public:
00163
00164 #ifdef __DEBUG_DSPI
00165     virtual void    dumpRegs();
00166 #endif
00167
00176     SPIModule( uint8_t SPIModule );
00177
00195     SPIModule( uint8_t SPIModule, uint32_t baudRateInBps,
00196             uint8_t transferSizeInBits = 8, uint8_t peripheralChipSelects = 0x00,
00197             uint8_t chipSelectPolarity = 0x0F, uint8_t clockPolarity = 0,
00198             uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00199             uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 );
00200
00216     virtual uint8_t Init( uint32_t baudRateInBps = 2000000,
00217             uint8_t transferSizeInBits = 8, uint8_t peripheralChipSelects = 0x00,
00218             uint8_t chipSelectPolarity = 0x0F, uint8_t clockPolarity = 0,
00219             uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00220             uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 );
00221
00231     virtual uint32_t SetBusSpeed(uint32_t maxSpeed);
00232
```

```
00252     virtual uint8_t Start( uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr,
00253                 uint32_t byteCount, int csReturnToInactive = DEASSERT_AFTER_LAST );
00254
00264     virtual inline uint8_t Tx( uint8_t *transmitBufferPtr, uint32_t byteCount,
00265                     int csReturnToInactive = DEASSERT_AFTER_LAST )
00266     { return Start(transmitBufferPtr, NULL, byteCount, csReturnToInactive); }
00267
00277     virtual inline uint8_t Rx( uint8_t *receiveBufferPtr, uint32_t byteCount,
00278                     int csReturnToInactive = DEASSERT_AFTER_LAST )
00279     { return Start(NULL, receiveBufferPtr, byteCount, csReturnToInactive); }
00280
00281     // The SAME70 always uses DMA. These functions kept here for reference to other platforms
00282     //    bool EnableDMA(bool enableDMA = true);
00283     //    inline bool DisableDMA() { return EnableDMA(false); }
00284
00294     virtual bool RegisterSem( OS_SEM *finishedSem );
00295
00301     virtual inline bool ClrSem() { return RegisterSem(NULL); }
00302
00308     virtual inline OS_SEM * GetSem() { return m_finishedSem; }
00309
00317     virtual inline bool Done() { return !m_inProgress; }
00318
00328     static BOOL Done( uint8_t SPIModule );
00329
00338     virtual inline uint32_t GetActualBaudrate() { return m_busSpeed; }
00339
00347     virtual inline bool SetCS( uint8_t CS )
00348     {
00349         OSLockObj lock;
00350         if (m_inProgress) { return false; }
00351         m_regMR = (m_regMR & ~SPI_MR_PCS_Msk);
00352         switch(CS)
00353         {
00354             case 0:
00355                 m_regMR |= SPI_MR_PCS(0xE);
00356                 break;
00357             case 1:
00358                 m_regMR |= SPI_MR_PCS(0xD);
00359                 break;
00360             case 2:
00361                 m_regMR |= SPI_MR_PCS(0xB);
00362                 break;
00363             case 3:
00364                 m_regMR |= SPI_MR_PCS(0x7);
00365                 break;
00366             default:
00367                 m_regMR |= SPI_MR_PCS(0xF);
00368         }
00369
00370
00371         return true;
00372     }
00373
00374     // The ISR used by the DSPI driver. Internal use only (dspi.cpp).
00375     friend void SPI_DMA_Isr(XdmaCh_t *dma, int module);
00376 };
00377
00378 typedef SPIModule DSPIModule;
00379 typedef SPIModule SPI_SPI;
00380
00410 uint8_t DSPIInit( uint8_t SPIModule = DEFAULT_DSPI_MODULE, uint32_t Baudrate = 2000000,
00411                   uint8_t QueueBitSize = 8,                uint8_t CS = 0x00,
00412                   uint8_t CSPol = 0x0F,                    uint8_t ClkPolarity = 0,
00413                   uint8_t ClkPhase = 1,                    BOOL DoutHiz = TRUE,
00414                   uint8_t QCD = 0,                         uint8_t DTL = 0 );
00415 // Note: csToClockDelay:  0 default is 17/(system clock / 2), in keeping with interface to QSPI
00416
00417
00440 uint8_t DSPIStart( uint8_t SPIModule, puint8_t transmitBufferPtr, volatile uint8_t* receiveBufferPtr,
00441       uint32_t byteCount, OS_SEM* finishedSem = NULL, uint8_t enableDMA = TRUE,
00442       int csReturnToInactive = DEASSERT_AFTER_LAST );
00443
00449 BOOL DSPIdone( uint8_t SPIModule = DEFAULT_DSPI_MODULE );
00450
00451 // QSPI to DSPI Translation macros. Note that the 'Q' stands for Queued SPI
00452
00481 inline uint8_t QSPIInit( uint32_t baudRateInBps = 2000000, uint8_t transferSizeInBits = 8,
00482       uint8_t peripheralChipSelects = 0x0F, uint8_t chipSelectPolarity = 1,
00483       uint8_t clockPolarity = 0, uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00484       uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 )
00485 {
00486     return DSPIInit( DEFAULT_DSPI_MODULE, baudRateInBps, transferSizeInBits,
00487                 peripheralChipSelects, chipSelectPolarity, clockPolarity, clockPhase,
00488                 doutHiz, csToClockDelay, delayAfterTransfer );
00489 }
00490
```

```
00512 inline uint8_t QSPIStart( puint8_t transmitBufferPtr, volatile uint8_t* receiveBufferPtr,
00513                           uint32_t byteCount, OS_SEM* finishedSem = NULL )
00514 {
00515     return DSPIStart( DEFAULT_DSPI_MODULE, transmitBufferPtr, receiveBufferPtr, byteCount, finishedSem
      );
00516 }
00517
00525 inline BOOL QSPIdone()
00526 {
00527     return DSPIdone();
00528 }
00529
00530 #endif   /* ----- #ifndef _DMA_SPI_H_INC  ----- */
00531
```

24.11 DualEthernet.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DUAL_ETHER_H
00006 #define _DUAL_ETHER_H
00007
00008 /* Add the 2nd Ethernet interface return the interface block */
00009 void Add2ndEthernet();
00010
00011 #endif
```

24.12 etherswitch.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __ETHERSWITCH_H
00006 #define __ETHERSWITCH_H
00007
00008 #define ESW_PER_FULL_EN 0x00070007
00009 void EtherSwitchInit();
00010
00011 #endif /* ----- #ifndef __ETHERSWITCH_H  ----- */
```

24.13 coldfire/cpu/MCF5441X/include/ethervars.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _ETHER_VARS_H
00006 #define _EHTER_VARS_H
00007
00008 // PHY ID Register #3 bits with the last 4 revision number bits being don't cares
00009 #define MICREL_ID (0x1610)
00010 #define MICREL8041_ID (0x1510)
00011 #define DAVICOM_ID (0xB880)
00012 #define AMD_ID (0x5610)
00013 #define NATIONAL_ID (0x5C90)
00014 #define NATIONAL_IEEE1588_ID (0x5CE0)
00015 #define MICREL8081_ID (0x1560)
00016
00017 #if (defined MCF5272)
00018 #define CORE72
00019 #else
00020 #define CORE70
00021 #endif
00022
00023 // PHY Register addresses
00024 #define PHY_REG_BASIC_CTL 0x00
00025 #define PHY_REG_BASIC_ST 0x01
00026 #define PHY_REG_ID_1 0x02
00027 #define PHY_REG_ID_2 0x03
00028 #define PHY_REG_A_NEG_ADVERT 0x04
00029 #define PHY_REG_A_NEG_LPA 0x05
00030 #define PHY_REG_A_NEG_EXP 0x06
00031 #define PHY_REG_A_NEG_NEXT_PG 0x07
00032 #define PHY_REG_LP_NEXT_PG_ABIL 0x08
00033 #define PHY_AFE_CTL_1 0x10
00034 #define PHY_REG_MILL_CTL 0x14
```

```
00035 #define PHY_RXER_CNTR 0x15
00036 #define PHY_MODE_STRAP_OVER 0x16
00037 #define PHY_MODE_STRAP_STATUS 0x17
00038 #define PHY_EXPANDED_CTL 0x18
00039 #define PHY_REG_IRQ_CTL_ST 0x1B
00040 #define PHY_REG_PHY_CTL_1 0x1E
00041 #define PHY_REG_PHY_CTL_2 0x1F
00042
00043 // PHY Register masks
00044 #define PHY_LINK_STAT 0x0004
00045 #define PHY_A_NEG_COMP 0x0020
00046 #define PHY_IRQ_DOWN_EN 0x0400
00047 #define PHY_IRQ_UP_EN 0x0100
00048
00049 #define IP_20uint8_t_ID (0x4500)
00050 #define ETHER_TYPE_IP (0x0800)
00051 #define ETHER_TYPE_ARP (0x0806)
00052
00053 #define NUM_TX_BD (3)
00054
00055 #ifdef _DEBUG
00056 /******************WARNING **********************/
00057 // This number must be a power of 2 and match the mask value
00058 // OK to mess with the non debug value, leave the debug value alone.
00059 #define NUM_RX_BD (8)
00060 #define NUM_DEBUG_RX_BUFF (8)
00061 #define DB_RX_MASK (0x07)
00062 /******************WARNING **********************/
00063
00064 #else
00065
00066 #define NUM_RX_BD (10)
00067
00068 #endif
00069
00070 #define FEC_RMII_10T_MODE (0x00000200)
00071
00072 #include <netinterface.h>
00073 #include <sim5441x.h>
00074
00075 typedef enum
00076 {
00077     PORT1_WAITING = 0,
00078     PORT1_INITED = 1,
00079     PORT1_SWITCH = 2
00080 } Port1_State;
00081
00082 class EthernetVars : public EtherLikeInterface
00083 {
00084    public:
00085     int fecn;   // FEC(Freescale Ethernet Controller) number
00086     bool inited;
00087     uint8_t supMacInUse;   // Bitmask indicating which Supplemental MACs are in use
00088
00089     OS_SEM SemSend;
00090
00091     int nTx;   // TX and RX counters
00092     int nRx;
00093     uint32_t TxIsrCnt;
00094     uint32_t RxIsrCnt;
00095     uint32_t LrnIsrCnt;
00096
00097     // Place for all the buffers
00098     volatile EtherBD TxBD[NUM_TX_BD] __attribute__((aligned(16)));
00099     volatile EtherBD RxBD[NUM_RX_BD] __attribute__((aligned(16)));
00100
00101     volatile PoolPtr RxBDpp[NUM_RX_BD] __attribute__((aligned(4)));
00102     volatile PoolPtr TxBDpp[NUM_TX_BD] __attribute__((aligned(4)));
00103
00104     volatile uint8_t RxBDFlag[NUM_RX_BD] __attribute__((aligned(4)));
00105     volatile uint8_t TxBDFlag[NUM_RX_BD] __attribute__((aligned(4)));
00106
00107     int last_rxbd;
00108
00109     uint32_t LastIsrStatus;
00110
00111     volatile uint32_t RxPosts;
00112     volatile uint32_t TxcPosts;
00113     volatile uint32_t RxReject;
00114
00115     // Current link status
00116     BOOL bEthLink;   /* do we have link? */
00117     BOOL bEthDuplex; /* Are we set-up for duplex? */
00118     BOOL bEth100Mb;  /* Are we 100Mb? */
00119     BOOL bPromisc;   /* Are we in Promiscuous Mode? */
00120
00121     // Discovered PHY data
```

```
00122     BOOL RMII;
00123     uint8_t PHY_addr;
00124     uint16_t PHY_ID;
00125
00126     // Phy forcing modes
00127     BOOL SetEthDuplex;
00128     BOOL SetEth100Mb;
00129     BOOL AutoNeg;
00130
00131     // Link status info
00132     BOOL bNewLink;
00133     BOOL bNewDuplex;
00134     BOOL bEverHadLink;        /*Have we ever had link */
00135     bool bDescriptorsReadied; /* Have we readied the buffer descriptors? */
00136
00137     volatile uint32_t phy_data_pd;
00138     volatile uint32_t phy_data;
00139
00140     uint32_t last_phy_read;
00141
00142     uint32_t AutoTimeOut;   // Link but no Auto Negotiation counter
00143
00144     // Real public interface
00145     EthernetVars(int fec, const char *name);
00146
00147     void ManualEthernetConfig(BOOL Speed100, BOOL FullDuplex, BOOL AutoNegotiate);
00148
00149     void SetPromiscuous(BOOL promisc);
00150     inline void EnablePromiscuous() { SetPromiscuous(TRUE); }
00151     inline void DisablePromiscuous() { SetPromiscuous(FALSE); }
00152     void EnablePHY();
00153     void DisablePHY();
00154
00155     virtual void send_func(PoolPtr poolPtr);
00156     virtual void kill_if();
00157     virtual void EnableMulticast(MACADR macAddress, BOOL addAddress);
00158     virtual bool LinkActive();
00159     virtual bool LinkDuplex();
00160     virtual int LinkSpeed();
00161     virtual const char *GetInterfaceName();
00162
00163     void InitializeEthernet(bool bRegisterInterface);
00164
00165     uint32_t GetMII(uint8_t addr);
00166     uint32_t GetMII(uint8_t mii_dev, uint8_t addr);
00167     uint32_t SetMII(uint8_t addr, uint16_t value);
00168     uint32_t SetMII(uint8_t mii_dev, uint8_t addr, uint16_t value);
00169
00170     // Internal processing functions
00171     void DoRX(PoolPtr pp, uint16_t length);
00172     int ProcessRxBuffer(PoolPtr pp);
00173     void ResetEnet();
00174     void ResetEnetPart2();
00175     void ReadyDescriptors();
00176     void WarmEnetReset();
00177     void ProcessDuplexStatus(uint16_t phyStat);
00178     void RxIsrProc(volatile uint32_t &isr, uint32_t frameMask, uint32_t bufferMask);
00179     void TxIsrProc(volatile uint32_t &isr, uint32_t frameMask, uint32_t bufferMask);
00180
00181     // Info display functions
00182     void ShowLinkStatus();
00183     void ShowMII(uint8_t mii_dev, uint8_t addr, PCSTR name, uint16_t mask);
00184     void ShowFECRegisters();
00185     void ShowPhyterMII();
00186 #ifdef _DEBUG
00187     void DebugEthernetTx(PoolPtr pb);   // Transmits a buffer and frees it afterword.
00188     void NMIEthernetYield(volatile uint32_t &isr, uint32_t frameMask, uint32_t bufferMask);
00189     void NMI_Eth_Transmit(PoolPtr pb);
00190 #endif
00191 };
00192
00193 void SetPromiscuous(BOOL promisc);
00194 inline void EnablePromiscuous()
00195 {
00196     SetPromiscuous(TRUE);
00197 }
00198 inline void DisablePromiscuous()
00199 {
00200     SetPromiscuous(FALSE);
00201 }
00202 void ShowESWRegisters();
00203
00204 extern "C"
00205 {
00206     void DoDumpEnet();
00207 }
00208
```

```
00209 extern EthernetVars enet0;
00210 extern EthernetVars enet1;
00211
00212 // Set to true save config and reboot to enter switch mode.
00213 extern config_bool DoEtherSwitch;
00214
00215 #ifdef _DEBUG
00216 #define DEBUG_INTERFACE_MAGIC_IFNUM (0x80)
00217 #endif
00218
00219 #endif
```

24.14 cortex-m7/cpu/SAME70/include/ethervars.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _ETHER_VARS_H
00006 #define _ETHER_VARS_H
00007 #include <same70q21.h>
00008
00009 //PHY ID Register #3 bits with the last 4 revision number bits being don't cares
00010 #define MICREL_ID               (0x1610)
00011 #define MICREL8041_ID           (0x1510)
00012 #define DAVICOM_ID              (0xB880)
00013 #define AMD_ID                  (0x5610)
00014 #define NATIONAL_ID             (0x5C90)
00015 #define NATIONAL_IEEE1588_ID    (0x5CE0)
00016 #define MICREL8081_ID           (0x1560)
00017
00018 #if ( defined MCF5272 )
00019 #define CORE72
00020 #else
00021 #define CORE70
00022 #endif
00023
00024 // PHY Register addresses
00025 #define PHY_REG_BASIC_CTL       0x00
00026 #define PHY_REG_BASIC_ST        0x01
00027 #define PHY_REG_ID_1            0x02
00028 #define PHY_REG_ID_2            0x03
00029 #define PHY_REG_A_NEG_ADVERT    0x04
00030 #define PHY_REG_A_NEG_LPA       0x05
00031 #define PHY_REG_A_NEG_EXP       0x06
00032 #define PHY_REG_A_NEG_NEXT_PG   0x07
00033 #define PHY_REG_LP_NEXT_PG_ABIL 0x08
00034 #define PHY_AFE_CTL_1           0x10
00035 #define PHY_REG_MILL_CTL        0x14
00036 #define PHY_RXER_CNTR           0x15
00037 #define PHY_MODE_STRAP_OVER     0x16
00038 #define PHY_MODE_STRAP_STATUS   0x17
00039 #define PHY_EXPANDED_CTL        0x18
00040 #define PHY_REG_IRQ_CTL_ST      0x1B
00041 #define PHY_REG_PHY_CTL_1       0x1E
00042 #define PHY_REG_PHY_CTL_2       0x1F
00043
00044 // PHY Register masks
00045 #define PHY_LINK_STAT           0x0004
00046 #define PHY_A_NEG_COMP          0x0020
00047 #define PHY_IRQ_DOWN_EN         0x0400
00048 #define PHY_IRQ_UP_EN           0x0100
00049
00050 #define IP_20BYTE_ID           (0x4500)
00051 #define ETHER_TYPE_IP          (0x0800)
00052 #define ETHER_TYPE_ARP         (0x0806)
00053
00054 // *Do not* set below 3. HW loads two frames (aka, descriptors) in parallel.
00055 // See SAME70 Datasheet 38.8.8
00056 #define NUM_TX_BD   (4)
00057
00058
00059 #ifdef _DEBUG
00060 /******************WARNING **********************/
00061 // This number must be a power of 2 and match the mask value
00062 //OK to mess with the non debug value, leave the debug value alone.
00063 #define NUM_RX_BD   (8)
00064 #define NUM_DEBUG_RX_BUFF   (8)
00065 #define DB_RX_MASK (0x07)
00066 /******************WARNING **********************/
00067
00068 #else
00069
00070 #define NUM_RX_BD   (10)
00071
```

```
00072 #endif
00073
00074 #define ETHER_MAC_COUNT 1
00075
00076
00077 #define RXBD_Error_Mask (GMAC_BD_RX_BAD_FCS)
00078
00079
00080
00081
00082 #define FEC_RMII_10T_MODE      (0x00000200)
00083
00084 #include <netinterface.h>
00085
00086 typedef enum {
00087     PORT1_WAITING   = 0,
00088     PORT1_INITED    = 1,
00089     PORT1_SWITCH    = 2
00090 } Port1_State;
00091
00092 struct EthernetVars : public EtherLikeInterface
00093 {
00094     bool inited;
00095     uint8_t supMacInUse; // Bitmask indicating which Supplemental MACs are in use
00096
00097     OS_SEM SemSend;
00098
00099     int nTx; //TX and RX counters
00100     int nRx;
00101     int nextTxCheck;
00102     uint32_t TxCnt;
00103     uint32_t IsrCount;
00104     uint32_t TxIsrCnt;
00105     uint32_t RxIsrCnt;
00106
00107     //Place for all the buffers
00108     //Buffer descriptors need to be Device/Strongly Ordered, but driver context
00109     // needs to allow non-aligned access and therefore must be Normal memory.
00110     // Refer to ARMv7-M Architecture Reference Manual for further details.
00111 //    volatile EtherBD TxBD[NUM_TX_BD] __attribute__( ( aligned( 8 ) ) );
00112 //    volatile EtherBD RxBD[NUM_RX_BD] __attribute__( ( aligned( 8 ) ) );
00113     volatile EtherBD *TxBD;
00114     volatile EtherBD *RxBD;
00115
00116     volatile PoolPtr RxBDpp[NUM_RX_BD] __attribute__( ( aligned( 4 ) ) );
00117     volatile PoolPtr TxBDpp[NUM_TX_BD] __attribute__( ( aligned( 4 ) ) );
00118
00119     volatile uint8_t RxBDFlag[NUM_RX_BD] __attribute__( ( aligned( 4 ) ) );
00120     volatile uint8_t TxBDFlag[NUM_TX_BD] __attribute__( ( aligned( 4 ) ) );
00121
00122     int last_rxbd;
00123
00124     uint32_t LastIsrStatus;
00125
00126     volatile uint32_t RxPosts;
00127     volatile uint32_t TxcPosts;
00128     volatile uint32_t RxReject;
00129
00130     //Discovered PHY data
00131     uint8_t PHY_addr;
00132     uint16_t PHY_ID;
00133
00134     //Current link status
00135     bool bEthLink : 1; /* do we have link? */
00136     bool bEthDuplex : 1; /* Are we set-up for duplex? */
00137     bool bEth100Mb : 1; /* Are we 100Mb? */
00138     bool bPromisc : 1;  /* Are we in Promiscuous Mode? */
00139
00140
00141     //Phy forcing modes
00142     bool SetEthDuplex : 1;
00143     bool SetEth100Mb : 1;
00144     bool AutoNeg : 1;
00145
00146     //Link status info
00147     bool bNewLink : 1;
00148     bool bNewDuplex : 1;
00149     bool bEverHadLink : 1; /*Have we ever had link */
00150     bool bDescriptorsReadied : 1; /* Have we readied the buffer descriptors? */
00151
00152
00153     volatile uint32_t phy_data_pd ;
00154     volatile uint32_t phy_data;
00155
00156     uint32_t last_phy_read;
00157
00158     uint32_t AutoTimeOut;  // Link but no Auto Negotiation counter
```

```
00159
00160     //Real public interface
00161     BOOL InitializeEthernet();
00162     EthernetVars(const char * name, EtherBD *txbd, EtherBD *rxbd );
00163
00164     void ManualEthernetConfig( BOOL Speed100, BOOL FullDuplex, BOOL AutoNegotiate );
00165     void SetPromiscuous(BOOL promisc);
00166     inline void EnablePromiscuous() { SetPromiscuous( TRUE ); }
00167     inline void DisablePromiscuous() { SetPromiscuous( FALSE ); }
00168     void EnableMulticast( MACADR macAddress );
00169     void DisableMulticast( MACADR macAddress );
00170     void EnablePHY();
00171     void DisablePHY();
00172
00173     virtual void send_func( PoolPtr poolPtr );
00174     virtual void kill_if();
00175     virtual void EnableMulticast( MACADR macAddress, BOOL addAddress );
00176     virtual bool LinkActive();
00177     virtual bool LinkDuplex();
00178     virtual int  LinkSpeed();
00179     virtual const char* GetInterfaceName();
00180
00181
00182     uint32_t GetMII( uint8_t addr );
00183     uint32_t GetMII( uint8_t mii_dev, uint8_t addr );
00184     uint32_t SetMII( uint8_t addr, uint16_t value );
00185     uint32_t SetMII( uint8_t mii_dev, uint8_t addr, uint16_t value );
00186
00187     //Internal processing functions
00188     void DoRX( PoolPtr pp, uint16_t length );
00189     int ProcessRxBuffer(PoolPtr pp);
00190     void ResetEnet();
00191     void ResetEnetPart2();
00192     void ReadyDescriptors();
00193     void WarmEnetReset();
00194     void ProcessDuplexStatus( uint16_t phyStat );
00195     void RxIsrProc(volatile uint32_t isr);
00196     void TxIsrProc(volatile uint32_t isr);
00197
00198     //Info display functions
00199     void ShowLinkStatus();
00200     void ShowMII( uint8_t mii_dev, uint8_t addr, PCSTR name, uint16_t mask );
00201     void ShowFECRegisters();
00202     void ShowPhyterMII();
00203 #ifdef _DEBUG
00204     void DebugEthernetTx( PoolPtr pb ); //Transmits a buffer and frees it afterword.
00205     void NMIEthernetYield();
00206     void NMI_Eth_Transmit(PoolPtr pb);
00207 #endif
00208 };
00209
00210 void SetPromiscuous(BOOL promisc);
00211 inline void EnablePromiscuous() { SetPromiscuous( TRUE ); }
00212 inline void DisablePromiscuous() { SetPromiscuous( FALSE ); }
00213 void ShowESWRegisters();
00214 extern "C"
00215 {
00216 void DoDumpEnet();
00217 }
00218
00219
00220 extern EthernetVars enet0;
00221
00222
00223 #ifdef _DEBUG
00224 #define DEBUG_INTERFACE_MAGIC_IFNUM (0x80)
00225 #endif
00226
00227
00228 #endif
```

24.15 HiResTimer.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _HIRESTIMER_H
00007 #define _HIRESTIMER_H
00008
00009 #include <limits.h>
00010 #include <nbrtos.h>
00011 #include <sim.h>
00012 #include <stdio.h>
```

```
00013
00014 #define LOGME                                                           \
00015     iprintf("We got to line %d in file %s\r\n", __LINE__, __FILE__); \
00016     OSTimeDly(2);
00017
00018 #define DEFAULT_TIMER -1
00019 #define TIMER_COUNT 4
00020 #define TIMER_RELEASED 2
00021
00022 // Clock speed macros
00023 extern unsigned long CPU_CLOCK;
00024 #define NB_CPU_CLK (CPU_CLOCK / 2)
00025
00026 #define TIME_PER_CLK (1.0 / NB_CPU_CLK)
00027
00028 class HiResTimer
00029 {
00030     static HiResTimer timers[TIMER_COUNT];
00031     static uint8_t timerCreated[TIMER_COUNT];
00032
00033     int timer_;                         // Timer number
00034     volatile timerstruct *simTimer_;    // Timer struct from sim.h
00035     volatile int prescaler_;            // The value of the hardware prescaler
00036     volatile uint32_t resetCount_;      // Holds the count of the timer resets
00037     OS_SEM delaySem_;                   // OS semaphore struct
00038     uint32_t firstInit_;                // Have we set up this timer before ?
00039     volatile uint32_t delayCalled_;     // Is the timer being used for a delay?
00040
00041     // The method that will be called when the timer triggers an interupt
00042     void (*interruptFunction_)();
00043
00044     // Disable construction and copy construction
00045     HiResTimer(){};
00046     HiResTimer(const HiResTimer &from){};
00047     HiResTimer &operator=(const HiResTimer &from);
00048     // Only actual constructor
00049     HiResTimer(int timer);
00050     ~HiResTimer(){};
00051
00052     // Fire the object's interrupt method
00053     void fireInterrupt();
00054
00055   public:
00056     // class function to fire the interrupt function for the given timer
00057     // called by the Interrupt macro routine
00058     static void fireInterrupt(int timer);
00059
00060     void init(double InterruptTime = 0);   // Initializes the timer for use
00061     void init_ticks(uint32_t ReferenceTicks, int prescale = 0);
00062
00063     /*
00064      * Starts the selected timer. The user MUST call HiResTimerInit for the given timer
00065      * before calling this method.
00066      */
00067     inline void start()
00068     {
00069         // Setting the CLK bits to 01 on the DTMR register, starting the timer
00070 #ifdef MOD5213
00071         simTimer_->dtmr |= 0x2;
00072 #else
00073         simTimer_->tmr |= 0x2;
00074 #endif
00075     };
00076
00077     inline uint32_t readHigh() { return resetCount_; };   // Reads the number of timer
    overflows/resets/interrupts
00078     inline uint32_t readLow()
00079     {
00080         // Reads the timer register value
00081 #ifdef MOD5213
00082         return simTimer_->dtcn;
00083 #else
00084         return simTimer_->tcn;
00085 #endif
00086     };
00087     inline int getPrescaler() { return prescaler_; };   // Returns the value of the prescaler
00088     double readTime();                                  // Ouputs the number of seconds since the
    timer was started
00089
00090     inline void stop()
00091     {
00092         // Stops the timer
00093         simTimer_->tmr &= ~(0x6);
00094     };
00095
00096     inline void stopClear()
00097     {
```

```
00098         simTimer_->tmr &= ~(0x6);
00099         simTimer_->tcn = 0x0;
00100         simTimer_->ter = 0x3;
00101         resetCount_ = 0;
00102     }
00103
00104     void delay(
00105         double DelayTime);   // Starts a precise delay that will switch tasks while waiting (Do not
      use for shorter than ~150 useconds)
00106     void delay_uSec(uint32_t DelayTime);   // Starts a precise delay that will switch tasks while
      waiting
00107     void pollingDelay(double DelayTime);   // Starts a precise delay that will busy wait
00108     inline void pollingDelay_uSec(uint32_t delayTime)
00109     {
00110         if (delayTime < (INT_MAX / NB_CPU_CLK)) { pollingDelay_refTicks(delayTime * NB_CPU_CLK /
      1000000UL); }
00111         else
00112         {
00113             pollingDelay_refTicks(delayTime » 2 * (NB_CPU_CLK / 250000UL));
00114         }
00115     }
00116     void pollingDelay_refTicks(uint32_t ticks);
00117
00118     void clockGenerator(double Frequency);   // Set up a clock at the chosen frequency
00119
00120     inline void setInterruptFunction(void (*interruptFunction)())
00121     {
00122         interruptFunction_ = interruptFunction;
00123     };                                                                  // Set the method to be
      called on interrupt triggers
00124     inline void clearInterruptFunction() { interruptFunction_ = NULL; };   // Clear the interrupt
      method
00125
00126     char *toString();   // Converts the timer object to a printable c string
00127
00128     // Returns a pointer for the given timer number, returns NULL if the timer does not exist
00129     static HiResTimer *getHiResTimer(int timer = DEFAULT_TIMER);
00130     void releaseTimer();
00131
00132     int getTimerNumber() { return timer_; }
00133 };
00134
00135 #endif /* _HIRESTIMER_H */
```

24.16 i2c_class.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _I2C_CLASS_H
00006 #define _I2C_CLASS_H
00007 #include <basictypes.h>
00008 #include <nbrtos.h>
00009 #include <sim.h>
00010
00011 /* defined I2C Timeout values will be used if user does not include the 'ticks_to_wait'
00012    parameter when calling I2C functions                                                 */
00013 #define I2C_RX_TX_TIMEOUT (5)    // Ticks allowed before timeout of a single byte transmission
00014 #define I2C_START_TIMEOUT (20)   // Ticks allowed before timeout when attempting start on I2C bus
00015
00016 #define NUM_I2C_MODULES (6)
00017
00018 enum e_i2cstatus
00019 {
00020     OKAY = 0,
00021     NEXT_WRITE_OK,
00022     NEXT_READ_OK,
00023     MASTER_OK,
00024     TIMEOUT,
00025     BUS_NOT_AVAIL,
00026     NOT_READY,
00027     LOST_ARB,
00028     LOST_ARB_ADD,
00029     NO_LINK_RX_ACK
00030 };
00031
00032 // //Struct that contains I2C Slave I/O Buffers
00033 // struct I2C_Slave_Record
00034 // {
00035 // OS_SEM I2C_Slave_RX_Semaphore;  //semaphore used to determine when a slave RX interrupt occurs
00036
00037 // volatile uint8_t * pI2CRxbuf;
00038 // volatile uint8_t * pI2CTxbuf;
00039
```

```
00040 // volatile uint32_t I2Crx_put;
00041 // volatile uint32_t I2Crx_get;
00042
00043 // volatile uint32_t I2Ctx_put;
00044 // volatile uint32_t I2Ctx_get;
00045 // };
00046
00047 class i2c_master
00048 {
00049    private:
00050     volatile i2cstruct *i2cRegs;
00051
00052    protected:
00053     inline bool busBusy() { return (0x20 & i2cRegs->i2sr) == 0x20; };
00054     inline bool slaveMode() { return (0x20 & i2cRegs->i2cr) == 0x00; };
00055     inline bool arbLost() { return (0x10 & i2cRegs->i2sr) == 0x10; };
00056     inline bool addressedAsSlave() { return (0x40 & i2cRegs->i2sr) == 0x40; };
00057     inline bool addressedSlaveTX() { return (0x04 & i2cRegs->i2sr) == 0x04; };
00058     inline bool transmitting() { return (0x10 & i2cRegs->i2cr) == 0x10; };
00059     inline bool receivedRXAck() { return (0x01 & i2cRegs->i2sr) == 0x00; };
00060     inline bool sentRXAck() { return (0x08 & i2cRegs->i2cr) == 0x00; };
00061     inline void setAck() { i2cRegs->i2cr &= 0xF7; };
00062     inline void setRX() { i2cRegs->i2cr &= 0xEF; };
00063     inline void setRepeatStart() { i2cRegs->i2cr |= 0x04; };
00064     inline void clrArbLost() { i2cRegs->i2sr &= 0xEF; };
00065     inline void clrIntFlag() { i2cRegs->i2sr &= 0xFD; };
00066     inline void sendStop() { i2cRegs->i2cr &= 0xDF; }
00067
00068     inline void sendData(uint8_t data) { i2cRegs->i2dr = data; };
00069     inline void setFreq(uint8_t freq) { i2cRegs->i2fdr = freq; }
00070
00071    public:
00072     inline void setTX() { i2cRegs->i2cr |= 0x10; };
00073     inline void setNoAck() { i2cRegs->i2cr |= 0x08; };
00074     inline uint8_t readData() { return i2cRegs->i2dr; };
00075     void isr();
00076     OS_SEM I2C_Semaphore;   // semaphore for when interrupt occurs
00077     volatile e_i2cstatus status;
00078     i2c_master();
00079     void init(uint8_t moduleNum, uint8_t freqdiv);
00080     void stop(uint32_t ticks_to_wait);
00081     void disable();
00082     void reset();
00083     void read8(uint8_t *val, uint32_t ticks_to_wait);
00084     void write8(uint8_t, uint32_t);
00085     void start(uint8_t, bool, uint32_t);
00086     void restart(uint8_t, bool, uint32_t);
00087     volatile bool bLastTX;      // boolean to tell when we have completed a master-TX
00088     volatile uint8_t RnW;       // 2 = Last TX was not address, 1 = RX request, 0 = TX request
00089     volatile bool bRestarted;   // Used in buf RX, TX functions to identify if we restarted
00090 };
00091
00092 extern i2c_master I2C[NUM_I2C_MODULES];
00093 // i2c_master I2C0(0, 0x3C);
00094
00095 #endif
```

24.17 intcdefs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _INTC_H_
00006 #define _INTC_H_
00007
00008 void SetIntc(int intcnum, long func, int vector, int level);
00009
00010 #define SETUP_IRQ1_EDGEPORT1_ISR(f, l) SetIntc(0, (long)f, 1, l)
00011 #define SETUP_IRQ2_EDGEPORT2_ISR(f, l) SetIntc(0, (long)f, 2, l)
00012 #define SETUP_IRQ3_EDGEPORT3_ISR(f, l) SetIntc(0, (long)f, 3, l)
00013 #define SETUP_IRQ4_EDGEPORT4_ISR(f, l) SetIntc(0, (long)f, 4, l)
00014 #define SETUP_IRQ5_EDGEPORT5_ISR(f, l) SetIntc(0, (long)f, 5, l)
00015 #define SETUP_IRQ6_EDGEPORT6_ISR(f, l) SetIntc(0, (long)f, 6, l)
00016 #define SETUP_IRQ7_EDGEPORT7_ISR(f, l) SetIntc(0, (long)f, 7, l)
00017
00018 #define SETUP_DMA0_ISR(f, l) SetIntc(0, (long)f, 8, l)
00019 #define SETUP_DMA1_ISR(f, l) SetIntc(0, (long)f, 9, l)
00020 #define SETUP_DMA2_ISR(f, l) SetIntc(0, (long)f, 10, l)
00021 #define SETUP_DMA3_ISR(f, l) SetIntc(0, (long)f, 11, l)
00022 #define SETUP_DMA4_ISR(f, l) SetIntc(0, (long)f, 12, l)
00023 #define SETUP_DMA5_ISR(f, l) SetIntc(0, (long)f, 13, l)
00024 #define SETUP_DMA6_ISR(f, l) SetIntc(0, (long)f, 14, l)
00025 #define SETUP_DMA7_ISR(f, l) SetIntc(0, (long)f, 15, l)
00026 #define SETUP_DMA8_ISR(f, l) SetIntc(0, (long)f, 16, l)
```

```
00027 #define SETUP_DMA9_ISR(f, l) SetIntc(0, (long)f, 17, l)
00028 #define SETUP_DMA10_ISR(f, l) SetIntc(0, (long)f, 18, l)
00029 #define SETUP_DMA11_ISR(f, l) SetIntc(0, (long)f, 19, l)
00030 #define SETUP_DMA12_ISR(f, l) SetIntc(0, (long)f, 20, l)
00031 #define SETUP_DMA13_ISR(f, l) SetIntc(0, (long)f, 21, l)
00032 #define SETUP_DMA14_ISR(f, l) SetIntc(0, (long)f, 22, l)
00033 #define SETUP_DMA15_ISR(f, l) SetIntc(0, (long)f, 23, l)
00034 #define SETUP_DMA_ERR_ISR(f, l) SetIntc(0, (long)f, 24, l)
00035
00036 #define SETUP_CORE_WD_ISR(f, l) SetIntc(0, (long)f, 25, l)
00037
00038 #define SETUP_UART0_ISR(f, l) SetIntc(0, (long)f, 26, l)
00039 #define SETUP_UART1_ISR(f, l) SetIntc(0, (long)f, 27, l)
00040 #define SETUP_UART2_ISR(f, l) SetIntc(0, (long)f, 28, l)
00041 #define SETUP_UART3_ISR(f, l) SetIntc(0, (long)f, 29, l)
00042
00043 #define SETUP_I2C0_ISR(f, l) SetIntc(0, (long)f, 30, l)
00044
00045 #define SETUP_DSPI0_ISR(f, l) SetIntc(0, (long)f, 31, l)
00046
00047 #define SETUP_DMATIMER0_ISR(f, l) SetIntc(0, (long)f, 32, l)
00048 #define SETUP_DMATIMER1_ISR(f, l) SetIntc(0, (long)f, 33, l)
00049 #define SETUP_DMATIMER2_ISR(f, l) SetIntc(0, (long)f, 34, l)
00050 #define SETUP_DMATIMER3_ISR(f, l) SetIntc(0, (long)f, 35, l)
00051
00052 #define SETUP_ENET0_TXF_ISR(f, l) SetIntc(0, (long)f, 36, l)
00053 #define SETUP_ENET0_TXB_ISR(f, l) SetIntc(0, (long)f, 37, l)
00054 #define SETUP_ENET0_UN_ISR(f, l) SetIntc(0, (long)f, 38, l)
00055 #define SETUP_ENET0_RL_ISR(f, l) SetIntc(0, (long)f, 39, l)
00056 #define SETUP_ENET0_RXF_ISR(f, l) SetIntc(0, (long)f, 40, l)
00057 #define SETUP_ENET0_RXB_ISR(f, l) SetIntc(0, (long)f, 41, l)
00058 #define SETUP_ENET0_MII_ISR(f, l) SetIntc(0, (long)f, 42, l)
00059 #define SETUP_ENET0_LC_ISR(f, l) SetIntc(0, (long)f, 43, l)
00060 // 44 is unused
00061 #define SETUP_ENET0_GRA_ISR(f, l) SetIntc(0, (long)f, 45, l)
00062 #define SETUP_ENET0_EBERR_ISR(f, l) SetIntc(0, (long)f, 46, l)
00063 #define SETUP_ENET0_BABT_ISR(f, l) SetIntc(0, (long)f, 47, l)
00064 #define SETUP_ENET0_BABR_ISR(f, l) SetIntc(0, (long)f, 48, l)
00065
00066 #define SETUP_ENET1_TXF_ISR(f, l) SetIntc(0, (long)f, 49, l)
00067 #define SETUP_ENET1_TXB_ISR(f, l) SetIntc(0, (long)f, 50, l)
00068 #define SETUP_ENET1_UN_ISR(f, l) SetIntc(0, (long)f, 51, l)
00069 #define SETUP_ENET1_RL_ISR(f, l) SetIntc(0, (long)f, 52, l)
00070 #define SETUP_ENET1_RXF_ISR(f, l) SetIntc(0, (long)f, 53, l)
00071 #define SETUP_ENET1_RXB_ISR(f, l) SetIntc(0, (long)f, 54, l)
00072 #define SETUP_ENET1_MII_ISR(f, l) SetIntc(0, (long)f, 55, l)
00073 #define SETUP_ENET1_LC_ISR(f, l) SetIntc(0, (long)f, 56, l)
00074 // 57 is unused
00075 #define SETUP_ENET1_GRA_ISR(f, l) SetIntc(0, (long)f, 58, l)
00076 #define SETUP_ENET1_EBERR_ISR(f, l) SetIntc(0, (long)f, 59, l)
00077 #define SETUP_ENET1_BABT_ISR(f, l) SetIntc(0, (long)f, 60, l)
00078 #define SETUP_ENET1_BABR_ISR(f, l) SetIntc(0, (long)f, 61, l)
00079
00080 #define SETUP_SCM_SCMIR_ISR(f, l) SetIntc(0, (long)f, 62, l)
00081 #define SETUP_OW_ISR_ISR(f, l) SetIntc(0, (long)f, 63, l)
00082
00083 #define SETUP_CAN0_IFLAG_ISR(f, l) SetIntc(1, (long)f, 0, l)
00084 #define SETUP_CAN0_ERRSTAT_BOF_ISR(f, l) SetIntc(1, (long)f, 1, l)
00085 // Unused
00086 #define SETUP_CAN0_ERRSTAT_TXRXWARN_ISR(f, l) SetIntc(1, (long)f, 3, l)
00087
00088 #define SETUP_CAN1_IFLAG_ISR(f, l) SetIntc(1, (long)f, 4, l)
00089 #define SETUP_CAN1_ERRSTAT_BOF_ISR(f, l) SetIntc(1, (long)f, 5, l)
00090 // Unused
00091 #define SETUP_CAN1_ERRSTAT_TXRXWARN_ISR(f, l) SetIntc(1, (long)f, 7, l)
00092
00093 #define SETUP_DMA16_ISR(f, l) SetIntc(1, (long)f, 8, l)
00094 #define SETUP_DMA17_ISR(f, l) SetIntc(1, (long)f, 9, l)
00095 #define SETUP_DMA18_ISR(f, l) SetIntc(1, (long)f, 10, l)
00096 #define SETUP_DMA19_ISR(f, l) SetIntc(1, (long)f, 11, l)
00097 #define SETUP_DMA20_ISR(f, l) SetIntc(1, (long)f, 12, l)
00098 #define SETUP_DMA21_ISR(f, l) SetIntc(1, (long)f, 13, l)
00099 #define SETUP_DMA22_ISR(f, l) SetIntc(1, (long)f, 14, l)
00100 #define SETUP_DMA23_ISR(f, l) SetIntc(1, (long)f, 15, l)
00101 #define SETUP_DMA24_ISR(f, l) SetIntc(1, (long)f, 16, l)
00102 #define SETUP_DMA25_ISR(f, l) SetIntc(1, (long)f, 17, l)
00103 #define SETUP_DMA26_ISR(f, l) SetIntc(1, (long)f, 18, l)
00104 #define SETUP_DMA27_ISR(f, l) SetIntc(1, (long)f, 19, l)
00105 #define SETUP_DMA28_ISR(f, l) SetIntc(1, (long)f, 20, l)
00106 #define SETUP_DMA29_ISR(f, l) SetIntc(1, (long)f, 21, l)
00107 #define SETUP_DMA30_ISR(f, l) SetIntc(1, (long)f, 22, l)
00108 #define SETUP_DMA31_ISR(f, l) SetIntc(1, (long)f, 23, l)
00109 #define SETUP_DMA32_ISR(f, l) SetIntc(1, (long)f, 24, l)
00110 #define SETUP_DMA33_ISR(f, l) SetIntc(1, (long)f, 25, l)
00111 #define SETUP_DMA34_ISR(f, l) SetIntc(1, (long)f, 26, l)
00112 #define SETUP_DMA35_ISR(f, l) SetIntc(1, (long)f, 27, l)
00113 #define SETUP_DMA36_ISR(f, l) SetIntc(1, (long)f, 28, l)
```

```
00114 #define SETUP_DMA37_ISR(f, l) SetIntc(1, (long)f, 29, l)
00115 #define SETUP_DMA38_ISR(f, l) SetIntc(1, (long)f, 30, l)
00116 #define SETUP_DMA39_ISR(f, l) SetIntc(1, (long)f, 31, l)
00117 #define SETUP_DMA40_ISR(f, l) SetIntc(1, (long)f, 32, l)
00118 #define SETUP_DMA41_ISR(f, l) SetIntc(1, (long)f, 33, l)
00119 #define SETUP_DMA42_ISR(f, l) SetIntc(1, (long)f, 34, l)
00120 #define SETUP_DMA43_ISR(f, l) SetIntc(1, (long)f, 35, l)
00121 #define SETUP_DMA44_ISR(f, l) SetIntc(1, (long)f, 36, l)
00122 #define SETUP_DMA45_ISR(f, l) SetIntc(1, (long)f, 37, l)
00123 #define SETUP_DMA46_ISR(f, l) SetIntc(1, (long)f, 38, l)
00124 #define SETUP_DMA47_ISR(f, l) SetIntc(1, (long)f, 39, l)
00125 #define SETUP_DMA48_ISR(f, l) SetIntc(1, (long)f, 40, l)
00126 #define SETUP_DMA49_ISR(f, l) SetIntc(1, (long)f, 41, l)
00127 #define SETUP_DMA50_ISR(f, l) SetIntc(1, (long)f, 42, l)
00128 #define SETUP_DMA51_ISR(f, l) SetIntc(1, (long)f, 43, l)
00129 #define SETUP_DMA52_ISR(f, l) SetIntc(1, (long)f, 44, l)
00130 #define SETUP_DMA53_ISR(f, l) SetIntc(1, (long)f, 45, l)
00131 #define SETUP_DMA54_ISR(f, l) SetIntc(1, (long)f, 46, l)
00132 #define SETUP_DMA55_ISR(f, l) SetIntc(1, (long)f, 47, l)
00133
00134 #define SETUP_UART4_ISR(f, l) SetIntc(1, (long)f, 48, l)
00135 #define SETUP_UART5_ISR(f, l) SetIntc(1, (long)f, 49, l)
00136 #define SETUP_UART6_ISR(f, l) SetIntc(1, (long)f, 50, l)
00137 #define SETUP_UART7_ISR(f, l) SetIntc(1, (long)f, 51, l)
00138 #define SETUP_UART8_ISR(f, l) SetIntc(1, (long)f, 52, l)
00139 #define SETUP_UART9_ISR(f, l) SetIntc(1, (long)f, 53, l)
00140
00141 #define SETUP_DSPI1_ISR(f, l) SetIntc(1, (long)f, 54, l)
00142 #define SETUP_DSPI2_ISR(f, l) SetIntc(1, (long)f, 55, l)
00143 #define SETUP_DSPI3_ISR(f, l) SetIntc(1, (long)f, 56, l)
00144
00145 #define SETUP_I2C1_ISR(f, l) SetIntc(1, (long)f, 57, l)
00146 #define SETUP_I2C2_ISR(f, l) SetIntc(1, (long)f, 58, l)
00147 #define SETUP_I2C3_ISR(f, l) SetIntc(1, (long)f, 59, l)
00148 #define SETUP_I2C4_ISR(f, l) SetIntc(1, (long)f, 60, l)
00149 #define SETUP_I2C5_ISR(f, l) SetIntc(1, (long)f, 61, l)
00150 // 62 and 63 unused
00151
00152 #define SETUP_DMA56_63_ISR(f, l) SetIntc(2, (long)f, 0, l)
00153
00154 #define SETUP_PWM_SM0SR_CFM_ISR(f, l) SetIntc(2, (long)f, 1, l)
00155 #define SETUP_PWM_SM1SR_CFM_ISR(f, l) SetIntc(2, (long)f, 2, l)
00156 #define SETUP_PWM_SM2SR_CFM_ISR(f, l) SetIntc(2, (long)f, 3, l)
00157 #define SETUP_PWM_SM3SR_CFM_ISR(f, l) SetIntc(2, (long)f, 4, l)
00158
00159 #define SETUP_PWM_SM0SR_RF_ISR(f, l) SetIntc(2, (long)f, 5, l)
00160 #define SETUP_PWM_SM1SR_RF_ISR(f, l) SetIntc(2, (long)f, 6, l)
00161 #define SETUP_PWM_SM2SR_RF_ISR(f, l) SetIntc(2, (long)f, 7, l)
00162 #define SETUP_PWM_SM3SR_RF_ISR(f, l) SetIntc(2, (long)f, 8, l)
00163
00164 #define SETUP_PWM_FSIR_ISR(f, l) SetIntc(2, (long)f, 9, l)
00165 #define SETUP_PWM_SMSR_REF_ISR(f, l) SetIntc(2, (long)f, 10, l)
00166
00167 #define SETUP_PLL_LOCF_ISR(f, l) SetIntc(2, (long)f, 11, l)
00168 #define SETUP_PLL_LOLF_ISR(f, l) SetIntc(2, (long)f, 12, l)
00169
00170 #define SETUP_PIT0_ISR(f, l) SetIntc(2, (long)f, 13, l)
00171 #define SETUP_PIT1_ISR(f, l) SetIntc(2, (long)f, 14, l)
00172 #define SETUP_PIT2_ISR(f, l) SetIntc(2, (long)f, 15, l)
00173 #define SETUP_PIT3_ISR(f, l) SetIntc(2, (long)f, 16, l)
00174
00175 #define SETUP_USB_OTG_ISR(f, l) SetIntc(2, (long)f, 17, l)
00176 #define SETUP_USB_HOST_ISR(f, l) SetIntc(2, (long)f, 18, l)
00177
00178 #define SETUP_PWM_SM0SR_CMPF_ISR(f, l) SetIntc(2, (long)f, 19, l)
00179 #define SETUP_PWM_SM1SR_CMPF_ISR(f, l) SetIntc(2, (long)f, 20, l)
00180 #define SETUP_PWM_SM2SR_CMPF_ISR(f, l) SetIntc(2, (long)f, 21, l)
00181 #define SETUP_PWM_SM3SR_CMPF_ISR(f, l) SetIntc(2, (long)f, 22, l)
00182
00183 #define SETUP_SSI0_ISR(f, l) SetIntc(2, (long)f, 23, l)
00184 #define SETUP_SSI1_ISR(f, l) SetIntc(2, (long)f, 24, l)
00185
00186 #define SETUP_NFC_ISR(f, l) SetIntc(2, (long)f, 25, l)
00187
00188 #define SETUP_RTC_ISR(f, l) SetIntc(2, (long)f, 26, l)
00189
00190 #define SETUP_USB_SR_ISR(f, l) SetIntc(2, (long)f, 27, l)
00191
00192 #define SETUP_RNG_ISR(f, l) SetIntc(2, (long)f, 28, l)
00193
00194 #define SETUP_SIM_TSR_ISR(f, l) SetIntc(2, (long)f, 29, l)
00195 #define SETUP_SIM_RSR_ISR(f, l) SetIntc(2, (long)f, 30, l)
00196
00197 #define SETUP_SDHC_ISR(f, l) SetIntc(2, (long)f, 31, l)
00198
00199 #define SETUP_ADC_SR_EOSI0_ISR(f, l) SetIntc(2, (long)f, 32, l)
00200 #define SETUP_ADC_SR_EOSI1_ISR(f, l) SetIntc(2, (long)f, 33, l)
```

```
00201 #define SETUP_ADC_LSR_ISR(f, l) SetIntc(2, (long)f, 34, l)
00202 // 35 na
00203 #define SETUP_SDRAM_ISR(f, l) SetIntc(2, (long)f, 36, l)
00204 // 37 na
00205
00206 #define SETUP_ENET_SW_EBERR_ISR(f, l) SetIntc(2, (long)f, 38, l)
00207 #define SETUP_ENET_SW_RXB_ISR(f, l) SetIntc(2, (long)f, 39, l)
00208 #define SETUP_ENET_SW_RXF_ISR(f, l) SetIntc(2, (long)f, 40, l)
00209 #define SETUP_ENET_SW_TXB_ISR(f, l) SetIntc(2, (long)f, 41, l)
00210 #define SETUP_ENET_SW_TXF_ISR(f, l) SetIntc(2, (long)f, 42, l)
00211 #define SETUP_ENET_SW_QM_ISR(f, l) SetIntc(2, (long)f, 43, l)
00212 #define SETUP_ENET_SW_OD0_ISR(f, l) SetIntc(2, (long)f, 44, l)
00213 #define SETUP_ENET_SW_OD1_ISR(f, l) SetIntc(2, (long)f, 45, l)
00214 #define SETUP_ENET_SW_OD2_ISR(f, l) SetIntc(2, (long)f, 46, l)
00215 #define SETUP_ENET_SW_LRN_ISR(f, l) SetIntc(2, (long)f, 47, l)
00216
00217 #define SETUP_MAC_NET0_EIR_TSAVAIL(f, l) SetIntc(2, (long)f, 48, l)
00218 #define SETUP_MAC_NET0_EIR_WAKE(f, l) SetIntc(2, (long)f, 49, l)
00219 #define SETUP_MAC_NET0_EIR_PLR(f, l) SetIntc(2, (long)f, 50, l)
00220 // 51,52,53,4 na
00221 #define SETUP_MAC_NET1_EIR_TSAVAIL(f, l) SetIntc(2, (long)f, 55, l)
00222 #define SETUP_MAC_NET1_EIR_WAKE(f, l) SetIntc(2, (long)f, 56, l)
00223 #define SETUP_MAC_NET1_EIR_PLR(f, l) SetIntc(2, (long)f, 57, l)
00224 // 58..63 NA
00225
00226 #endif /* _SIM_H_ */
```

24.18 mcf5441x_rtc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifdef __cplusplus
00005 extern "C"
00006 {
00007 #endif /* __cplusplus */
00008
00017     int MCF541X_RTCGetTime(struct tm &bts);
00018
00026     int MCF541X_RTCSetTime(struct tm &bts);
00027
00033     int MCF541X_RTCSetSystemFromRTCTime(void);
00034
00040     int MCF541X_RTCSetRTCfromSystemTime(void);
00041
00042     /* Read and write data to from the 2K bytes of back up SRAM */
00043     int MCF541X_GetBackUpRam(puint8_t pDest, int len);
00044
00045     int MCF541X_SetBackUpRam(puint8_t pSrc, int len);
00046
00047 #ifdef __cplusplus
00048 };
00049 #endif /* __cplusplus */
```

24.19 multican.h File Reference

ColdFire MCF5441x Control Area Network (CAN)

Classes

- class canMCF5441x::CanRxMessage

  Class to hold received CAN messages.

Namespaces

- namespace canMCF5441x

  canMCF5441x namespace

24.19.1 Detailed Description

ColdFire MCF5441x Control Area Network (CAN)

24.20 multican.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00012 #ifndef _MULTICAN_H
00013 #define _MULTICAN_H
00014
00015 #ifdef DOXYGEN_STUFF
00019 namespace canMCF5441x
00020 {
00021 #endif
00022
00033 #if (defined MOD5441X)
00034 #define DEFAULT_CAN_MOD (0)
00035 #elif (defined NANO54415)
00036 #define DEFAULT_CAN_MOD (1)
00037 #elif (defined SB800EX)
00038 #define DEFAULT_CAN_MOD (0)
00039 #endif
00040
00041 #define CAN_OK (0)
00042 #define CAN_RATE_FAIL (-1)
00043 #define CAN_ALREAD_OPEN (-2)
00044 #define CAN_CHANNEL_USED (-3)
00045 #define CAN_CHANNEL_NOT_USED (-4)
00046 #define CAN_TIMEOUT (-5)
00047
00048 #define DONT_WAIT (0xFFFF)
00049
00050 struct PrivateCanData;
00051
00055 class CanRxMessage
00056 {
00057    private:
00058     PrivateCanData *pData;
00059     /* Private constructor used for received frames */
00060     CanRxMessage(PrivateCanData *pData);
00061
00062     /* Helper method for the RTR constructors */
00063     void MultiCanSendRTR(int moduleNum, uint32_t id, uint16_t timeout, uint8_t length);
00064
00065     inline void CanSendRTR(uint32_t id, uint16_t timeout, uint8_t length) { return
      MultiCanSendRTR(DEFAULT_CAN_MOD, id, timeout, length); }
00066
00067    public:
00071     uint8_t GetLength();
00072
00081     uint8_t GetData(uint8_t *buffer, uint8_t max_len);
00082
00086     uint32_t GetId();
00087
00091     uint16_t GetTimeStamp();
00092
00099     BOOL IsValid();
00100
00101     /* Constructors */
00102
00113     CanRxMessage(OS_FIFO *pFifo, uint16_t timeout);
00114
00127     CanRxMessage(int moduleNum, uint32_t id, uint16_t timeout);
00128
00129     CanRxMessage(int moduleNum, uint32_t id, uint16_t timeout, uint8_t length);
00130
00134     ~CanRxMessage();
00135
00144     BOOL GetNewMessage(OS_FIFO *pFifo, uint16_t timeout);
00145 };
00146
00147 /*****************************************************************************
00148
00149                 A special note about CAN identifiers
00150
00151 *****************************************************************************
00152  Can identifiers come in two flavors, Normal (11 bits ) and Extended. (29 bits)
00153  In this software API we always refer to can identifiers as 32 bit uint32_tS.
00154  A 32 bit uint32_t is bigger than either Identifier.
00155
00156  A Normal identifier will always have bits 0 to 17 as zero
00157
00158  An extended identifier can have bits 0 to 17 low. So extended identifiers
00159  that are received will have bit 29 set to 1. Any ID input into the
00160  system will be treated as extended if bit 29 is set or if bits
00161  0 to 17 are not zero.
```

```
00162
00163  The following MACROS can help work with this ID scheme.
00164  */
00165
00166 /* The single bit used by this API to indicate an extended ID */
00167 #define CAN_EXTENDED_ID_BIT (0x20000000)
00168
00169 /* This macro takes and id and returs an ID that is guarenteed
00170 to be treated as an extended ID by the system
00171 Make an id that is extended from either extended or normal */
00172 #define ExtToNbId(id) (id | CAN_EXTENDED_ID_BIT)
00173
00174 /*Make Normal Id make an ID set from a normal id in the range 0 to 2048 */
00175 #define NormToNbId(id) ((id & 0x7ff) « 18)
00176
00177 /* Is the ID extended ?  returns non zero if the id passed in was an extended ID */
00178 #define IsNBIdExt(id) ((id & (CAN_EXTENDED_ID_BIT | 0x3FFFF)) != 0)
00179
00180 /* Strip the extra flag remove the API extended flage from the ID*/
00181 #define NbToExtId(id) (id & 0x1FFFFFFF)
00182
00183 /* Shift a Normal ID so it hase value 0 to 1023
00184 Some CAN systems will treat normal ID's as an integer from 0 to 2048
00185 Other systems may treat normal ID's as 28 bit values wheer the bottom 17 bits are zero
00186 This MACRO will convert our Normal ID format into the 0 to 2048 format.*/
00187 #define NbToNormId(id) ((id » 18) & 0x7FF)
00188
00189 /* Initialize the CAN system
00190 parameters:
00191
00192    bit_rate is the bitrate to run the CAN system at.
00193          The system will get as close as possible, but 1000000, 500000, 250000 and 125000 are
00194          the only values that are known to work.
00195
00196    Global_Mask is the mask used to mask received ID's a bit =0 is don't care, 1= care.
00197
00198    irq_level The interrupt level you want the CAN system to operate at.
00199
00200 Returns
00201    CAN_OK on success.
00202    CAN_RATE_FAIL if the bit rate could not be set withing 1.5%
00203    CAN_ALREADOPEN if the CAN system is already running. Youmust call CanShutDown first.
00204 */
00205 int MultiCanInit(int moduleNum, uint32_t bit_rate, uint32_t Global_Mask, uint8_t irq_level = 4);
00206
00207 /* Shut down the CAN system */
00208 void MultiCanShutDown(int moduleNum);
00209
00210 /* Change the global receive mask after the CAN system is started. */
00211 void MultiCanChangeGlobalMask(int moduleNum, uint32_t Global_Mask);
00212
00213 /* The CAN system has 16 availible channels this tells how many are currently in use */
00214 int MultiCanFreeCanChannels(int moduleNum);
00215
00216 /* This tells if a specific channel is currently free */
00217 BOOL MultiCanIsChannelFree(int moduleNum, int channel);
00218
00219 /* RegisterCanRxFifo and RegisterCanSpecialRxFifo
00220 Tell the CAN system to start listening for a specific CAN ID.
00221 Any incoming CAN frames that match the id as set by the apropriate mask will
00222 be placed into the fifo.
00223
00224 Parameters:
00225    uint32_t id The identifer to match on received frames. This is modified by
00226             the global mask, or in the case of the RegisterCanSpecial
00227             the passed in mask.
00228
00229    uint32_t spl_mask Only used by RegisterCanSpecialRxFifo. There are only
00230                two channels availible for use with the speical mask
00231                so use this call sparinging and only if really needed.
00232
00233    OSFifo * pFifo The Fifo used to communicate between the CAN subsystem and
00234                the CanRxMessageClass. This FIFO must be initialized before use.
00235                The same fifo can be passed to multiple receive regestration functions.
00236
00237    int channel The Channel to place the receive request. A value of -1 allows the system to
00238             select an unused channel.
00239
00240
00241 Returns
00242    Any value >=0 the channel this request is assigned to. This value must be stored to
00243             later call UnRegisterCanFifo.
00244    CAN_CHANNEL_USED If the channel is used or there are no free channels.
00245
00246 */
00247 int RegisterMultiCanRxFifo(int moduleNum, uint32_t id, OS_FIFO *pFifo, int channel = -1);
00248 int RegisterMultiCanSpecialRxFifo(int moduleNum, uint32_t id, uint32_t spl_mask, OS_FIFO *pFifo, int
```

```
    channel = -1);
00249
00250 /* disconnect a receiver channel form a Fifo
00251 Parameters:
00252    int channel The channel to remove.
00253
00254 returns
00255    CAN_OK if succesful.
00256    CAN_CHANNEL_NOT_USED if the channel was not currently in use
00257 */
00258 int UnRegisterMultiCanFifo(int moduleNum, int channel);
00259
00260 /* Send a message
00261 Parameters:
00262    uint32_t id The Identifier to send. See the earlier discussion of identifiers.
00263          and the identifier macros.
00264
00265    uint8_t * data A pointer to the data to send
00266
00267    uint8_t len The length of the Data must be <=8.
00268
00269    uint16_t timeout How lont to wait for confirmation it sent.
00270       0 = wait forevver.
00271       0xFFFF = don't wait at all.
00272
00273    int channel The channel to use. A value of -1
00274        will allow the system to pick and unused channel.
00275
00276 Returns
00277    CAN_OK   If the message was sent
00278    CAN_CHANNEL_USED Can't send because the channel was already in use or no channels availible.
00279    CAN_TIMEOUT Did not send in the time alotted
00280 */
00281 int MultiCanSendMessage(int moduleNum, uint32_t id, uint8_t *data, uint8_t len, uint16_t timeout, int
    channel = -1);
00282
00283 /*RTR messages
00284 Can can be configured to send messages in response to memessage requests.
00285 The next three functions handle that functionality */
00286
00287 /* Set RTRMessage
00288 This sets up a message to be sent in response to an incoming message that matches the
00289 RTR message id. The system will continue to respond with this message forever.
00290
00291 Parameters:
00292    uint32_t id. The id to respond to and the id responded with.
00293    uint8_t * data The data to respond with
00294    uint8_t len The length of the responding data.
00295    int channel The channel to use, or -1 to let the system pic one.
00296
00297 Returns:
00298    value >0 the channel assigned to this tsk or the channel requested.
00299    CAN_CHANNEL_USED if the channel is already used or no free channels are availible.
00300 */
00301 int MultiCanSetRTRMessage(int moduleNum, uint32_t id, uint8_t *data, uint8_t len, int channel = -1);
00302
00303 /*ReplaceRTRMessage
00304 Replace the outgoing message in an RTR channel already set up
00305 with a call to SetRTRMessage. Use of this function can
00306 cause a brief instant (a few uSec.) when no message response would be sent
00307 to an incoming request. If this is unacceptible register two identical RTR
00308 messages and change them in sequence.
00309
00310
00311 Parameters:
00312    int channel The channel value returned from the SetRTRMessage call
00313    uint8_t * data the data to send.
00314    uint8_t len the length of the data.
00315
00316
00317 Returns
00318    CAN_OK if the replacement succeded.
00319    CAN_CHANNEL_NOT_USED if the passed in channed was not set up for RTR messages
00320
00321 */
00322 int MultiCanReplaceRTRMessage(int moduleNum, int channel, uint8_t *data, uint8_t len);
00323
00324 /* Stop an RTR message
00325 Parameters:
00326    int channel The channel value returned from the SetRTRMessage call
00327
00328 Returns
00329    CAN_OK if the replacement succeded.
00330    CAN_CHANNEL_NOT_USED if the passed in channed was not set up for RTR messages
00331
00332 */
00333 int MultiCanStopRTRMessage(int moduleNum, int channel);
```

```
00334
00335 // Should a user include functions used in the canif.h file, these inline functions
00336 // will call the MultiCan module function associated with that function.
00337
00338 inline int CanInit(uint32_t bit_rate, uint32_t Global_Mask, uint8_t irq_level = 4)
00339 {
00340     return MultiCanInit(DEFAULT_CAN_MOD, bit_rate, Global_Mask, irq_level);
00341 }
00342
00343 inline void CanShutDown(void)
00344 {
00345     return MultiCanShutDown(DEFAULT_CAN_MOD);
00346 }
00347
00348 inline void ChangeGlobalMask(uint32_t Global_Mask)
00349 {
00350     return MultiCanChangeGlobalMask(DEFAULT_CAN_MOD, Global_Mask);
00351 }
00352
00353 inline int RegisterCanRxFifo(uint32_t id, OS_FIFO *pFifo, int channel = -1)
00354 {
00355     return RegisterMultiCanRxFifo(DEFAULT_CAN_MOD, id, pFifo, channel);
00356 }
00357
00358 inline int RegisterCanSpecialRxFifo(uint32_t id, uint32_t spl_mask, OS_FIFO *pFifo, int channel = -1)
00359 {
00360     return RegisterMultiCanSpecialRxFifo(DEFAULT_CAN_MOD, id, spl_mask, pFifo, channel);
00361 }
00362
00363 inline int UnRegisterCanFifo(int channel)
00364 {
00365     return UnRegisterMultiCanFifo(DEFAULT_CAN_MOD, channel);
00366 }
00367
00368 inline int SendMessage(uint32_t id, uint8_t *data, uint8_t len, uint16_t timeout, int channel = -1)
00369 {
00370     return MultiCanSendMessage(DEFAULT_CAN_MOD, id, data, len, timeout, channel);
00371 }
00372
00373 inline int SetRTRMessage(uint32_t id, uint8_t *data, uint8_t len, int channel = -1)
00374 {
00375     return MultiCanSetRTRMessage(DEFAULT_CAN_MOD, id, data, len, channel);
00376 }
00377
00378 inline int ReplaceRTRMessage(int channel, uint8_t *data, uint8_t len)
00379 {
00380     return MultiCanReplaceRTRMessage(DEFAULT_CAN_MOD, channel, data, len);
00381 }
00382
00383 inline int StopRTRMessage(int channel)
00384 {
00385     return MultiCanStopRTRMessage(DEFAULT_CAN_MOD, channel);
00386 }
00387
00388 /* The CAN system has 16 availible channels this tells how many are currently in use */
00389 inline int FreeCanChannels()
00390 {
00391     return MultiCanFreeCanChannels(DEFAULT_CAN_MOD);
00392 }
00393
00394 /* This tells if a specific channel is currently free */
00395 inline BOOL IsChannelFree(int channel)
00396 {
00397     return MultiCanIsChannelFree(DEFAULT_CAN_MOD, channel);
00398 }
00399
00400 /*****************************************************************************
00401  Example #1
00402
00403  To set up to receive frames:
00404
00405
00406 OS_FIFO fifo;
00407 OSFifoInit(&fifo);
00408
00409 //REgister to listen for CAN data
00410 //We will listed for frames with a normal is of 123
00411 int chan1=RegisterCanRxFifo( MakeIdFromNormal(123),& fifoifo);
00412
00413 //We will also listen for the extended frame id of 0x1234
00414 int chan2=RegisterCanRxFifo( ox1234,& fifoifo);
00415
00416 if ((chan1 >0) || (chan2> 0)_
00417  {//We succeded in registering the fifo
00418   while(1)
00419   {
00420    CanRxMessage can_msg(&fifo,30*TICKS_PER_SECOND); //Wait up to 20 seconds
```

```
00421    if (can_msg.IsValid())
00422     {uint8_t buffer[8];
00423      uint8_t len;
00424      len=can_msg.GetData(buffer,8);
00425
00426      iprintf("got a can message of %d bytes from ID %08X \r\n",len ,can_msg.GetId());
00427      iprintf("Data = [");
00428      for (int i=0; i<len; i++)
00429        iprintf("%02X ",buffer[i]);
00430      iprintf("]\r\n");
00431     }
00432    else
00433    iprintf("CAN received timed out ");
00434   }
00435  }
00436
00437 ********************************************************************************
00438 End of example #1
00439 ********************************************************************************/
00440
00441 /*******************************************************************************
00442  Example #2
00443  Setup an automatic response buffer (an RTR buffer)
00444
00445 // we want to setup an automatic response message (RTR
00446 //We will update it every 5 seconds
00447 uint16_t value=GetValue();
00448 int RTRChan=SetRTRMessage(0x5678,(puint8_t),&value,sizeof(value));
00449
00450
00451 while (1)
00452 {
00453 //Every 5 seconds update the value stored in the message
00454 OSTimeDly(5 * TICKS_PER_SECOND);
00455
00456 //Get the new value
00457 value=GetValue();
00458 //Store it in the automatic response channel
00459 ReplaceRTRMessage(RTRChan,(puint8_t),&value,sizeof(value) );
00460 }
00461
00462
00463 ********************************************************************************
00464 End of example #2
00465 ********************************************************************************/
00466
00469 #ifdef DOXYGEN_STUFF
00470 }   // namespace
00471 #endif
00472
00473 #endif
```

24.21 multichanneli2c.h File Reference

NetBurner I2C API for MOD5441x and NANO54415.

```
#include <basictypes.h>
#include <nbrtos.h>
#include <i2c_class.h>
```

Macros

- #define **I2C_OK** (0)

  Last instruction terminated correctly.
- #define **I2C_NEXT_WRITE_OK** (1)

  I2C bus is OK for a write.
- #define **I2C_NEXT_READ_OK** (2)

  I2C bus is OK for a read.
- #define **I2C_MASTER_OK** (3)

  I2C finished transmission but still owns but (need to stop or restart)
- #define **I2C_TIMEOUT** (4)

  A timeout occurred while trying communicate on I2C bus.
- #define **I2C_BUS_NOT_AVAIL** (5)

A timeout occurred while trying gain I2C bus control.

- #define **I2C_NOT_READY** (6)

  A read or write was attempted before I2C ready or during a slave transmission.

- #define **I2C_LOST_ARB** (7)

  Lost arbitration during start.

- #define **I2C_LOST_ARB_ADD** (8)

  Lost arbitration and then winner addressed our slave address.

- #define **I2C_NO_LINK_RX_ACK** (9)

  We are in Master TX mode and received no ACK from slave device, possibly during start.

- #define **I2C_SR_BUSY** (((0x20 & I2C_SR) == 0x20))

  Bus is busy (bit 5 of I2SR)

- #define **I2C_CR_SLAVE** (((0x20 & I2C_CR) == 0x00))

  Bus set as slave (bit 5 of I2CR)

- #define **I2C_SR_ARB_LOST** (((0x10 & I2C_SR) == 0x10))

  Bus arbitration was lost (bit 4 of I2SR)

- #define **I2C_SR_ADRES_AS_SLAVE** (((0x40 & I2C_SR) == 0x40))

  Addressed as a slave (bit 6 of I2SR)

- #define **I2C_SR_SLAVE_TX** (((0x04 & I2C_SR) == 0x04))

  State of read/write bit of the received address command (bit 2 of I2SR)

- #define **I2C_CR_TX** (((0x10 & I2C_CR) == 0x10))

  Configured for transmit (bit 4 of I2CR)

- #define **I2C_SR_RX_ACK** (((0x01 & I2C_SR) == 0x00))

  Received a RX ACK after last transmit (bit 0 of I2SR)

- #define **I2C_CR_RX_ACK** (((0x08 & I2C_CR) == 0x00))

  Configured to RX Ack.

- #define **I2C_SET_NO_ACK** ((I2C_CR |= 0x08))

  Configure I2C module not to send a RX ACK (bit 3 of I2CR)

- #define **I2C_SET_ACK** ((I2C_CR &= 0xF7))

  Configure I2C module to send a RX ACK (bit 3 of I2CR)

- #define **I2C_SET_TX** ((I2C_CR |= 0x10))

  Configure I2C module to be in TX mode (bit 4 of I2CR)

- #define **I2C_SET_RX** ((I2C_CR &= 0xEF))

  Configure I2C module to be in RX mode (bit 4 of I2CR)

- #define **I2C_SET_REPEAT_START** ((I2C_CR |= 0x04))

  Configure I2C to send a repeated start signal.

- #define **I2C_CLR_ARB_LOST** ((I2C_SR &= 0xEF))

  Clear Arbitration lost error condition.

Functions

- void MultiChannel_I2CInit (int moduleNum=DEFAULT_I2C_MODULE, uint8_t slave_Addr=0x08, uint8_t freqdiv=0x3C)

  Initialize the I2C peripheral module.

- uint8_t MultiChannel_I2CSendBuf (int moduleNum, uint8_t addr, puint8_t buf, int num, bool stop=true)

  Send a buffer of bytes to an I2C device.

- uint8_t MultiChannel_I2CReadBuf (int moduleNum, uint8_t addr, puint8_t buf, int num, bool stop=true)

  Read a number of bytes from an I2C device and store in the specified buffer.

- void I2CMultiChannelResetPeripheral (int moduleNum)

  Reset the specified I2C peripheral module.

- uint8_t MultiChannel_I2CRestart (int moduleNum, uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

Restart communication with a I2C device.

- uint8_t MultiChannel_I2CStart (int moduleNum, uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait=I2C_START_TIMEOUT)

  Send an I2C start to an I2C device to begin communication.
- uint8_t MultiChannel_I2CStop (int moduleNum=DEFAULT_I2C_MODULE, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Issue an I2C stop terminate communication with an I2C device and release the bus.
- uint8_t MultiChannel_I2CSend (int moduleNum, uint8_t val, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Send a single byte on the I2C bus.
- uint8_t MultiChannel_I2CRead (int moduleNum, puint8_t val, uint32_t ticks_to_wait=I2C_RX_TX_TIMEOUT)

  Read a single byte from the I2C bus.

24.21.1 Detailed Description

NetBurner I2C API for MOD5441x and NANO54415.

24.22 multichanneli2c.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-------------------------------------------------------------------------
00006    NetBurner I2C Multi-Master I2C Driver.
00007    Multi-Master I2C will enable operation as a master device on the I2C bus when
00008    activly seeking to communicate.  When idle,  the NetBurner will act as a slave
00009    device so other I2C masters can use the bus.
00010
00011    Note: If your NetBurner device is the only master on the I2C bus, you may
00012    choose to use the Master-Only I2C driver to conserve system resources
00013 -------------------------------------------------------------------------*/
00014
00050 #ifndef _MULTICHANNELI2C_H
00051 #define _MULTICHANNELI2C_H
00052 #include <basictypes.h>
00053 #include <nbrtos.h>
00054 #include <i2c_class.h>
00055
00056 #ifdef __cplusplus
00057 extern "C"
00058 {
00059 #endif
00060
00061 #define I2C_MAX_BUF_SIZE (64)   // Size allocated to input and output buffers in slave mode I2C
00062
00063 #define DEFAULT_I2C_MODULE (0)
00064
00065 /* defined I2C Timeout values will be used if user does not include the 'ticks_to_wait'
00066    parameter when calling I2C functions                                          */
00067 #define I2C_RX_TX_TIMEOUT (5)    // Ticks allowed before timeout of a single byte transmission
00068 #define I2C_START_TIMEOUT (20)   // Ticks allowed before timeout when attempting start on I2C bus
00069
00070 #define I2C_SLAVE_TX_TERM_CHAR (0)   // Terminating char to be sent when Slave TX buffer is empty
00071
00077 #define I2C_OK (0)
00078 #define I2C_NEXT_WRITE_OK (1)
00079 #define I2C_NEXT_READ_OK (2)
00080 #define I2C_MASTER_OK (3)
00081 #define I2C_TIMEOUT (4)
00082 #define I2C_BUS_NOT_AVAIL (5)
00083 #define I2C_NOT_READY (6)
00084 #define I2C_LOST_ARB (7)
00085 #define I2C_LOST_ARB_ADD (8)
00086 #define I2C_NO_LINK_RX_ACK (9)
00089 // The following defines were created to allow the multi and master-only drivers to
00090 // have the same function calls.  i.e. Multi_I2CInit(); is the same as calling I2CInit();
00091 //#define I2CInit      MultiChannel_I2CInit
00092 //#define I2CSendBuf   MultiChannel_I2CSendBuf
00093 //#define I2CReadBuf   MultiChannel_I2CReadBuf
00094 //#define I2CRestart   MultiChannel_I2CRestart
00095 //#define I2CStart     MultiChannel_I2CStart
00096 //#define I2CStop      MultiChannel_I2CStop
00097 //#define I2CSend      MultiChannel_I2CSend
00098 //#define I2CRead      MultiChannel_I2CRead
```

```
00099
00100 // Setup sim references to i2c struct
00101 #define I2C_SR i2cModule->i2sr
00102 #define I2C_CR i2cModule->i2cr
00103 #define I2C_DR i2cModule->i2dr
00104 #define I2C_FDR i2cModule->i2fdr
00105 #define I2C_ADR i2cModule->i2adr
00106
00120     void MultiChannel_I2CInit(int moduleNum = DEFAULT_I2C_MODULE, uint8_t slave_Addr = 0x08, uint8_t
      freqdiv = 0x3C);
00121
00138     uint8_t MultiChannel_I2CSendBuf(int moduleNum, uint8_t addr, puint8_t buf, int num, bool stop =
      true);
00139
00156     uint8_t MultiChannel_I2CReadBuf(int moduleNum, uint8_t addr, puint8_t buf, int num, bool stop =
      true);
00157
00165     void I2CMultiChannelResetPeripheral(int moduleNum);
00166
00167     /*-------------------------------------------------------------------------
00168     bool I2CRXAvail();
00169
00170     returns true if data is available in the Slave RX buffer
00171     */
00172     bool I2CRXAvail();
00173
00174     /*-------------------------------------------------------------------------
00175     uint32_t I2CTXAvail();
00176
00177     returns amount of free space available in Slave TX buffer
00178     */
00179     uint32_t I2CTXAvail();
00180
00181     /*-------------------------------------------------------------------------
00182     uint8_t I2CGetByte();
00183
00184     This function will pend on a slave recieve I2C semaphore
00185
00186     returns last unread uint8_t received as a I2C Slave
00187     */
00188     uint8_t I2CGetByte();
00189
00190     /*-------------------------------------------------------------------------
00191     bool I2CFillSlaveTXBuf(puint8_t buf, uint32_t num, bool restart = true);
00192
00193     This will put the first 'num' BYTES from the buffer 'buf' into the
00194     I2C Slave TX buffer and clear the previously-read contents of the buffer.
00195     The 'restart' parameter tells the slave transmitter how to terminate
00196     if 'restart' = true then restart next tx from begining TX buffer (A new slave fill replaces
      buffer);
00197     if 'false' continue next tx from last slave tx,(A new slave fill adds to buffer at last read
      byte);
00198     In I2C multi.h there is a I2C_SLAVE_TX_TERM_CHAR defined that will decide the terminating char (if
      any) to send
00199     when the slave tx buffer is empty
00200
00201     returns false if failed to copy data
00202     */
00203     uint8_t I2CFillSlaveTXBuf(puint8_t buf, uint32_t num, bool restart = true);
00204
00205     /*-------------------------------------------------------------------------
00206     uint8_t ( *I2C_SlaveTX_Callback )( );
00207
00208     If this callback is pointed to a function then it will be called when
00209     retreiving the data to be sent to the master.  This will bypass the circular
00210     buffer functions "I2CTXAvail" and "I2CFillSlaveTXBuf" used when transmitting
00211     data as a slave.
00212
00213     returns the data to be sent to the master
00214     */
00215     extern uint8_t (*I2C_SlaveTX_Callback)();
00216
00217     /*-------------------------------------------------------------------------
00218     void ( *I2C_SlaveTX_NAK_Callback )( );
00219
00220     If this callback is pointed to a function then it will be called when
00221     the recieving master device sends a NAK.  This indicates that the master
00222     device will not be reading any more data from the slave transmitter.
00223
00224     */
00225     extern void (*I2C_SlaveTX_NAK_Callback)();
00226
00227     /*-------------------------------------------------------------------------
00228     void ( *I2C_SlaveRX_Callback )( uint8_t RX_Data);
00229
00230     If this callback is pointed to a function then it will be called when
00231     storing the data received from the master.  This will bypass the circular
```

```
00232     buffer functions "I2CRXAvail" and "I2CGetByte" used when receiving
00233     data as a slave.
00234
00235     uint8_t RX_Data - Passes the received data to the callback function
00236
00237     returns nothing
00238     */
00239     extern void (*I2C_SlaveRX_Callback)(uint8_t RX_Data);
00240
00241 #define I2C_START_READ (1)   // defines to be used for bRead_Not_Write
00242 #define I2C_START_WRITE (0)
00260     uint8_t MultiChannel_I2CRestart(int moduleNum, uint8_t addr, bool Read_Not_Write, uint32_t
      ticks_to_wait = I2C_RX_TX_TIMEOUT);
00261
00262     /*-------------------------------------------------------------------------------------------
00263           Advanced NetBurner I2C Functions
00264
00265     The following functions are used for advanced communications on using the I2C bus.  These
00266     functions are useful if user wishes to talk to a device that does not follow the Phillips
00267     I2C standard.  One example is an eeprom that first must be addressed and then read from
00268     with no restart in between.
00269     Data must be sent or received a byte at a time.  Also user must check function returns
00270     to verify the status of the bus before performing the next option.
00271     -------------------------------------------------------------------------------------------
00272     ===========================================================================================*/
00273
00291     uint8_t MultiChannel_I2CStart(int moduleNum, uint8_t addr, bool Read_Not_Write, uint32_t
      ticks_to_wait = I2C_START_TIMEOUT);
00292
00306     uint8_t MultiChannel_I2CStop(int moduleNum = DEFAULT_I2C_MODULE, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT);
00307
00324     uint8_t MultiChannel_I2CSend(int moduleNum, uint8_t val, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT);
00325
00346     uint8_t MultiChannel_I2CRead(int moduleNum, puint8_t val, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT);
00347
00348     // Struct that contains I2C Slave I/O Buffers
00349     struct I2C_Slave_Record
00350     {
00351         OS_SEM I2C_Slave_RX_Semaphore;   // semaphore used to determine when a slave RX interrupt
      occurs
00352
00353         volatile uint8_t *pI2CRxbuf;
00354         volatile uint8_t *pI2CTxbuf;
00355
00356         volatile uint32_t I2Crx_put;
00357         volatile uint32_t I2Crx_get;
00358
00359         volatile uint32_t I2Ctx_put;
00360         volatile uint32_t I2Ctx_get;
00361     };
00362
00363 /*-------------------------------------------------------------------------
00364          Useful I2C macros
00365 -------------------------------------------------------------------------*/
00371 #define I2C_SR_BUSY (((0x20 & I2C_SR) == 0x20))
00372 #define I2C_CR_SLAVE (((0x20 & I2C_CR) == 0x00))
00373 #define I2C_SR_ARB_LOST (((0x10 & I2C_SR) == 0x10))
00374 #define I2C_SR_ADRES_AS_SLAVE (((0x40 & I2C_SR) == 0x40))
00375 #define I2C_SR_SLAVE_TX (((0x04 & I2C_SR) == 0x04))
00376
00377 #define I2C_CR_TX (((0x10 & I2C_CR) == 0x10))
00378 #define I2C_SR_RX_ACK (((0x01 & I2C_SR) == 0x00))
00379 #define I2C_CR_RX_ACK (((0x08 & I2C_CR) == 0x00))
00380
00381 #define I2C_SET_NO_ACK ((I2C_CR |= 0x08))
00382 #define I2C_SET_ACK ((I2C_CR &= 0xF7))
00383 #define I2C_SET_TX ((I2C_CR |= 0x10))
00384 #define I2C_SET_RX ((I2C_CR &= 0xEF))
00385 #define I2C_SET_REPEAT_START ((I2C_CR |= 0x04))
00386 #define I2C_CLR_ARB_LOST ((I2C_SR &= 0xEF))
00389     //-------------------------------------------------------------------------
00390
00391     inline void I2CInit(uint8_t slave_Addr = 0x08, uint8_t freqdiv = 0x3C)
00392     {
00393         MultiChannel_I2CInit(DEFAULT_I2C_MODULE, slave_Addr, freqdiv);
00394     }
00395
00396     inline void Mulit_I2CInit(uint8_t slave_Addr = 0x08, uint8_t freqdiv = 0x3C)
00397     {
00398         MultiChannel_I2CInit(DEFAULT_I2C_MODULE, slave_Addr, freqdiv);
00399     }
00400
00401     inline uint8_t I2CSendBuf(uint8_t addr, puint8_t buf, int num, bool stop = true)
00402     {
```

```
00403         return MultiChannel_I2CSendBuf(DEFAULT_I2C_MODULE, addr, buf, num, stop);
00404     }
00405
00406     inline uint8_t Multi_I2CSendBuf(uint8_t addr, puint8_t buf, int num, bool stop = true)
00407     {
00408         return MultiChannel_I2CSendBuf(DEFAULT_I2C_MODULE, addr, buf, num, stop);
00409     }
00410
00411     inline uint8_t I2CReadBuf(uint8_t addr, puint8_t buf, int num, bool stop = true)
00412     {
00413         return MultiChannel_I2CReadBuf(DEFAULT_I2C_MODULE, addr, buf, num, stop);
00414     }
00415
00416     inline uint8_t Multi_I2CReadBuf(uint8_t addr, puint8_t buf, int num, bool stop = true)
00417     {
00418         return MultiChannel_I2CReadBuf(DEFAULT_I2C_MODULE, addr, buf, num, stop);
00419     }
00420
00421     inline uint8_t I2CRestart(uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT)
00422     {
00423         return MultiChannel_I2CRestart(DEFAULT_I2C_MODULE, addr, Read_Not_Write, ticks_to_wait);
00424     }
00425
00426     inline uint8_t Multi_I2CRestart(uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT)
00427     {
00428         return MultiChannel_I2CRestart(DEFAULT_I2C_MODULE, addr, Read_Not_Write, ticks_to_wait);
00429     }
00430
00431     inline uint8_t I2CStart(uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT)
00432     {
00433         return MultiChannel_I2CStart(DEFAULT_I2C_MODULE, addr, Read_Not_Write, ticks_to_wait);
00434     }
00435
00436     inline uint8_t Multi_I2CStart(uint8_t addr, bool Read_Not_Write, uint32_t ticks_to_wait =
      I2C_RX_TX_TIMEOUT)
00437     {
00438         return MultiChannel_I2CStart(DEFAULT_I2C_MODULE, addr, Read_Not_Write, ticks_to_wait);
00439     }
00440
00441     inline uint8_t I2CStop(uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT) { return
      MultiChannel_I2CStop(DEFAULT_I2C_MODULE, ticks_to_wait); }
00442
00443     inline uint8_t Multi_I2CStop(uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT)
00444     {
00445         return MultiChannel_I2CStop(DEFAULT_I2C_MODULE, ticks_to_wait);
00446     }
00447
00448     inline uint8_t I2CSend(uint8_t val, uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT)
00449     {
00450         return MultiChannel_I2CSend(DEFAULT_I2C_MODULE, val, ticks_to_wait);
00451     }
00452
00453     inline uint8_t Multi_I2CSend(uint8_t val, uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT)
00454     {
00455         return MultiChannel_I2CSend(DEFAULT_I2C_MODULE, val, ticks_to_wait);
00456     }
00457
00458     inline uint8_t I2CRead(puint8_t val, uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT)
00459     {
00460         return MultiChannel_I2CRead(DEFAULT_I2C_MODULE, val, ticks_to_wait);
00461     }
00462
00463     inline uint8_t Multi_I2CRead(puint8_t val, uint32_t ticks_to_wait = I2C_RX_TX_TIMEOUT)
00464     {
00465         return MultiChannel_I2CRead(DEFAULT_I2C_MODULE, val, ticks_to_wait);
00466     }
00467
00468 #ifdef __cplusplus
00469 }
00470 #endif
00471
00472 #endif
00473
```

24.23 periph_clocks.h

```
00001 #ifndef __PERIPH_CLOCKS_H
00002 #define __PERIPH_CLOCKS_H
00003
00004 #include <sim5441x.h>
00005
00006 // Peripheral Power Management Low Register 0
```

```
00007 #define PERIPH_CLOCK_FLEXBUS 2
00008 #define PERIPH_CLOCK_CAN_0 8
00009 #define PERIPH_CLOCK_CAN_1 9
00010 #define PERIPH_CLOCK_I2C_1 14
00011 #define PERIPH_CLOCK_DSPI_1 15
00012 #define PERIPH_CLOCK_DMA 17
00013 #define PERIPH_CLOCK_INTC_0 18
00014 #define PERIPH_CLOCK_INTC_1 19
00015 #define PERIPH_CLOCK_INTC_2 20
00016 #define PERIPH_CLOCK_I2C_0 22
00017 #define PERIPH_CLOCK_DSPI_0 23
00018 #define PERIPH_CLOCK_UART_0 24
00019 #define PERIPH_CLOCK_UART_1 25
00020 #define PERIPH_CLOCK_UART_2 26
00021 #define PERIPH_CLOCK_UART_3 27
00022 #define PERIPH_CLOCK_DMA_TMR_0 28
00023 #define PERIPH_CLOCK_DMA_TMR_1 29
00024 #define PERIPH_CLOCK_DMA_TMR_2 30
00025 #define PERIPH_CLOCK_DMA_TMR_3 31
00026
00027 // Peripheral Power Management High Register 0
00028 #define PERIPH_CLOCK_PIT_0 32
00029 #define PERIPH_CLOCK_PIT_1 33
00030 #define PERIPH_CLOCK_PIT_2 34
00031 #define PERIPH_CLOCK_PIT_3 35
00032 #define PERIPH_CLOCK_EDGE_PORT 36
00033 #define PERIPH_CLOCK_ADC 37
00034 #define PERIPH_CLOCK_DAC_0 38
00035 #define PERIPH_CLOCK_RTC 42
00036 #define PERIPH_CLOCK_SIM 43
00037 #define PERIPH_CLOCK_USB_OTG 44
00038 #define PERIPH_CLOCK_USB_HOST 45
00039 #define PERIPH_CLOCK_DDR 46
00040 #define PERIPH_CLOCK_SSI_0 47
00041 #define PERIPH_CLOCK_PLL 48
00042 #define PERIPH_CLOCK_RNG 49
00043 #define PERIPH_CLOCK_SSI_1 50
00044 #define PERIPH_CLOCK_SDHC 52
00045 #define PERIPH_CLOCK_MACNET_0 53
00046 #define PERIPH_CLOCK_MACNET_1 54
00047 #define PERIPH_CLOCK_ETHERNET_SWITCH_0 55
00048 #define PERIPH_CLOCK_ETHERNET_SWITCH_1 56
00049 #define PERIPH_CLOCK_NAND_FLASH 63
00050
00051 // Peripheral Power Management High Register 1
00052 #define PERIPH_CLOCK_PWM 34
00053 #define PERIPH_CLOCK_CCM_RESET 36
00054 #define PERIPH_CLOCK_GPIO 37
00055
00056 // Peripheral Power Management Low Register 1
00057 #define PERIPH_CLOCK_1_WIRE 2
00058 #define PERIPH_CLOCK_I2C_2 4
00059 #define PERIPH_CLOCK_I2C_3 5
00060 #define PERIPH_CLOCK_I2C_4 6
00061 #define PERIPH_CLOCK_I2C_5 7
00062 #define PERIPH_CLOCK_DSPI_2 14
00063 #define PERIPH_CLOCK_DSPI_3 15
00064 #define PERIPH_CLOCK_UART_4 24
00065 #define PERIPH_CLOCK_UART_5 25
00066 #define PERIPH_CLOCK_UART_6 26
00067 #define PERIPH_CLOCK_UART_7 27
00068 #define PERIPH_CLOCK_UART_8 28
00069 #define PERIPH_CLOCK_UART_9 29
00070
00071 // Disable clocks in Power Management Low Register 0
00072 #define PERIPH_DISABLE_FLEXBUS sim2.scm.ppmsr0 = PERIPH_CLOCK_FLEXBUS
00073 #define PERIPH_DISABLE_CAN_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_CAN_0
00074 #define PERIPH_DISABLE_CAN_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_CAN_1
00075 #define PERIPH_DISABLE_I2C_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_I2C_1
00076 #define PERIPH_DISABLE_DSPI_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_DSPI_1
00077 #define PERIPH_DISABLE_DMA sim2.scm.ppmsr0 = PERIPH_CLOCK_DMA
00078 #define PERIPH_DISABLE_INTC_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_INTC_0
00079 #define PERIPH_DISABLE_INTC_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_INTC_1
00080 #define PERIPH_DISABLE_INTC_2 sim2.scm.ppmsr0 = PERIPH_CLOCK_INTC_2
00081 #define PERIPH_DISABLE_I2C_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_I2C_0
00082 #define PERIPH_DISABLE_DSPI_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_DSPI_0
00083 #define PERIPH_DISABLE_UART_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_UART_0
00084 #define PERIPH_DISABLE_UART_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_UART_1
00085 #define PERIPH_DISABLE_UART_2 sim2.scm.ppmsr0 = PERIPH_CLOCK_UART_2
00086 #define PERIPH_DISABLE_UART_3 sim2.scm.ppmsr0 = PERIPH_CLOCK_UART_3
00087 #define PERIPH_DISABLE_DMA_TMR_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_DMA_TMR_0
00088 #define PERIPH_DISABLE_DMA_TMR_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_DMA_TMR_1
00089 #define PERIPH_DISABLE_DMA_TMR_2 sim2.scm.ppmsr0 = PERIPH_CLOCK_DMA_TMR_2
00090
00091 // Enable clocks in Power Management Low Register 0
00092 #define PERIPH_ENABLE_FLEXBUS sim2.scm.ppmcr0 = PERIPH_CLOCK_FLEXBUS
00093 #define PERIPH_ENABLE_CAN_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_CAN_0
```

```
00094 #define PERIPH_ENABLE_CAN_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_CAN_1
00095 #define PERIPH_ENABLE_I2C_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_I2C_1
00096 #define PERIPH_ENABLE_DSPI_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_DSPI_1
00097 #define PERIPH_ENABLE_DMA sim2.scm.ppmcr0 = PERIPH_CLOCK_DMA
00098 #define PERIPH_ENABLE_INTC_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_INTC_0
00099 #define PERIPH_ENABLE_INTC_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_INTC_1
00100 #define PERIPH_ENABLE_INTC_2 sim2.scm.ppmcr0 = PERIPH_CLOCK_INTC_2
00101 #define PERIPH_ENABLE_I2C_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_I2C_0
00102 #define PERIPH_ENABLE_DSPI_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_DSPI_0
00103 #define PERIPH_ENABLE_UART_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_UART_0
00104 #define PERIPH_ENABLE_UART_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_UART_1
00105 #define PERIPH_ENABLE_UART_2 sim2.scm.ppmcr0 = PERIPH_CLOCK_UART_2
00106 #define PERIPH_ENABLE_UART_3 sim2.scm.ppmcr0 = PERIPH_CLOCK_UART_3
00107 #define PERIPH_ENABLE_DMA_TMR_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_DMA_TMR_0
00108 #define PERIPH_ENABLE_DMA_TMR_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_DMA_TMR_1
00109 #define PERIPH_ENABLE_DMA_TMR_2 sim2.scm.ppmcr0 = PERIPH_CLOCK_DMA_TMR_2
00110
00111 // Disable clocks in Power Management High Register 0
00112 #define PERIPH_DISABLE_PIT_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_PIT_0
00113 #define PERIPH_DISABLE_PIT_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_PIT_1
00114 #define PERIPH_DISABLE_PIT_2 sim2.scm.ppmsr0 = PERIPH_CLOCK_PIT_2
00115 #define PERIPH_DISABLE_PIT_3 sim2.scm.ppmsr0 = PERIPH_CLOCK_PIT_3
00116 #define PERIPH_DISABLE_EDGE_PORT sim2.scm.ppmsr0 = PERIPH_CLOCK_EDGE_PORT
00117 #define PERIPH_DISABLE_ADC sim2.scm.ppmsr0 = PERIPH_CLOCK_ADC
00118 #define PERIPH_DISABLE_DAC_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_DAC_0
00119 #define PERIPH_DISABLE_RTC sim2.scm.ppmsr0 = PERIPH_CLOCK_RTC
00120 #define PERIPH_DISABLE_SIM sim2.scm.ppmsr0 = PERIPH_CLOCK_SIM
00121 #define PERIPH_DISABLE_USB_OTG sim2.scm.ppmsr0 = PERIPH_CLOCK_USB_OTG
00122 #define PERIPH_DISABLE_USB_HOST sim2.scm.ppmsr0 = PERIPH_CLOCK_USB_HOST
00123 #define PERIPH_DISABLE_DDR sim2.scm.ppmsr0 = PERIPH_CLOCK_DDR
00124 #define PERIPH_DISABLE_SSI_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_SSI_0
00125 #define PERIPH_DISABLE_PLL sim2.scm.ppmsr0 = PERIPH_CLOCK_PLL
00126 #define PERIPH_DISABLE_RNG sim2.scm.ppmsr0 = PERIPH_CLOCK_RNG
00127 #define PERIPH_DISABLE_SSI_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_SSI_1
00128 #define PERIPH_DISABLE_SDHC sim2.scm.ppmsr0 = PERIPH_CLOCK_SDHC
00129 #define PERIPH_DISABLE_MACNET_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_MACNET_0
00130 #define PERIPH_DISABLE_MACNET_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_MACNET_1
00131 #define PERIPH_DISABLE_ETHERNET_SWITCH_0 sim2.scm.ppmsr0 = PERIPH_CLOCK_ETHERNET_SWITCH_0
00132 #define PERIPH_DISABLE_ETHERNET_SWITCH_1 sim2.scm.ppmsr0 = PERIPH_CLOCK_ETHERNET_SWITCH_1
00133 #define PERIPH_DISABLE_NAND_FLASH sim2.scm.ppmsr0 = PERIPH_CLOCK_NAND_FLASH
00134
00135 // Enable clocks in Power Management High Register 0
00136 #define PERIPH_ENABLE_PIT_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_PIT_0
00137 #define PERIPH_ENABLE_PIT_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_PIT_1
00138 #define PERIPH_ENABLE_PIT_2 sim2.scm.ppmcr0 = PERIPH_CLOCK_PIT_2
00139 #define PERIPH_ENABLE_PIT_3 sim2.scm.ppmcr0 = PERIPH_CLOCK_PIT_3
00140 #define PERIPH_ENABLE_EDGE_PORT sim2.scm.ppmcr0 = PERIPH_CLOCK_EDGE_PORT
00141 #define PERIPH_ENABLE_ADC sim2.scm.ppmcr0 = PERIPH_CLOCK_ADC
00142 #define PERIPH_ENABLE_DAC_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_DAC_0
00143 #define PERIPH_ENABLE_RTC sim2.scm.ppmcr0 = PERIPH_CLOCK_RTC
00144 #define PERIPH_ENABLE_SIM sim2.scm.ppmcr0 = PERIPH_CLOCK_SIM
00145 #define PERIPH_ENABLE_USB_OTG sim2.scm.ppmcr0 = PERIPH_CLOCK_USB_OTG
00146 #define PERIPH_ENABLE_USB_HOST sim2.scm.ppmcr0 = PERIPH_CLOCK_USB_HOST
00147 #define PERIPH_ENABLE_DDR sim2.scm.ppmcr0 = PERIPH_CLOCK_DDR
00148 #define PERIPH_ENABLE_SSI_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_SSI_0
00149 #define PERIPH_ENABLE_PLL sim2.scm.ppmcr0 = PERIPH_CLOCK_PLL
00150 #define PERIPH_ENABLE_RNG sim2.scm.ppmcr0 = PERIPH_CLOCK_RNG
00151 #define PERIPH_ENABLE_SSI_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_SSI_1
00152 #define PERIPH_ENABLE_SDHC sim2.scm.ppmcr0 = PERIPH_CLOCK_SDHC
00153 #define PERIPH_ENABLE_MACNET_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_MACNET_0
00154 #define PERIPH_ENABLE_MACNET_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_MACNET_1
00155 #define PERIPH_ENABLE_ETHERNET_SWITCH_0 sim2.scm.ppmcr0 = PERIPH_CLOCK_ETHERNET_SWITCH_0
00156 #define PERIPH_ENABLE_ETHERNET_SWITCH_1 sim2.scm.ppmcr0 = PERIPH_CLOCK_ETHERNET_SWITCH_1
00157 #define PERIPH_ENABLE_NAND_FLASH sim2.scm.ppmcr0 = PERIPH_CLOCK_NAND_FLASH
00158
00159 // Disable clocks in Power Management Low Register 1
00160 #define PERIPH_DISABLE_1_WIRE sim2.scm.ppmsr1 = PERIPH_CLOCK_1_WIRE
00161 #define PERIPH_DISABLE_I2C_2 sim2.scm.ppmsr1 = PERIPH_CLOCK_I2C_2
00162 #define PERIPH_DISABLE_I2C_3 sim2.scm.ppmsr1 = PERIPH_CLOCK_I2C_3
00163 #define PERIPH_DISABLE_I2C_4 sim2.scm.ppmsr1 = PERIPH_CLOCK_I2C_4
00164 #define PERIPH_DISABLE_I2C_5 sim2.scm.ppmsr1 = PERIPH_CLOCK_I2C_5
00165 #define PERIPH_DISABLE_DSPI_2 sim2.scm.ppmsr1 = PERIPH_CLOCK_DSPI_2
00166 #define PERIPH_DISABLE_DSPI_3 sim2.scm.ppmsr1 = PERIPH_CLOCK_DSPI_3
00167 #define PERIPH_DISABLE_UART_4 sim2.scm.ppmsr1 = PERIPH_CLOCK_UART_4
00168 #define PERIPH_DISABLE_UART_5 sim2.scm.ppmsr1 = PERIPH_CLOCK_UART_5
00169 #define PERIPH_DISABLE_UART_6 sim2.scm.ppmsr1 = PERIPH_CLOCK_UART_6
00170 #define PERIPH_DISABLE_UART_7 sim2.scm.ppmsr1 = PERIPH_CLOCK_UART_7
00171 #define PERIPH_DISABLE_UART_8 sim2.scm.ppmsr1 = PERIPH_CLOCK_UART_8
00172 #define PERIPH_DISABLE_UART_9 sim2.scm.ppmsr1 = PERIPH_CLOCK_UART_9
00173
00174 // Enable clocks in Power Management Low Register 1
00175 #define PERIPH_ENABLE_1_WIRE sim2.scm.ppmcr1 = PERIPH_CLOCK_1_WIRE
00176 #define PERIPH_ENABLE_I2C_2 sim2.scm.ppmcr1 = PERIPH_CLOCK_I2C_2
00177 #define PERIPH_ENABLE_I2C_3 sim2.scm.ppmcr1 = PERIPH_CLOCK_I2C_3
00178 #define PERIPH_ENABLE_I2C_4 sim2.scm.ppmcr1 = PERIPH_CLOCK_I2C_4
00179 #define PERIPH_ENABLE_I2C_5 sim2.scm.ppmcr1 = PERIPH_CLOCK_I2C_5
00180 #define PERIPH_ENABLE_DSPI_2 sim2.scm.ppmcr1 = PERIPH_CLOCK_DSPI_2
```

```
00181 #define PERIPH_ENABLE_DSPI_3 sim2.scm.ppmcr1 = PERIPH_CLOCK_DSPI_3
00182 #define PERIPH_ENABLE_UART_4 sim2.scm.ppmcr1 = PERIPH_CLOCK_UART_4
00183 #define PERIPH_ENABLE_UART_5 sim2.scm.ppmcr1 = PERIPH_CLOCK_UART_5
00184 #define PERIPH_ENABLE_UART_6 sim2.scm.ppmcr1 = PERIPH_CLOCK_UART_6
00185 #define PERIPH_ENABLE_UART_7 sim2.scm.ppmcr1 = PERIPH_CLOCK_UART_7
00186 #define PERIPH_ENABLE_UART_8 sim2.scm.ppmcr1 = PERIPH_CLOCK_UART_8
00187 #define PERIPH_ENABLE_UART_9 sim2.scm.ppmcr1 = PERIPH_CLOCK_UART_9
00188
00189 // Disable clocks in Power Management High Register 1
00190 #define PERIPH_DISABLE_PWM sim2.scm.ppmsr1 = PERIPH_CLOCK_PWM
00191 #define PERIPH_DISABLE_CCM_RESET sim2.scm.ppmsr1 = PERIPH_CLOCK_CCM_RESET
00192 #define PERIPH_DISABLE_GPIO sim2.scm.ppmsr1 = PERIPH_CLOCK_GPIO
00193
00194 // Enable clocks in Power Management High Register 1
00195 #define PERIPH_ENABLE_PWM sim2.scm.ppmcr1 = PERIPH_CLOCK_PWM
00196 #define PERIPH_ENABLE_CCM_RESET sim2.scm.ppmcr1 = PERIPH_CLOCK_CCM_RESET
00197 #define PERIPH_ENABLE_GPIO sim2.scm.ppmcr1 = PERIPH_CLOCK_GPIO
00198
00199 #endif /* ----- #ifndef __PERIPH_CLOCKS_H  ----- */
```

24.24 coldfire/cpu/MCF5441X/include/pin_irq.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00010 BOOL SetPinIrq(int pin, int polarity, void (*func)(void));
00011
00012 void EnableIrq(int pin);
00013 void DisableIrq(int pin);
```

24.25 cortex-m7/cpu/SAME70/include/pin_irq.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef  __PIN_IRQ_H
00006 #define  __PIN_IRQ_H
00007
00008
00009 #include <predef.h>
00010 #include <constants.h>
00011 #include <cpu_pins.h>
00012
00027 typedef void (*PinIrq_t) (int pio_idx, int pin);
00028
00045 bool SetPortIrqPriority( PinIO pin, int priority);
00046
00063 bool SetPortIrqPriority( int portNum, int priority);
00064
00081 bool SetPinIrq( PinIO pin, int polarity, PinIrq_t func);
00082
00087 void EnableIrq( PinIO pin );
00092 void DisableIrq( PinIO pin );
00093
00111 uint32_t ClearISR(PinIO pin);
00112
00113 #endif   /* ----- #ifndef __PIN_IRQ_H  ----- */
```

24.26 pitr_sem.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 extern volatile uint32_t gPitCount[4];
00006
00007 /*-------------------------------------------------------------------
00008  * Initialize a PIT timer to interrupt the specified numbe of times
00009  * per second. Maximum is 20,000 providing 50us per tick.
00010  *
00011  * PARAMETERS
00012  * timer          Select PIT timer. Value can be 1 or 2.
00013  *                 0 is reserved for the rtos system clock,
00014  *                 3 is reserved for the debugger
00015  *
00016  * pit_per_sec    Number of interrupts per second
00017  *
```

```
00018  * RETURNS
00019  * 0  = success
00020  * -1 = invalid channel
00021  *-------------------------------------------------------------------*/
00022
00023 // Posts a semaphore when timer expires
00024 int InitPitOSSem(int timer, OS_SEM *p_toSem, int pit_per_sec);
00025
00026 // Sets a flag when timer expires
00027 int InitPitOSFlag(int timer, OS_FLAGS *p_toFlag, uint32_t fv, int pit_per_sec);
00028
00029 // Calls a function when timer expires
00030 int InitPitInterruptCallback(int timer, void (*p_toCallbackFunc)(), int pit_per_sec);
00031
00032 // Clear semaphore/flag/callback associated with a timer
00033 void ClearTimerVariables(int timer);
```

24.27 coldfire/cpu/MCF5441X/include/sim.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SIM_H_
00006 #define _SIM_H_
00007
00008 #include "sim5441x.h"
00009
00010 #endif /* _SIM_H_ */
```

24.28 cortex-m7/cpu/SAME70/include/sim.h

```
00001 #ifndef  __SIM_H
00002 #define  __SIM_H
00003 #include <same70q21.h>
00004
00005 #endif   /* ----- #ifndef __SIM_H  ----- */
```

24.29 sim5441x.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SIM54418_H_
00006 #define _SIM54418_H_
00007
00008 /* #define ENHANCED_ETHER_BD */   // Enable enhanced buffer descriptors
00009
00010 typedef volatile unsigned char vubyte;
00011 typedef volatile unsigned short vuword;
00012 typedef volatile unsigned long vudword;
00013
00014 /*
00015  * RAPID GPIO
00016  */
00017 typedef struct
00018 {
00019     vuword dir0; /* 0x8C00_0000 -> 0x8C00_0001 - (Read)  Data Direction Register
00020                                                   (Write) Data Direction Register
      */
00021     vuword data; /* 0x8C00_0002 -> 0x8C00_0003 - (Read)  Write Data Register
00022                                                   (Write) Write Data Register
      */
00023     vuword enb;  /* 0x8C00_0004 -> 0x8C00_0005 - (Read)  Pin Enable Register
00024                                                   (Write) Pin Enable Register
      */
00025     vuword clr;  /* 0x8C00_0006 -> 0x8C00_0007 - (Read)  Write Data Register
00026                                                   (Write) Write Data Clear Register
      */
00027     vuword dir1; /* 0x8C00_0008 -> 0x8C00_0009 - (Read)  Data Direction Register
      */
00028     vuword set;  /* 0x8C00_000A -> 0x8C00_000B - (Read)  Write Data Register
00029                                                   (Write) Write Data Set Register
      */
00030     vuword dir2; /* 0x8C00_000C -> 0x8C00_000D - (Read)  Data Direction Register
      */
00031     vuword tog;  /* 0x8C00_000E -> 0x8C00_000F - (Read)  Write Data Register
```

```
00032                                                   (Write) Write Data Toggle Register
      */
00033 } rgpiostruct;
00034
00035 /*
00036  * 1-WIRE MODULE
00037  */
00038 typedef struct
00039 {
00040     vubyte cr;        /* 0xEC00_8000 -> 0xEC00_8000 - Control Register
      */
00041     vubyte pack00[3]; /* 0xEC00_8001 -> 0xEC00_8003 - RESERVED
      */
00042     vubyte div;       /* 0xEC00_8004 -> 0xEC00_8004 - Time Divider Register
      */
00043     vubyte pack01[3]; /* 0xEC00_8005 -> 0xEC00_8007 - RESERVED
      */
00044     vubyte rst;       /* 0xEC00_8008 -> 0xEC00_8008 - Reset Register
      */
00045     vubyte pack02[3]; /* 0xEC00_8009 -> 0xEC00_800B - RESERVED
      */
00046     vubyte cmd;       /* 0xEC00_800C -> 0xEC00_800C - Command Register
      */
00047     vubyte pack03[3]; /* 0xEC00_800D -> 0xEC00_800F - RESERVED
      */
00048     vubyte txrx;      /* 0xEC00_8010 -> 0xEC00_8010 - Transmit/Receive Register
      */
00049     vubyte pack04[3]; /* 0xEC00_8011 -> 0xEC00_8013 - RESERVED
      */
00050     vubyte isr;       /* 0xEC00_8014 -> 0xEC00_8014 - Interrupt Status Register
      */
00051     vubyte pack05[3]; /* 0xEC00_8015 -> 0xEC00_8017 - RESERVED
      */
00052     vubyte ier;       /* 0xEC00_8018 -> 0xEC00_8018 - Interrupt Enable Register
      */
00053     vubyte pack06[3]; /* 0xEC00_8019 -> 0xEC00_801B - RESERVED
      */
00054 } owstruct;
00055
00056 /*
00057  * I2C MODULE 2-5 (i2c25[4] = 0xEC01_0000 -> 0xEC01_FFFF)
00058  */
00059 typedef struct
00060 {
00061     vubyte i2adr;         /*      0x0000 ->      0x0000 - I2C Address Register
      */
00062     vubyte pack00[3];     /*      0x0001 ->      0x0003 - RESERVED
      */
00063     vubyte i2fdr;         /*      0x0004 ->      0x0004 - I2C Frequency Divider Register
      */
00064     vubyte pack01[3];     /*      0x0005 ->      0x0007 - RESERVED
      */
00065     vubyte i2cr;          /*      0x0008 ->      0x0008 - I2C Control Register
      */
00066     vubyte pack02[3];     /*      0x0009 ->      0x000B - RESERVED
      */
00067     vubyte i2sr;          /*      0x000C ->      0x000C - I2C Status Register
      */
00068     vubyte pack03[3];     /*      0x000D ->      0x000F - RESERVED
      */
00069     vubyte i2dr;          /*      0x0010 ->      0x0010 - I2C Data I/O Register
      */
00070     vubyte pack04[16367]; /*      0x0011 ->      0x3FFF - RESERVED
      */
00071 } i2cstruct;
00072 //} i2c25struct;
00073
00074 // These typedefs are to prevent issues with code that used references to the old
00075 // i2c#struct structures, prior to unifying the type definitions.
00076 typedef i2cstruct i2c25struct;
00077 typedef i2cstruct i2c1struct;
00078 typedef i2cstruct i2c0struct;
00079
00080 /*
00081  * DMA SERIAL PERIPHERAL INTERFACE 2
00082  */
00083 typedef struct
00084 {
00085     vudword mcr;      /* 0xEC03_8000 -> 0xEC03_8003 - Module Configuration Register
      */
00086     vubyte pack00[4]; /* 0xEC03_8004 -> 0xEC03_8007 - RESERVED
      */
00087     vudword tcr;      /* 0xEC03_8008 -> 0xEC03_800B - Transfer Count Register
      */
00088     vudword ctar[8];  /* 0xEC03_800C -> 0xEC03_802B - Clock and Transfer Attributes Register 0-7
      */
00089     vudword sr;       /* 0xEC03_802C -> 0xEC03_802F - Status Register
```

```
      */
00090     vudword rser;     /* 0xEC03_8030 -> 0xEC03_8033 - DMA/Interrupt Request Select and Enable Register
      */
00091     vudword pushr;    /* 0xEC03_8034 -> 0xEC03_8037 - Push Tx FIFO Register
      */
00092     vudword popr;     /* 0xEC03_8038 -> 0xEC03_803B - Pop Rx FIFO Register
      */
00093     vudword txfr[16]; /* 0xEC03_803C -> 0xEC03_807B - Transmit FIFO Register 0-15
      */
00094     vudword rxfr[16]; /* 0xEC03_807C -> 0xEC03_80BB - Receive FIFO Register 0-15
      */
00095 } dspistruct;
00096 //} dspi2struct;
00097
00098 // These typedefs are to prevent issues with code that used references to the old
00099 // dspi#struct structures, prior to unifying the type definitions.
00100 typedef dspistruct dspi0struct;
00101 typedef dspistruct dspi1struct;
00102 typedef dspistruct dspi2struct;
00103 typedef dspistruct dspi3struct;
00104
00105 /*
00106  * DMA SERIAL PERIPHERAL INTERFACE 3
00107  */
00108 // typedef struct {
00109 //    vudword mcr;             /* 0xEC03_C000 -> 0xEC03_C003 - Module Configuration Register
      */
00110 //    vubyte  pack00[4];       /* 0xEC03_C004 -> 0xEC03_C007 - RESERVED
      */
00111 //    vudword tcr;             /* 0xEC03_C008 -> 0xEC03_C00B - Transfer Count Register
      */
00112 //    vudword ctar[8];         /* 0xEC03_C00C -> 0xEC03_C02B - Clock and Transfer Attributes Register
      0-7              */
00113 //    vudword sr;              /* 0xEC03_C02C -> 0xEC03_C02F - Status Register
      */
00114 //    vudword rser;            /* 0xEC03_C030 -> 0xEC03_C033 - DMA/Interrupt Request Select and Enable
      Register        */
00115 //    vudword pushr;           /* 0xEC03_C034 -> 0xEC03_C037 - Push Tx FIFO Register
      */
00116 //    vudword popr;            /* 0xEC03_C038 -> 0xEC03_C03B - Pop Rx FIFO Register
      */
00117 //    vudword txfr[16];        /* 0xEC03_C03C -> 0xEC03_C07B - Transmit FIFO Register 0-15
      */
00118 //    vudword rxfr[16];        /* 0xEC03_C07C -> 0xEC03_C0BB - Receive FIFO Register 0-15
      */
00119 // } dspi3struct;
00120
00121 /*
00122  * UART MODULE 4-9 (uarts[6] = 0xEC06_0000 -> 0xEC07_7FFF)
00123  * Same as uartstruct.
00124  */
00125
00126 /*
00127  * MOTOR CONTROL PULSE-WIDTH MODULATOR SUBMODULE 0-3 (sm[4] = 0xEC08_8000 -> 0xEC08_813F)
00128  */
00129 typedef struct
00130 {
00131     vuword cnt;       /*        0x00 ->        0x01 - Counter Register
      */
00132     vuword init;      /*        0x02 ->        0x03 - Initial Count Register
      */
00133     vuword cr2;       /*        0x04 ->        0x05 - Control Register 2
      */
00134     vuword cr1;       /*        0x06 ->        0x07 - Control Register 1
      */
00135     vuword val[6];    /*        0x08 ->        0x13 - Value Register 0-5
      */
00136     vubyte pack00[4]; /*        0x14 ->        0x17 - RESERVED
      */
00137     vuword ocr;       /*        0x18 ->        0x19 - Output Control Register
      */
00138     vuword sr;        /*        0x1A ->        0x1B - Status Register
      */
00139     vuword ier;       /*        0x1C ->        0x1D - Interrupt Enable Register
      */
00140     vuword dmaen;     /*        0x1E ->        0x1F - DMA Enable Register
      */
00141     vuword otcr;      /*        0x20 ->        0x21 - Output Trigger Control Register
      */
00142     vuword dismap;    /*        0x22 ->        0x23 - Fault Disable Mapping Register
      */
00143     vuword dtcnt0;    /*        0x24 ->        0x25 - Deadtime Count Register 0
      */
00144     vuword dtcnt1;    /*        0x26 ->        0x27 - Deadtime Count Register 1
      */
00145     vuword ccra;      /*        0x28 ->        0x29 - Capture Control Register A
      */
```

```
00146     vuword ccmpa;     /*        0x2A ->        0x2B - Capture Compare Register A
      */
00147     vuword ccrb;      /*        0x2C ->        0x2D - Capture Control Register B
      */
00148     vuword ccmpb;     /*        0x2E ->        0x2F - Capture Compare Register B
      */
00149     vuword ccrx;      /*        0x30 ->        0x31 - Capture Control Register X
      */
00150     vuword ccmpx;     /*        0x32 ->        0x33 - Capture Compare Register X
      */
00151     vuword cval0;     /*        0x34 ->        0x35 - Capture Value 0 Register
      */
00152     vuword ccyc0;     /*        0x36 ->        0x37 - Capture Value 0 Cycle Register
      */
00153     vuword cval1;     /*        0x38 ->        0x39 - Capture Value 1 Register
      */
00154     vuword ccyc1;     /*        0x3A ->        0x3B - Capture Value 1 Cycle Register
      */
00155     vuword cval2;     /*        0x3C ->        0x3D - Capture Value 2 Register
      */
00156     vuword ccyc2;     /*        0x3E ->        0x3F - Capture Value 2 Cycle Register
      */
00157     vuword cval3;     /*        0x40 ->        0x41 - Capture Value 3 Register
      */
00158     vuword ccyc3;     /*        0x42 ->        0x43 - Capture Value 3 Cycle Register
      */
00159     vuword cval4;     /*        0x44 ->        0x45 - Capture Value 4 Register
      */
00160     vuword ccyc4;     /*        0x46 ->        0x47 - Capture Value 4 Cycle Register
      */
00161     vuword cval5;     /*        0x48 ->        0x49 - Capture Value 5 Register
      */
00162     vuword ccyc5;     /*        0x4A ->        0x4B - Capture Value 5 Cycle Register
      */
00163     vubyte pack01[4]; /*        0x4C ->        0x4F - RESERVED
      */
00164 } mcpwm_smstruct;
00165
00166 /*
00167  * MOTOR CONTROL PULSE-WIDTH MODULATOR
00168  */
00169 typedef struct
00170 {
00171     mcpwm_smstruct sm[4]; /* 0xEC08_8000 -> 0xEC08_813F - Submodule 0-3
      */
00172     vuword outen;         /* 0xEC08_8140 -> 0xEC08_8141 - Output Enable Register
      */
00173     vuword mask;          /* 0xEC08_8142 -> 0xEC08_8143 - Output Mask Register
      */
00174     vuword swcout;        /* 0xEC08_8144 -> 0xEC08_8145 - Software Controlled Output Register
      */
00175     vuword dtss;          /* 0xEC08_8146 -> 0xEC08_8147 - Deadtime Source Select Register
      */
00176     vuword mcr;           /* 0xEC08_8148 -> 0xEC08_8149 - Master Control Register
      */
00177     vubyte pack00[2];     /* 0xEC08_814A -> 0xEC08_814B - RESERVED
      */
00178     vuword fcr;           /* 0xEC08_814C -> 0xEC08_814D - Fault Control Register
      */
00179     vuword fsr;           /* 0xEC08_814E -> 0xEC08_814F - Fault Status Register
      */
00180     vuword ffilt;         /* 0xEC08_8150 -> 0xEC08_8151 - Fault Filter Register
      */
00181     vubyte pack01[2];     /* 0xEC08_8152 -> 0xEC08_8153 - RESERVED
      */
00182 } mcpwmstruct;
00183
00184 /*
00185  * RESET CONTROLLER
00186  */
00187 typedef struct
00188 {
00189     vubyte rcr;       /* 0xEC09_0000 -> 0xEC09_0000 - Reset Control Register
      */
00190     vubyte rsr;       /* 0xEC09_0001 -> 0xEC09_0001 - Reset Status Register
      */
00191     vubyte pack00[2]; /* 0xEC09_0002 -> 0xEC09_0003 - RESERVED
      */
00192 } resetstruct;
00193
00194 #define CPUID_MCF_54410 0x09F
00195 #define CPUID_MCF_54415 0x0A0
00196 #define CPUID_MCF_54416 0x0A1
00197 #define CPUID_MCF_54417 0x0A2
00198 #define CPUID_MCF_54418 0x0A3
00199
00200 /*
```

```
00201  * CHIP CONFIGURATION MODULE
00202  */
00203 typedef struct
00204 {
00205     vuword ccr;       /* 0xEC09_0004 -> 0xEC09_0005 - Chip Configuration Register
      */
00206     vubyte pack00;    /* 0xEC09_0006 -> 0xEC09_0006 - RESERVED
      */
00207     vubyte lpcr;      /* 0xEC09_0007 -> 0xEC09_0007 - Low-Power Control Register
      */
00208     vuword rcon;      /* 0xEC09_0008 -> 0xEC09_0009 - Reset Configuration Register
      */
00209     vuword cir;       /* 0xEC09_000A -> 0xEC09_000B - Chip Identification Register
      */
00210     vubyte pack01[2]; /* 0xEC09_000C -> 0xEC09_000D - RESERVED
      */
00211     vuword misccr;    /* 0xEC09_000E -> 0xEC09_000F - Miscellaneous Control Register
      */
00212     vuword cdrh;      /* 0xEC09_0010 -> 0xEC09_0011 - Clock Divider Register High
      */
00213     vuword cdrl;      /* 0xEC09_0012 -> 0xEC09_0013 - Clock Divider Register Low
      */
00214     vuword uocsr;     /* 0xEC09_0014 -> 0xEC09_0015 - USB On-the-Go Controller Status Register
      */
00215     vuword uhcsr;     /* 0xEC09_0016 -> 0xEC09_0017 - USB Host Controller Status Register
      */
00216     vuword misccr3;   /* 0xEC09_0018 -> 0xEC09_0019 - Miscellaneous Control Register 3
      */
00217     vuword misccr2;   /* 0xEC09_001A -> 0xEC09_001B - Miscellaneous Control Register 2
      */
00218     vuword adctsr;    /* 0xEC09_001C -> 0xEC09_001D - ADC Trigger Select Register
      */
00219     vuword dactsr;    /* 0xEC09_001E -> 0xEC09_001F - DAC Trigger Select Register
      */
00220     vuword sbfsr;     /* 0xEC09_0020 -> 0xEC09_0021 - Serial Boot Facility Status Register
      */
00221     vuword sbfcr;     /* 0xEC09_0022 -> 0xEC09_0023 - Serial Boot Facility Control Register
      */
00222     vudword fnacr;    /* 0xEC09_0024 -> 0xEC09_0027 - FlexBus/NAND Flash Arbiter Control Register
      */
00223 } ccmstruct;
00224
00225 /*
00226  * PIN-MULTIPLEXING AND CONTROL (GPIO)
00227  */
00228 typedef struct
00229 {
00230     vubyte podr_a;      /* 0xEC09_4000 -> 0xEC09_4000 - Port Output Data Register A
      */
00231     vubyte podr_b;      /* 0xEC09_4001 -> 0xEC09_4001 - Port Output Data Register B
      */
00232     vubyte podr_c;      /* 0xEC09_4002 -> 0xEC09_4002 - Port Output Data Register C
      */
00233     vubyte podr_d;      /* 0xEC09_4003 -> 0xEC09_4003 - Port Output Data Register D
      */
00234     vubyte podr_e;      /* 0xEC09_4004 -> 0xEC09_4004 - Port Output Data Register E
      */
00235     vubyte podr_f;      /* 0xEC09_4005 -> 0xEC09_4005 - Port Output Data Register F
      */
00236     vubyte podr_g;      /* 0xEC09_4006 -> 0xEC09_4006 - Port Output Data Register G
      */
00237     vubyte podr_h;      /* 0xEC09_4007 -> 0xEC09_4007 - Port Output Data Register H
      */
00238     vubyte podr_i;      /* 0xEC09_4008 -> 0xEC09_4008 - Port Output Data Register I
      */
00239     vubyte podr_j;      /* 0xEC09_4009 -> 0xEC09_4009 - Port Output Data Register J
      */
00240     vubyte podr_k;      /* 0xEC09_400A -> 0xEC09_400A - Port Output Data Register K
      */
00241     vubyte pack00;      /* 0xEC09_400B -> 0xEC09_400B - RESERVED
      */
00242     vubyte pddr_a;      /* 0xEC09_400C -> 0xEC09_400C - Port Data Direction Register A
      */
00243     vubyte pddr_b;      /* 0xEC09_400D -> 0xEC09_400D - Port Data Direction Register B
      */
00244     vubyte pddr_c;      /* 0xEC09_400E -> 0xEC09_400E - Port Data Direction Register C
      */
00245     vubyte pddr_d;      /* 0xEC09_400F -> 0xEC09_400F - Port Data Direction Register D
      */
00246     vubyte pddr_e;      /* 0xEC09_4010 -> 0xEC09_4010 - Port Data Direction Register E
      */
00247     vubyte pddr_f;      /* 0xEC09_4011 -> 0xEC09_4011 - Port Data Direction Register F
      */
00248     vubyte pddr_g;      /* 0xEC09_4012 -> 0xEC09_4012 - Port Data Direction Register G
      */
00249     vubyte pddr_h;      /* 0xEC09_4013 -> 0xEC09_4013 - Port Data Direction Register H
      */
```

```
00250     vubyte pddr_i;      /* 0xEC09_4014 -> 0xEC09_4014 - Port Data Direction Register I
      */
00251     vubyte pddr_j;      /* 0xEC09_4015 -> 0xEC09_4015 - Port Data Direction Register J
      */
00252     vubyte pddr_k;      /* 0xEC09_4016 -> 0xEC09_4016 - Port Data Direction Register K
      */
00253     vubyte pack01;      /* 0xEC09_4017 -> 0xEC09_4017 - RESERVED
      */
00254     vubyte ppdsdr_a;    /* 0xEC09_4018 -> 0xEC09_4018 - Port Pin Data/Set Data Register A
      */
00255     vubyte ppdsdr_b;    /* 0xEC09_4019 -> 0xEC09_4019 - Port Pin Data/Set Data Register B
      */
00256     vubyte ppdsdr_c;    /* 0xEC09_401A -> 0xEC09_401A - Port Pin Data/Set Data Register C
      */
00257     vubyte ppdsdr_d;    /* 0xEC09_401B -> 0xEC09_401B - Port Pin Data/Set Data Register D
      */
00258     vubyte ppdsdr_e;    /* 0xEC09_401C -> 0xEC09_401C - Port Pin Data/Set Data Register E
      */
00259     vubyte ppdsdr_f;    /* 0xEC09_401D -> 0xEC09_401D - Port Pin Data/Set Data Register F
      */
00260     vubyte ppdsdr_g;    /* 0xEC09_401E -> 0xEC09_401E - Port Pin Data/Set Data Register G
      */
00261     vubyte ppdsdr_h;    /* 0xEC09_401F -> 0xEC09_401F - Port Pin Data/Set Data Register H
      */
00262     vubyte ppdsdr_i;    /* 0xEC09_4020 -> 0xEC09_4020 - Port Pin Data/Set Data Register I
      */
00263     vubyte ppdsdr_j;    /* 0xEC09_4021 -> 0xEC09_4021 - Port Pin Data/Set Data Register J
      */
00264     vubyte ppdsdr_k;    /* 0xEC09_4022 -> 0xEC09_4022 - Port Pin Data/Set Data Register K
      */
00265     vubyte pack02;      /* 0xEC09_4023 -> 0xEC09_4023 - RESERVED
      */
00266     vubyte pclrr_a;     /* 0xEC09_4024 -> 0xEC09_4024 - Port Clear Output Data Register A
      */
00267     vubyte pclrr_b;     /* 0xEC09_4025 -> 0xEC09_4025 - Port Clear Output Data Register B
      */
00268     vubyte pclrr_c;     /* 0xEC09_4026 -> 0xEC09_4026 - Port Clear Output Data Register C
      */
00269     vubyte pclrr_d;     /* 0xEC09_4027 -> 0xEC09_4027 - Port Clear Output Data Register D
      */
00270     vubyte pclrr_e;     /* 0xEC09_4028 -> 0xEC09_4028 - Port Clear Output Data Register E
      */
00271     vubyte pclrr_f;     /* 0xEC09_4029 -> 0xEC09_4029 - Port Clear Output Data Register F
      */
00272     vubyte pclrr_g;     /* 0xEC09_402A -> 0xEC09_402A - Port Clear Output Data Register G
      */
00273     vubyte pclrr_h;     /* 0xEC09_402B -> 0xEC09_402B - Port Clear Output Data Register H
      */
00274     vubyte pclrr_i;     /* 0xEC09_402C -> 0xEC09_402C - Port Clear Output Data Register I
      */
00275     vubyte pclrr_j;     /* 0xEC09_402D -> 0xEC09_402D - Port Clear Output Data Register J
      */
00276     vubyte pclrr_k;     /* 0xEC09_402E -> 0xEC09_402E - Port Clear Output Data Register K
      */
00277     vubyte pack03;      /* 0xEC09_402F -> 0xEC09_402F - RESERVED
      */
00278     vuword pcr_a;       /* 0xEC09_4030 -> 0xEC09_4031 - Pull Control Register A
      */
00279     vuword pcr_b;       /* 0xEC09_4032 -> 0xEC09_4033 - Pull Control Register B
      */
00280     vuword pcr_c;       /* 0xEC09_4034 -> 0xEC09_4035 - Pull Control Register C
      */
00281     vuword pcr_d;       /* 0xEC09_4036 -> 0xEC09_4037 - Pull Control Register D
      */
00282     vuword pcr_e;       /* 0xEC09_4038 -> 0xEC09_4039 - Pull Control Register E
      */
00283     vuword pcr_f;       /* 0xEC09_403A -> 0xEC09_403B - Pull Control Register F
      */
00284     vuword pcr_g;       /* 0xEC09_403C -> 0xEC09_403D - Pull Control Register G
      */
00285     vuword pcr_h;       /* 0xEC09_403E -> 0xEC09_403F - Pull Control Register H
      */
00286     vuword pcr_i;       /* 0xEC09_4040 -> 0xEC09_4041 - Pull Control Register I
      */
00287     vuword pcr_j;       /* 0xEC09_4042 -> 0xEC09_4043 - Pull Control Register J
      */
00288     vuword pcr_k;       /* 0xEC09_4044 -> 0xEC09_4045 - Pull Control Register K
      */
00289     vubyte pack04[2];   /* 0xEC09_4046 -> 0xEC09_4047 - RESERVED
      */
00290     vubyte par_fbctl;   /* 0xEC09_4048 -> 0xEC09_4048 - Pin Assignment Register - FlexBus Control
      */
00291     vubyte par_be;      /* 0xEC09_4049 -> 0xEC09_4049 - Pin Assignment Register - Byte Enable
      */
00292     vubyte par_cs;      /* 0xEC09_404A -> 0xEC09_404A - Pin Assignment Register - Chip Select
      */
00293     vubyte par_cani2c;  /* 0xEC09_404B -> 0xEC09_404B - Pin Assignment Register - CAN1 and I2C0
```

```
    */
00294     vubyte par_irq0h;   /* 0xEC09_404C -> 0xEC09_404C - Pin Assignment Register - Edge Port 0 High
    */
00295     vubyte par_irq0l;   /* 0xEC09_404D -> 0xEC09_404D - Pin Assignment Register - Edge Port 0 Low
    */
00296     vubyte par_dspiowh; /* 0xEC09_404E -> 0xEC09_404E - Pin Assignment Register - DSPI0 and One-Wire
    High       */
00297     vubyte par_dspiowl; /* 0xEC09_404F -> 0xEC09_404F - Pin Assignment Register - DSPI0 and One-Wire
    Low        */
00298     vubyte par_timer;   /* 0xEC09_4050 -> 0xEC09_4050 - Pin Assignment Register - Timer
    */
00299     vubyte par_uart2;   /* 0xEC09_4051 -> 0xEC09_4051 - Pin Assignment Register - UART 2
    */
00300     vubyte par_uart1;   /* 0xEC09_4052 -> 0xEC09_4052 - Pin Assignment Register - UART 1
    */
00301     vubyte par_uart0;   /* 0xEC09_4053 -> 0xEC09_4053 - Pin Assignment Register - UART 0
    */
00302     vubyte par_sdhch;   /* 0xEC09_4054 -> 0xEC09_4054 - Pin Assignment Register - eSDHC High
    */
00303     vubyte par_sdhcl;   /* 0xEC09_4055 -> 0xEC09_4055 - Pin Assignment Register - eSDHC Low
    */
00304     vubyte par_simp0h;  /* 0xEC09_4056 -> 0xEC09_4056 - Pin Assignment Register - SIM Port 0 High
    */
00305     vubyte par_simp0l;  /* 0xEC09_4057 -> 0xEC09_4057 - Pin Assignment Register - SIM Port 0 Low
    */
00306     vubyte par_ssi0h;   /* 0xEC09_4058 -> 0xEC09_4058 - Pin Assignment Register - SSI0 High
    */
00307     vubyte par_ssi0l;   /* 0xEC09_4059 -> 0xEC09_4059 - Pin Assignment Register - SSI0 Low
    */
00308     vubyte par_debugh1; /* 0xEC09_405A -> 0xEC09_405A - Pin Assignment Register - Debug High 1
    */
00309     vubyte par_debugh0; /* 0xEC09_405B -> 0xEC09_405B - Pin Assignment Register - Debug High 0
    */
00310     vubyte par_debugl;  /* 0xEC09_405C -> 0xEC09_405C - Pin Assignment Register - Debug Low
    */
00311     vubyte pack05;      /* 0xEC09_405D -> 0xEC09_405D - RESERVED
    */
00312     vubyte par_fec;     /* 0xEC09_405E -> 0xEC09_405E - Pin Assignment Register - Fast Ethernet
    Channel          */
00313     vubyte pack06;      /* 0xEC09_405F -> 0xEC09_405F - RESERVED
    */
00314     vubyte mscr_sdramc; /* 0xEC09_4060 -> 0xEC09_4060 - Mode Select Control Register - SDRAM
    Controller          */
00315     vubyte pack07[3];   /* 0xEC09_4061 -> 0xEC09_4063 - RESERVED
    */
00316     vubyte srcr_fb1;    /* 0xEC09_4064 -> 0xEC09_4064 - Slew Rate Control Register - FlexBus 1
    */
00317     vubyte srcr_fb2;    /* 0xEC09_4065 -> 0xEC09_4065 - Slew Rate Control Register - FlexBus 2
    */
00318     vubyte srcr_fb3;    /* 0xEC09_4066 -> 0xEC09_4066 - Slew Rate Control Register - FlexBus 3
    */
00319     vubyte srcr_fb4;    /* 0xEC09_4067 -> 0xEC09_4067 - Slew Rate Control Register - FlexBus 4
    */
00320     vubyte srcr_dspiow; /* 0xEC09_4068 -> 0xEC09_4068 - Slew Rate Control Register - DSPI0 and
    One-Wire          */
00321     vubyte srcr_cani2c; /* 0xEC09_4069 -> 0xEC09_4069 - Slew Rate Control Register - CAN1 and I2C0
    */
00322     vubyte srcr_irq0;   /* 0xEC09_406A -> 0xEC09_406A - Slew Rate Control Register - Edge Port 0
    */
00323     vubyte srcr_timer;  /* 0xEC09_406B -> 0xEC09_406B - Slew Rate Control Register - Timer
    */
00324     vubyte srcr_uart;   /* 0xEC09_406C -> 0xEC09_406C - Slew Rate Control Register - UART
    */
00325     vubyte srcr_fec;    /* 0xEC09_406D -> 0xEC09_406D - Slew Rate Control Register - Fast Ethernet
    Channel       */
00326     vubyte srcr_sdhc;   /* 0xEC09_406E -> 0xEC09_406E - Slew Rate Control Register - eSDHC
    */
00327     vubyte srcr_simp0;  /* 0xEC09_406F -> 0xEC09_406F - Slew Rate Control Register - SIM Port 0
    */
00328     vubyte srcr_ssi0;   /* 0xEC09_4070 -> 0xEC09_4070 - Slew Rate Control Register - SSI0
    */
00329     vubyte pack08[3];   /* 0xEC09_4071 -> 0xEC09_4073 - RESERVED
    */
00330     vuword urts_pol;    /* 0xEC09_4074 -> 0xEC09_4075 - Miscellaneous UART Register - RTS Polarity
    Control       */
00331     vuword ucts_pol;    /* 0xEC09_4076 -> 0xEC09_4077 - Miscellaneous UART Register - CTS Polarity
    Control       */
00332     vuword utxd_wom;    /* 0xEC09_4078 -> 0xEC09_4079 - Miscellaneous UART Register - Tx Wired-Or Mode
    Control   */
00333     vubyte pack09[2];   /* 0xEC09_407A -> 0xEC09_407B - RESERVED
    */
00334     vudword urxd_wom;   /* 0xEC09_407C -> 0xEC09_407F - Miscellaneous UART Register - Rx Wired-Or Mode
    Control   */
00335     vudword hcr1;       /* 0xEC09_4080 -> 0xEC09_4083 - Hysteresis Control Register 1
    */
00336     vudword hcr0;       /* 0xEC09_4084 -> 0xEC09_4087 - Hysteresis Control Register 0
    */
```

```
00337 } gpiostruct;
00338
00339 /*
00340  * CROSSBAR SWITCH (XBS)
00341  */
00342 typedef struct
00343 {
00344     vudword prs0;       /* 0xFC00_4000 -> 0xFC00_4003 - Priority Register Slave 0
      */
00345     vubyte pack00[12];  /* 0xFC00_4004 -> 0xFC00_400F - RESERVED
      */
00346     vudword crs0;       /* 0xFC00_4010 -> 0xFC00_4013 - Control Register Slave 0
      */
00347     vubyte pack01[236]; /* 0xFC00_4014 -> 0xFC00_40FF - RESERVED
      */
00348     vudword prs1;       /* 0xFC00_4100 -> 0xFC00_4103 - Priority Register Slave 1
      */
00349     vubyte pack02[12];  /* 0xFC00_4104 -> 0xFC00_410F - RESERVED
      */
00350     vudword crs1;       /* 0xFC00_4110 -> 0xFC00_4113 - Control Register Slave 1
      */
00351     vubyte pack03[236]; /* 0xFC00_4114 -> 0xFC00_41FF - RESERVED
      */
00352     vudword prs2;       /* 0xFC00_4200 -> 0xFC00_4203 - Priority Register Slave 2
      */
00353     vubyte pack04[12];  /* 0xFC00_4204 -> 0xFC00_420F - RESERVED
      */
00354     vudword crs2;       /* 0xFC00_4210 -> 0xFC00_4213 - Control Register Slave 2
      */
00355     vubyte pack05[492]; /* 0xFC00_4214 -> 0xFC00_43FF - RESERVED
      */
00356     vudword prs4;       /* 0xFC00_4400 -> 0xFC00_4403 - Priority Register Slave 4
      */
00357     vubyte pack06[12];  /* 0xFC00_4404 -> 0xFC00_440F - RESERVED
      */
00358     vudword crs4;       /* 0xFC00_4410 -> 0xFC00_4413 - Control Register Slave 4
      */
00359     vubyte pack07[492]; /* 0xFC00_4414 -> 0xFC00_45FF - RESERVED
      */
00360     vudword prs6;       /* 0xFC00_4600 -> 0xFC00_4603 - Priority Register Slave 6
      */
00361     vubyte pack08[12];  /* 0xFC00_4604 -> 0xFC00_460F - RESERVED
      */
00362     vudword crs6;       /* 0xFC00_4610 -> 0xFC00_4613 - Control Register Slave 6
      */
00363     vubyte pack09[236]; /* 0xFC00_4614 -> 0xFC00_46FF - RESERVED
      */
00364     vudword prs7;       /* 0xFC00_4700 -> 0xFC00_4703 - Priority Register Slave 7
      */
00365     vubyte pack10[12];  /* 0xFC00_4704 -> 0xFC00_470F - RESERVED
      */
00366     vudword crs7;       /* 0xFC00_4710 -> 0xFC00_4713 - Control Register Slave 7
      */
00367 } xbsstruct;
00368
00369 /*
00370  * CHIP SELECT 0-5 (cs[6] = 0xFC00_8000 -> 0xFC00_8047)
00371  */
00372 typedef struct
00373 {
00374     vudword csar; /*      0x0000 ->      0x0003 - Chip Select Address Register
      */
00375     vudword csmr; /*      0x0004 ->      0x0007 - Chip Select Mask Register
      */
00376     vudword cscr; /*      0x0008 ->      0x000B - Chip Select Control Register
      */
00377 } csstruct;
00378
00379 /*
00380  * CONTROLLER AREA NETWORK MESSAGE BUFFER
00381  */
00382 typedef struct
00383 {
00384     vuword bStatus;     /*        0x00 ->        0x01 - Code[11:8]; SRR[6]; IDE[5]; RTR[4]; Length[3:0]
      */
00385     vuword bTimeStamp; /*        0x02 ->        0x03 - Time Stamp
      */
00386     vudword id;         /*        0x04 ->        0x07 - Standard ID[28:18]; Extended ID[17:0]
      */
00387     vubyte data[8];     /*        0x08 ->        0x0F - Data Bytes 0-7
      */
00388 } can_mbstruct;
00389
00390 /*
00391  * CONTROLLER AREA NETWORK (FLEXCAN) 0-1 (can[2] = 0xFC02_0000 -> 0xFC02_7FFF)
00392  */
00393 typedef struct
```

```
00394 {
00395     vudword canmcr;       /*      0x0000 ->      0x0003 - Module Configuration Register
      */
00396     vudword canctrl;      /*      0x0004 ->      0x0007 - Control Register
      */
00397     vudword timer;        /*      0x0008 ->      0x000B - Free Running Timer
      */
00398     vubyte pack00[4];     /*      0x000C ->      0x000F - RESERVED
      */
00399     vudword rxgmask;      /*      0x0010 ->      0x0013 - Rx Global Mask
      */
00400     vudword rx14mask;     /*      0x0014 ->      0x0017 - Rx Buffer 14 Mask
      */
00401     vudword rx15mask;     /*      0x0018 ->      0x001B - Rx Buffer 15 Mask
      */
00402     vudword errcnt;       /*      0x001C ->      0x001F - Error Counter Register
      */
00403     vudword errstat;      /*      0x0020 ->      0x0023 - Error and Status Register
      */
00404     vubyte pack01[4];     /*      0x0024 ->      0x0027 - RESERVED
      */
00405     vudword imask;        /*      0x0028 ->      0x002B - Interrupt Mask Register
      */
00406     vubyte pack02[4];     /*      0x002C ->      0x002F - RESERVED
      */
00407     vudword iflag;        /*      0x0030 ->      0x0033 - Interrupt Flag Register
      */
00408     vubyte pack03[76];    /*      0x0034 ->      0x007F - RESERVED
      */
00409     can_mbstruct mbs[16]; /*      0x0080 ->      0x017F - Message Buffers 0-15
      */
00410     vubyte pack04[1792];  /*      0x0180 ->      0x087F - RESERVED
      */
00411     vudword rximr[16];    /*      0x0880 ->      0x08BF - Rx Individual Mask Registers 0-15
      */
00412     vubyte pack05[14144]; /*      0x08C0 ->      0x3FFF - RESERVED
      */
00413 } canstruct;
00414
00415 /*
00416  * I2C MODULE 1
00417  */
00418 // typedef struct {
00419 //    vubyte  i2adr;           /* 0xFC03_8000 -> 0xFC03_8000 - I2C Address Register
      */
00420 //    vubyte  pack00[3];       /* 0xFC03_8001 -> 0xFC03_8003 - RESERVED
      */
00421 //    vubyte  i2fdr;           /* 0xFC03_8004 -> 0xFC03_8004 - I2C Frequency Divider Register
      */
00422 //    vubyte  pack01[3];       /* 0xFC03_8005 -> 0xFC03_8007 - RESERVED
      */
00423 //    vubyte  i2cr;            /* 0xFC03_8008 -> 0xFC03_8008 - I2C Control Register
      */
00424 //    vubyte  pack02[3];       /* 0xFC03_8009 -> 0xFC03_800B - RESERVED
      */
00425 //    vubyte  i2sr;            /* 0xFC03_800C -> 0xFC03_800C - I2C Status Register
      */
00426 //    vubyte  pack03[3];       /* 0xFC03_800D -> 0xFC03_800F - RESERVED
      */
00427 //    vubyte  i2dr;            /* 0xFC03_8010 -> 0xFC03_8010 - I2C Data I/O Register
      */
00428 //    vubyte  pack04[3];       /* 0xFC03_8011 -> 0xFC03_8013 - RESERVED
      */
00429 // } i2c1struct;
00430
00431 /*
00432  * DMA SERIAL PERIPHERAL INTERFACE 1
00433  */
00434 // typedef struct {
00435 //    vudword mcr;             /* 0xFC03_C000 -> 0xFC03_C003 - Module Configuration Register
      */
00436 //    vubyte  pack00[4];       /* 0xFC03_C004 -> 0xFC03_C007 - RESERVED
      */
00437 //    vudword tcr;             /* 0xFC03_C008 -> 0xFC03_C00B - Transfer Count Register
      */
00438 //    vudword ctar[8];         /* 0xFC03_C00C -> 0xFC03_C02B - Clock and Transfer Attributes Register
      0-7             */
00439 //    vudword sr;              /* 0xFC03_C02C -> 0xFC03_C02F - Status Register
      */
00440 //    vudword rser;            /* 0xFC03_C030 -> 0xFC03_C033 - DMA/Interrupt Request Select and Enable
      Register        */
00441 //    vudword pushr;           /* 0xFC03_C034 -> 0xFC03_C037 - Push Tx FIFO Register
      */
00442 //    vudword popr;            /* 0xFC03_C038 -> 0xFC03_C03B - Pop Rx FIFO Register
      */
00443 //    vudword txfr[16];        /* 0xFC03_C03C -> 0xFC03_C07B - Transmit FIFO Register 0-15
      */
```

```
00444 //    vudword rxfr[16];        /* 0xFC03_C07C -> 0xFC03_C0BB - Receive FIFO Register 0-15
      */
00445 // } dspi1struct;
00446
00447 /*
00448  * SYSTEM CONTROL MODULE AND POWER MANAGEMENT
00449  */
00450 typedef struct
00451 {
00452     vubyte pack00[3];  /* 0xFC04_0010 -> 0xFC04_0012 - RESERVED
      */
00453     vubyte wcr;        /* 0xFC04_0013 -> 0xFC04_0013 - Wakeup Control Register
      */
00454     vubyte pack01[2];  /* 0xFC04_0014 -> 0xFC04_0015 - RESERVED
      */
00455     vuword cwcr;       /* 0xFC04_0016 -> 0xFC04_0017 - Core Watchdog Control Register
      */
00456     vubyte pack02[3];  /* 0xFC04_0018 -> 0xFC04_001A - RESERVED
      */
00457     vubyte cwsr;       /* 0xFC04_001B -> 0xFC04_001B - Core Watchdog Service Register
      */
00458     vubyte pack03[3];  /* 0xFC04_001C -> 0xFC04_001E - RESERVED
      */
00459     vubyte scmisr;     /* 0xFC04_001F -> 0xFC04_001F - SCM Interrupt Status Register
      */
00460     vubyte pack04[4];  /* 0xFC04_0020 -> 0xFC04_0023 - RESERVED
      */
00461     vudword bcr;       /* 0xFC04_0024 -> 0xFC04_0027 - Burst Configuration Register
      */
00462     vubyte pack05[4];  /* 0xFC04_0028 -> 0xFC04_002B - RESERVED
      */
00463     vubyte ppmsr0;     /* 0xFC04_002C -> 0xFC04_002C - Peripheral Power Management Set Register 0
      */
00464     vubyte ppmcr0;     /* 0xFC04_002D -> 0xFC04_002D - Peripheral Power Management Clear Register 0
      */
00465     vubyte ppmsr1;     /* 0xFC04_002E -> 0xFC04_002E - Peripheral Power Management Set Register 1
      */
00466     vubyte ppmcr1;     /* 0xFC04_002F -> 0xFC04_002F - Peripheral Power Management Clear Register 1
      */
00467     vudword ppmhr0;    /* 0xFC04_0030 -> 0xFC04_0033 - Peripheral Power Management High Register 0
      */
00468     vudword ppmlr0;    /* 0xFC04_0034 -> 0xFC04_0037 - Peripheral Power Management Low Register 0
      */
00469     vudword ppmhr1;    /* 0xFC04_0038 -> 0xFC04_003B - Peripheral Power Management High Register 1
      */
00470     vudword ppmlr1;    /* 0xFC04_003C -> 0xFC04_003F - Peripheral Power Management Low Register 1
      */
00471     vubyte pack06[48]; /* 0xFC04_0040 -> 0xFC04_006F - RESERVED
      */
00472     vudword cfadr;     /* 0xFC04_0070 -> 0xFC04_0073 - Core Fault Address Register
      */
00473     vubyte pack07;     /* 0xFC04_0074 -> 0xFC04_0074 - RESERVED
      */
00474     vubyte cfier;      /* 0xFC04_0075 -> 0xFC04_0075 - Core Fault Interrupt Enable Register
      */
00475     vubyte cfloc;      /* 0xFC04_0076 -> 0xFC04_0076 - Core Fault Location Register
      */
00476     vubyte cfatr;      /* 0xFC04_0077 -> 0xFC04_0077 - Core Fault Attributes Register
      */
00477     vubyte pack08[4];  /* 0xFC04_0078 -> 0xFC04_007B - RESERVED
      */
00478     vudword cfdtr;     /* 0xFC04_007C -> 0xFC04_007F - Core Fault Data Register
      */
00479 } scmstruct;
00480
00481 /*
00482  * EDMA TRANSFER CONTROL DESCRIPTOR (tcd[16] = 0xFC04_5000 -> 0xFC04_51FF)
00483  */
00484 typedef struct
00485 {
00486     vudword saddr;     /*      0x0000 ->      0x0003 - Source Address
      */
00487     vuword attr;       /*      0x0004 ->      0x0005 - Transfer Attributes
      */
00488     vuword soff;       /*      0x0006 ->      0x0007 - Signed Source Address Offset
      */
00489     vudword nbytes;    /*      0x0008 ->      0x000B - Signed Minor Loop Offset/Minor Byte Count
      */
00490     vudword slast;     /*      0x000C ->      0x000F - Last Source Address Adjustment
      */
00491     vudword daddr;     /*      0x0010 ->      0x0013 - Destination Address
      */
00492     vuword citer;      /*      0x0014 ->      0x0015 - Current Minor Loop Link/Major Loop Count
      */
00493     vuword doff;       /*      0x0016 ->      0x0017 - Signed Destination Address Offset
      */
00494     vudword dlast_sga; /*      0x0018 ->      0x001B - Last Destination Addr. Adjustment/Scatter
```

```
      Gather Addr.  */
00495     vuword biter;       /*      0x001C ->      0x001D - Beginning Minor Loop Link/Major Loop Count
      */
00496     vuword csr;         /*      0x001E ->      0x001F - Control and Status
      */
00497 } edma_tcdstruct;
00498
00499 /*
00500  * ENHANCED DIRECT MEMORY ACCESS (EDMA) CONTROLLER
00501  */
00502 typedef struct
00503 {
00504     vudword cr;             /* 0xFC04_4000 -> 0xFC04_4003 - eDMA Control Register
      */
00505     vudword es;             /* 0xFC04_4004 -> 0xFC04_4007 - eDMA Error Status Register
      */
00506     vudword erqh;           /* 0xFC04_4008 -> 0xFC04_400B - eDMA Enable Request High Register
      */
00507     vudword erql;           /* 0xFC04_400C -> 0xFC04_400F - eDMA Enable Request Register
      */
00508     vudword eeih;           /* 0xFC04_4010 -> 0xFC04_4013 - eDMA Enable Error Interrupt High Register
      */
00509     vudword eeil;           /* 0xFC04_4014 -> 0xFC04_4017 - eDMA Enable Error Interrupt Low Register
      */
00510     vubyte serq;            /* 0xFC04_4018 -> 0xFC04_4018 - eDMA Set Enable Request
      */
00511     vubyte cerq;            /* 0xFC04_4019 -> 0xFC04_4019 - eDMA Clear Enable Request
      */
00512     vubyte seei;            /* 0xFC04_401A -> 0xFC04_401A - eDMA Set Enable Error Interrupt Register
      */
00513     vubyte ceei;            /* 0xFC04_401B -> 0xFC04_401B - eDMA Clear Enable Error Interrupt Register
      */
00514     vubyte cint;            /* 0xFC04_401C -> 0xFC04_401C - eDMA Clear Interrupt Request Register
      */
00515     vubyte cerr;            /* 0xFC04_401D -> 0xFC04_401D - eDMA Clear Error Register
      */
00516     vubyte ssrt;            /* 0xFC04_401E -> 0xFC04_401E - eDMA Set START Bit Register
      */
00517     vubyte cdne;            /* 0xFC04_401F -> 0xFC04_401F - eDMA Clear DONE Status Bit Register
      */
00518     vudword inth;           /* 0xFC04_4020 -> 0xFC04_4023 - eDMA Interrupt Request High Register
      */
00519     vudword intl;           /* 0xFC04_4024 -> 0xFC04_4027 - eDMA Interrupt Request Low Register
      */
00520     vudword errh;           /* 0xFC04_4028 -> 0xFC04_402B - eDMA Error High Register
      */
00521     vudword errl;           /* 0xFC04_402C -> 0xFC04_402F - eDMA Error Low Register
      */
00522     vudword rsh;            /* 0xFC04_4030 -> 0xFC04_4033 - eDMA Hardware Request Status High
      */
00523     vudword rsl;            /* 0xFC04_4034 -> 0xFC04_4037 - eDMA Hardware Request Status Low
      */
00524     vubyte pack00[200];     /* 0xFC04_4038 -> 0xFC04_40FF - RESERVED
      */
00525     vubyte dchpri[64];      /* 0xFC04_4100 -> 0xFC04_413F - eDMA Channel 0-64 Priority Registers
      */
00526     vubyte pack05[3776];    /* 0xFC04_4140 -> 0xFC04_4FFF - RESERVED
      */
00527     edma_tcdstruct tcd[64]; /* 0xFC04_5000 -> 0xFC04_57FF - Transfer Control Descriptor 0-64
      */
00528 } edmastruct;
00529
00530 /*
00531  * INTERRUPT CONTROLLER 0-2 (intc[3] = 0xFC04_8000 -> 0xFC05_3FFF)
00532  */
00533 typedef struct
00534 {
00535     vudword iprh;         /*      0x0000 ->      0x0003 - Interrupt Pending Register High
      */
00536     vudword iprl;         /*      0x0004 ->      0x0007 - Interrupt Pending Register Low
      */
00537     vudword imrh;         /*      0x0008 ->      0x000B - Interrupt Mask Register High
      */
00538     vudword imrl;         /*      0x000C ->      0x000F - Interrupt Mask Register Low
      */
00539     vudword intfrch;      /*      0x0010 ->      0x0013 - Interrupt Force Register High
      */
00540     vudword intfrcl;      /*      0x0014 ->      0x0017 - Interrupt Force Register Low
      */
00541     vubyte pack00[2];     /*      0x0018 ->      0x0019 - RESERVED
      */
00542     vuword iconfig;       /*      0x001A ->      0x001B - Interrupt Configuration Register
      */
00543     vubyte simr;          /*      0x001C ->      0x001C - Set Interrupt Mask
      */
00544     vubyte cimr;          /*      0x001D ->      0x001D - Clear Interrupt Mask
      */
```

```
00545     vubyte clmask;        /*      0x001E ->      0x001E - Current Level Mask
      */
00546     vubyte slmask;        /*      0x001F ->      0x001F - Saved Level Mask
      */
00547     vubyte pack01[32];    /*      0x0020 ->      0x003F - RESERVED
      */
00548     vubyte icrn[64];      /*      0x0040 ->      0x007F - Interrupt Control Registers (0-63)
      */
00549     vubyte pack02[96];    /*      0x0080 ->      0x00DF - RESERVED
      */
00550     vubyte swackr;        /*      0x00E0 ->      0x00E0 - Software IACK Register
      */
00551     vubyte pack03[3];     /*      0x00E1 ->      0x00E3 - RESERVED
      */
00552     vubyte l1ackr;        /*      0x00E4 ->      0x00E4 - Level 1 IACK Register
      */
00553     vubyte pack04[3];     /*      0x00E5 ->      0x00E7 - RESERVED
      */
00554     vubyte l2ackr;        /*      0x00E8 ->      0x00E8 - Level 2 IACK Register
      */
00555     vubyte pack05[3];     /*      0x00E9 ->      0x00EB - RESERVED
      */
00556     vubyte l3ackr;        /*      0x00EC ->      0x00EC - Level 3 IACK Register
      */
00557     vubyte pack06[3];     /*      0x00ED ->      0x00EF - RESERVED
      */
00558     vubyte l4ackr;        /*      0x00F0 ->      0x00F0 - Level 4 IACK Register
      */
00559     vubyte pack07[3];     /*      0x00F1 ->      0x00F3 - RESERVED
      */
00560     vubyte l5ackr;        /*      0x00F4 ->      0x00F4 - Level 5 IACK Register
      */
00561     vubyte pack08[3];     /*      0x00F5 ->      0x00F7 - RESERVED
      */
00562     vubyte l6ackr;        /*      0x00F8 ->      0x00F8 - Level 6 IACK Register
      */
00563     vubyte pack09[3];     /*      0x00F9 ->      0x00FB - RESERVED
      */
00564     vubyte l7ackr;        /*      0x00FC ->      0x00FC - Level 7 IACK Register
      */
00565     vubyte pack10[16131]; /*      0x00FD ->      0x3FFF - RESERVED
      */
00566 } intcstruct;
00567
00568 /*
00569  * GLOBAL INTERRUPT ACKNOWLEDGE CYCLES
00570  */
00571 typedef struct
00572 {
00573     vubyte gswiack;   /* 0xFC05_40E0 -> 0xFC05_40E0 - Global Software Interrupt Acknowledge
      */
00574     vubyte pack00[3]; /* 0xFC05_40E1 -> 0xFC05_40E3 - RESERVED
      */
00575     vubyte gl1iack;   /* 0xFC05_40E4 -> 0xFC05_40E4 - Global Level 1 Interrupt Acknowledge Register
      */
00576     vubyte pack01[3]; /* 0xFC05_40E5 -> 0xFC05_40E7 - RESERVED
      */
00577     vubyte gl2iack;   /* 0xFC05_40E8 -> 0xFC05_40E8 - Global Level 2 Interrupt Acknowledge Register
      */
00578     vubyte pack02[3]; /* 0xFC05_40E9 -> 0xFC05_40EB - RESERVED
      */
00579     vubyte gl3iack;   /* 0xFC05_40EC -> 0xFC05_40EC - Global Level 3 Interrupt Acknowledge Register
      */
00580     vubyte pack03[3]; /* 0xFC05_40ED -> 0xFC05_40EF - RESERVED
      */
00581     vubyte gl4iack;   /* 0xFC05_40F0 -> 0xFC05_40F0 - Global Level 4 Interrupt Acknowledge Register
      */
00582     vubyte pack04[3]; /* 0xFC05_40F1 -> 0xFC05_40F3 - RESERVED
      */
00583     vubyte gl5iack;   /* 0xFC05_40F4 -> 0xFC05_40F4 - Global Level 5 Interrupt Acknowledge Register
      */
00584     vubyte pack05[3]; /* 0xFC05_40F5 -> 0xFC05_40F7 - RESERVED
      */
00585     vubyte gl6iack;   /* 0xFC05_40F8 -> 0xFC05_40F8 - Global Level 6 Interrupt Acknowledge Register
      */
00586     vubyte pack06[3]; /* 0xFC05_40F9 -> 0xFC05_40FB - RESERVED
      */
00587     vubyte gl7iack;   /* 0xFC05_40FC -> 0xFC05_40FC - Global Level 7 Interrupt Acknowledge Register
      */
00588     vubyte pack07[3]; /* 0xFC05_40FD -> 0xFC05_40FF - RESERVED
      */
00589 } intc_iackstruct;
00590
00591 /*
00592  * I2C MODULE 0
00593  */
00594 // typedef struct {
```

```
00595 //    vubyte  i2adr;           /* 0xFC05_8000 -> 0xFC05_8000 - I2C Address Register
      */
00596 //    vubyte  pack00[3];       /* 0xFC05_8001 -> 0xFC05_8003 - RESERVED
      */
00597 //    vubyte  i2fdr;           /* 0xFC05_8004 -> 0xFC05_8004 - I2C Frequency Divider Register
      */
00598 //    vubyte  pack01[3];       /* 0xFC05_8005 -> 0xFC05_8007 - RESERVED
      */
00599 //    vubyte  i2cr;            /* 0xFC05_8008 -> 0xFC05_8008 - I2C Control Register
      */
00600 //    vubyte  pack02[3];       /* 0xFC05_8009 -> 0xFC05_800B - RESERVED
      */
00601 //    vubyte  i2sr;            /* 0xFC05_800C -> 0xFC05_800C - I2C Status Register
      */
00602 //    vubyte  pack03[3];       /* 0xFC05_800D -> 0xFC05_800F - RESERVED
      */
00603 //    vubyte  i2dr;            /* 0xFC05_8010 -> 0xFC05_8010 - I2C Data I/O Register
      */
00604 //    vubyte  pack04[3];       /* 0xFC05_8011 -> 0xFC05_8013 - RESERVED
      */
00605 // } i2c0struct;
00606
00607 /*
00608  * DMA SERIAL PERIPHERAL INTERFACE 0
00609  */
00610 // typedef struct {
00611 //    vudword mcr;             /* 0xFC05_C000 -> 0xFC05_C003 - Module Configuration Register
      */
00612 //    vubyte  pack00[4];       /* 0xFC05_C004 -> 0xFC05_C007 - RESERVED
      */
00613 //    vudword tcr;             /* 0xFC05_C008 -> 0xFC05_C00B - Transfer Count Register
      */
00614 //    vudword ctar[8];         /* 0xFC05_C00C -> 0xFC05_C02B - Clock and Transfer Attributes Register
      0-7              */
00615 //    vudword sr;              /* 0xFC05_C02C -> 0xFC05_C02F - Status Register
      */
00616 //    vudword rser;            /* 0xFC05_C030 -> 0xFC05_C033 - DMA/Interrupt Request Select and Enable
      Register        */
00617 //    vudword pushr;           /* 0xFC05_C034 -> 0xFC05_C037 - Push Tx FIFO Register
      */
00618 //    vudword popr;            /* 0xFC05_C038 -> 0xFC05_C03B - Pop Rx FIFO Register
      */
00619 //    vudword txfr[16];        /* 0xFC05_C03C -> 0xFC05_C07B - Transmit FIFO Register 0-15
      */
00620 //    vudword rxfr[16];        /* 0xFC05_C07C -> 0xFC05_C0BB - Receive FIFO Register 0-15
      */
00621 // } dspi0struct;
00622
00623 /*
00624  * UART MODULE 0-3 (uarts[4] = 0xFC06_0000 -> 0xFC06_FFFF)
00625  */
00626 typedef struct
00627 {
00628     vubyte umr;           /*      0x0000 ->      0x0000 - UART Mode Registers
      */
00629     vubyte pack00[3];     /*      0x0001 ->      0x0003 - RESERVED
      */
00630     vubyte usr;           /*      0x0004 ->      0x0004 - (Read)  UART Status Register
00631                                                           (Write) UART Clock Select Register
      */
00632     vubyte pack01[3];     /*      0x0005 ->      0x0007 - RESERVED
      */
00633     vubyte ucr;           /*      0x0008 ->      0x0008 - (Read)  Do Not Access
00634                                                           (Write) UART Command Register
      */
00635     vubyte pack02[3];     /*      0x0009 ->      0x000B - RESERVED
      */
00636     vubyte utb;           /*      0x000C ->      0x000C - (Read)  UART Receive Buffer
00637                                                           (Write) UART Transmit Buffer
      */
00638     vubyte pack03[3];     /*      0x000D ->      0x000F - RESERVED
      */
00639     vubyte uipcr;         /*      0x0010 ->      0x0010 - (Read)  UART Input Port Change Register
00640                                                           (Write) UART Auxiliary Control Register
      */
00641     vubyte pack04[3];     /*      0x0011 ->      0x0013 - RESERVED
      */
00642     vubyte uisr;          /*      0x0014 ->      0x0014 - (Read)  UART Interrupt Status Register
00643                                                           (Write) UART Interrupt Mask Register
      */
00644     vubyte pack05[3];     /*      0x0015 ->      0x0017 - RESERVED
      */
00645     vubyte dur;           /*      0x0018 ->      0x0018 - (Read)  Do Not Access
00646                                                           (Write) UART Divider Upper Register
      */
00647     vubyte pack06[3];     /*      0x0019 ->      0x001B - RESERVED
      */
```

```
00648     vubyte dlr;           /*      0x001C ->      0x001C - (Read)  Do Not Access
00649                                                           (Write) UART Divider Lower Register
      */
00650     vubyte pack07[23];    /*      0x001D ->      0x0033 - RESERVED
      */
00651     vubyte uip;           /*      0x0034 ->      0x0034 - (Read)  UART Input Port Register
00652                                                           (Write) Do Not Access
      */
00653     vubyte pack08[3];     /*      0x0035 ->      0x0037 - RESERVED
      */
00654     vubyte ops;           /*      0x0038 ->      0x0038 - (Read)  Do Not Access
00655                                                           (Write) UART Output Port Bit Set Command
      Register       */
00656     vubyte pack09[3];     /*      0x0039 ->      0x003B - RESERVED
      */
00657     vubyte opr;           /*      0x003C ->      0x003C - (Read)  Do Not Access
00658                                                           (Write) UART Output Port Bit Reset Command
      Register     */
00659     vubyte pack10[16323]; /*      0x003D ->      0x3FFF - RESERVED
      */
00660 } uartstruct;
00661
00662 /*
00663  * DMA TIMER MODULE 0-3 (timer[4] = 0xFC07_0000 -> 0xFC07_FFFF)
00664  */
00665 typedef struct
00666 {
00667     vuword tmr;           /*      0x0000 ->      0x0001 - DMA Timer Mode Register
      */
00668     vubyte txmr;          /*      0x0002 ->      0x0002 - DMA Timer Extended Mode Register
      */
00669     vubyte ter;           /*      0x0003 ->      0x0003 - DMA Timer Event Register
      */
00670     vudword trr;          /*      0x0004 ->      0x0007 - DMA Timer Reference Register
      */
00671     vudword tcr;          /*      0x0008 ->      0x000B - DMA Timer Capture Register
      */
00672     vudword tcn;          /*      0x000C ->      0x000F - DMA Timer Counter Register
      */
00673     vubyte pack00[16368]; /*      0x0010 ->      0x3FFF - RESERVED
      */
00674 } timerstruct;
00675
00676 /*
00677  * PROGRAMMABLE INTERRUPT TIMER MODULE 0-3 (pit[4] = 0xFC08_0000 -> 0xFC08_FFFF)
00678  */
00679 typedef struct
00680 {
00681     vuword pcsr;          /*      0x0000 ->      0x0001 - PIT Control and Status Register
      */
00682     vuword pmr;           /*      0x0002 ->      0x0003 - PIT Modulus Register
      */
00683     vuword pcntr;         /*      0x0004 ->      0x0005 - PIT Count Register
      */
00684     vubyte pack00[16378]; /*      0x0006 ->      0x3FFF - RESERVED
      */
00685 } pitstruct;
00686
00687 /*
00688  * EDGE PORT MODULE
00689  */
00690 typedef struct
00691 {
00692     vuword eppar;  /* 0xFC09_0000 -> 0xFC09_0001 - EPORT Pin Assignment Register
      */
00693     vubyte epddr;  /* 0xFC09_0002 -> 0xFC09_0002 - EPORT Data Direction Register
      */
00694     vubyte epier;  /* 0xFC09_0003 -> 0xFC09_0003 - EPORT Interrupt Enable Register
      */
00695     vubyte epdr;   /* 0xFC09_0004 -> 0xFC09_0004 - EPORT Data Register
      */
00696     vubyte eppdr;  /* 0xFC09_0005 -> 0xFC09_0005 - EPORT Pin Data Register
      */
00697     vubyte epfr;   /* 0xFC09_0006 -> 0xFC09_0006 - EPORT Flag Register
      */
00698     vubyte pack00; /* 0xFC09_0007 -> 0xFC09_0007 - RESERVED
      */
00699 } eportstruct;
00700
00701 /*
00702  * ANALOG-TO-DIGITAL CONVERTER
00703  */
00704 typedef struct
00705 {
00706     vuword cr1;     /* 0xFC09_4000 -> 0xFC09_4001 - Control Register 1
      */
00707     vuword cr2;     /* 0xFC09_4002 -> 0xFC09_4003 - Control Register 2
```

```
    */
00708     vuword zccr;     /* 0xFC09_4004 -> 0xFC09_4005 - Zero Crossing Control Register
    */
00709     vuword lst1;     /* 0xFC09_4006 -> 0xFC09_4007 - Channel List Register 1
    */
00710     vuword lst2;     /* 0xFC09_4008 -> 0xFC09_4009 - Channel List Register 2
    */
00711     vuword sdis;     /* 0xFC09_400A -> 0xFC09_400B - Sample Disable Register
    */
00712     vuword sr;       /* 0xFC09_400C -> 0xFC09_400D - Status Register
    */
00713     vuword lsr;      /* 0xFC09_400E -> 0xFC09_400F - Limit Status Register
    */
00714     vuword zcsr;     /* 0xFC09_4010 -> 0xFC09_4011 - Zero Crossing Status Register
    */
00715     vuword rslt[8]; /* 0xFC09_4012 -> 0xFC09_4021 - Result Register 0-7
    */
00716     vuword llmt[8]; /* 0xFC09_4022 -> 0xFC09_4031 - Low Limit Register 0-7
    */
00717     vuword hlmt[8]; /* 0xFC09_4032 -> 0xFC09_4041 - High Limit Register 0-7
    */
00718     vuword ofs[8];  /* 0xFC09_4042 -> 0xFC09_4051 - Offset Register 0-7
    */
00719     vuword pwr;      /* 0xFC09_4052 -> 0xFC09_4053 - Power Control Register
    */
00720     vuword cal;      /* 0xFC09_4054 -> 0xFC09_4055 - Calibration Register
    */
00721     vuword pwr2;     /* 0xFC09_4056 -> 0xFC09_4057 - Power Control Register 2
    */
00722     vuword div;      /* 0xFC09_4058 -> 0xFC09_4059 - Conversion Divisor Register
    */
00723     vuword asdiv;    /* 0xFC09_405A -> 0xFC09_405B - Auto-Standby Divisor Register
    */
00724 } adcstruct;
00725
00726 /*
00727  * DIGITAL-TO-ANALOG CONVERTER 0-1 (dac[2] = 0xFC09_8000 -> 0xFC09_FFFF)
00728  */
00729 typedef struct
00730 {
00731     vuword cr;             /*      0x0000 ->      0x0001 - Control Register
    */
00732     vuword data;           /*      0x0002 ->      0x0003 - Buffered Data Register
    */
00733     vuword step;           /*      0x0004 ->      0x0005 - Step Size Register
    */
00734     vuword min;            /*      0x0006 ->      0x0007 - Minimum Value Register
    */
00735     vuword max;            /*      0x0008 ->      0x0009 - Maximum Value Register
    */
00736     vuword sr;             /*      0x000A ->      0x000B - Status Register
    */
00737     vuword filtcnt;        /*      0x000C ->      0x000D - Filter Count Register
    */
00738     vubyte pack00[16370]; /*      0x000E ->      0x3FFF - RESERVED
    */
00739 } dacstruct;
00740
00741 /*
00742  * SERIAL BOOT FACILITY (SBF)
00743  */
00744 typedef struct
00745 {
00746     vuword sbfsr; /* 0xFC0A_0020 -> 0xFC0A_0021 - Serial Boot Facility Status Register
    */
00747     vuword sbfcr; /* 0xFC0A_0022 -> 0xFC0A_0023 - Serial Boot Facility Control Register
    */
00748 } sbfstruct;
00749
00750 /*
00751  * REAL-TIME CLOCK
00752  */
00753 typedef struct
00754 {
00755     vuword yearmon;         /* 0xFC0A_8000 -> 0xFC0A_8001 - Month and Year Counter Register
    */
00756     vuword days;            /* 0xFC0A_8002 -> 0xFC0A_8003 - Day and Day-of-Week Counter Register
    */
00757     vuword hourmin;         /* 0xFC0A_8004 -> 0xFC0A_8005 - Hour and Minute Counter Register
    */
00758     vuword seconds;         /* 0xFC0A_8006 -> 0xFC0A_8007 - Second Counter Register
    */
00759     vuword alm_yrmon;       /* 0xFC0A_8008 -> 0xFC0A_8009 - Year and Month Alarm Register
    */
00760     vuword alm_days;        /* 0xFC0A_800A -> 0xFC0A_800B - Day Alarm Register
    */
00761     vuword alm_hm;          /* 0xFC0A_800C -> 0xFC0A_800D - Hour and Minute Alarm Register
```

```
    */
00762     vuword alm_sec;        /* 0xFC0A_800E -> 0xFC0A_800F - Second Alarm Register
    */
00763     vuword cr;             /* 0xFC0A_8010 -> 0xFC0A_8011 - Control Register
    */
00764     vuword sr;             /* 0xFC0A_8012 -> 0xFC0A_8013 - Status Register
    */
00765     vuword isr;            /* 0xFC0A_8014 -> 0xFC0A_8015 - Interrupt Status Register
    */
00766     vuword ier;            /* 0xFC0A_8016 -> 0xFC0A_8017 - Interrupt Enable Register
    */
00767     vuword count_dn;       /* 0xFC0A_8018 -> 0xFC0A_8019 - Countdown Timer Register
    */
00768     vubyte pack00[6];      /* 0xFC0A_801A -> 0xFC0A_8019 - RESERVED
    */
00769     vuword cfg;            /* 0xFC0A_8020 -> 0xFC0A_8021 - RTC config register
    */
00770     vuword dst_hour;       /* 0xFC0A_8022 -> 0xFC0A_8023 - Daylight Saving Time Hour Register
    */
00771     vuword dst_mon;        /* 0xFC0A_8024 -> 0xFC0A_8025 - Daylight Saving Time Month Register
    */
00772     vuword dst_day;        /* 0xFC0A_8026 -> 0xFC0A_8027 - Daylight Saving Time Day Register
    */
00773     vuword compen;         /* 0xFC0A_8028 -> 0xFC0A_8029 - Compensation Register
    */
00774     vubyte pack01[8];      /* 0xFC0A_802A -> 0xFC0A_8031 - RESERVED
    */
00775     vuword cntrh;          /* 0xFC0A_8032 -> 0xFC0A_8033 - Count Up High Register
    */
00776     vuword cntrl;          /* 0xFC0A_8034 -> 0xFC0A_8035 - Count Up Low Register
    */
00777     vubyte pack02[10];     /* 0xFC0A_8036 -> 0xFC0A_803F - RESERVED
    */
00778     vudword stdbyram[512]; /* 0xFC0A_8040 -> 0xFC0A_883F - Standby RAM
    */
00779 } rtcstruct;
00780
00781 /*
00782  * SUBSCRIBER IDENTIFICATION MODULE
00783  */
00784 typedef struct
00785 {
00786     vudword cr1;     /* 0xFC0A_C000 -> 0xFC0A_C003 - SIM Port 1 Control Register
    */
00787     vudword setup;   /* 0xFC0A_C004 -> 0xFC0A_C007 - SIM Setup Register
    */
00788     vudword detect1; /* 0xFC0A_C008 -> 0xFC0A_C00B - SIM Port 1 Detect Register
    */
00789     vudword tbuf1;   /* 0xFC0A_C00C -> 0xFC0A_C00F - SIM Port 1 Transmit Buffer Register
    */
00790     vudword rbuf1;   /* 0xFC0A_C010 -> 0xFC0A_C013 - SIM Port 1 Receive Buffer Register
    */
00791     vudword cr0;     /* 0xFC0A_C014 -> 0xFC0A_C017 - SIM Port 0 Control Register
    */
00792     vudword cr;      /* 0xFC0A_C018 -> 0xFC0A_C01B - SIM Control Register
    */
00793     vudword pre;     /* 0xFC0A_C01C -> 0xFC0A_C01F - SIM Clock Prescaler Register
    */
00794     vudword rthr;    /* 0xFC0A_C020 -> 0xFC0A_C023 - SIM Receive Threshold Register
    */
00795     vudword en;      /* 0xFC0A_C024 -> 0xFC0A_C027 - SIM Enable Register
    */
00796     vudword tsr;     /* 0xFC0A_C028 -> 0xFC0A_C02B - SIM Transmit Status Register
    */
00797     vudword rsr;     /* 0xFC0A_C02C -> 0xFC0A_C02F - SIM Receive Status Register
    */
00798     vudword imr;     /* 0xFC0A_C030 -> 0xFC0A_C033 - SIM Interrupt Mask Register
    */
00799     vudword tbuf0;   /* 0xFC0A_C034 -> 0xFC0A_C037 - SIM Port 0 Transmit Buffer Register
    */
00800     vudword rbuf0;   /* 0xFC0A_C038 -> 0xFC0A_C03B - SIM Port 0 Receive Buffer Reigster
    */
00801     vudword detect0; /* 0xFC0A_C03C -> 0xFC0A_C03F - SIM Port 0 Detect Register
    */
00802     vudword format0; /* 0xFC0A_C040 -> 0xFC0A_C043 - SIM Data Format Register
    */
00803     vudword tthr;    /* 0xFC0A_C044 -> 0xFC0A_C047 - SIM Transmit Threshold Register
    */
00804     vudword tgcr;    /* 0xFC0A_C048 -> 0xFC0A_C04B - SIM Transmit Guard Control Register
    */
00805     vudword odcr;    /* 0xFC0A_C04C -> 0xFC0A_C04F - SIM Open Drain Configuration Control Register
    */
00806     vudword rcr;     /* 0xFC0A_C050 -> 0xFC0A_C053 - SIM Reset Control Register
    */
00807     vudword cwtr;    /* 0xFC0A_C054 -> 0xFC0A_C057 - SIM Character Wait Time Register
    */
00808     vudword gpcnt;   /* 0xFC0A_C058 -> 0xFC0A_C05B - SIM General Purpose Counter Register
```

```
    */
00809     vudword div;      /* 0xFC0A_C05C -> 0xFC0A_C05F - SIM Divisor Register
    */
00810     vudword bwt;      /* 0xFC0A_C060 -> 0xFC0A_C063 - SIM Block Wait Time Register
    */
00811     vudword bgt;      /* 0xFC0A_C064 -> 0xFC0A_C067 - SIM Block Guard Time Register
    */
00812     vudword bwth;     /* 0xFC0A_C068 -> 0xFC0A_C06B - SIM Block Wait Time Register High
    */
00813     vudword tfsr;     /* 0xFC0A_C06C -> 0xFC0A_C06F - SIM Transmit FIFO Status Register
    */
00814     vudword rfcr;     /* 0xFC0A_C070 -> 0xFC0A_C073 - SIM Receive FIFO Counter Register
    */
00815     vudword rfwp;     /* 0xFC0A_C074 -> 0xFC0A_C077 - SIM Receive FIFO Write Pointer Register
    */
00816     vudword rfrp;     /* 0xFC0A_C078 -> 0xFC0A_C07B - SIM Receive FIFO Read Pointer Register
    */
00817 } simstruct;
00818
00819 /*
00820  * USB ON-THE-GO
00821  */
00822 typedef struct
00823 {
00824     vudword id;               /* 0xFC0B_0000 -> 0xFC0B_0003 - Identification Register
    */
00825     vudword hwgeneral;        /* 0xFC0B_0004 -> 0xFC0B_0007 - General Hardware Parameters
    */
00826     vudword hwhost;           /* 0xFC0B_0008 -> 0xFC0B_000B - Host Hardware Parameters
    */
00827     vudword hwdevice;         /* 0xFC0B_000C -> 0xFC0B_000F - Device Hardware Parameters
    */
00828     vudword hwtxbuf;          /* 0xFC0B_0010 -> 0xFC0B_0013 - Tx Buffer Hardware Parameters
    */
00829     vudword hwrxbuf;          /* 0xFC0B_0014 -> 0xFC0B_0017 - Rx Buffer Hardware Parameters
    */
00830     vubyte pack00[104];       /* 0xFC0B_0018 -> 0xFC0B_007F - RESERVED
    */
00831     vudword gptimer0ld;       /* 0xFC0B_0080 -> 0xFC0B_0083 - General Purpose Timer 0 Load
    */
00832     vudword gptimer0ctl;      /* 0xFC0B_0084 -> 0xFC0B_0087 - General Purpose Timer 0 Control
    */
00833     vudword gptimer1ld;       /* 0xFC0B_0088 -> 0xFC0B_008B - General Purpose Timer 1 Load
    */
00834     vudword gptimer1ctl;      /* 0xFC0B_008C -> 0xFC0B_008F - General Purpose Timer 1 Control
    */
00835     vubyte pack01[112];       /* 0xFC0B_0090 -> 0xFC0B_00FF - RESERVED
    */
00836     vuword hciversion;        /* 0xFC0B_0100 -> 0xFC0B_0101 - Host Interface Version Number
    */
00837     vubyte pack02;            /* 0xFC0B_0102 -> 0xFC0B_0102 - RESERVED
    */
00838     vubyte caplength;         /* 0xFC0B_0103 -> 0xFC0B_0103 - Capability Register Length
    */
00839     vudword hcsparams;        /* 0xFC0B_0104 -> 0xFC0B_0107 - Host Structural Parameters
    */
00840     vudword hccparams;        /* 0xFC0B_0108 -> 0xFC0B_010B - Host Capability Parameters
    */
00841     vubyte pack03[22];        /* 0xFC0B_010C -> 0xFC0B_0121 - RESERVED
    */
00842     vuword dciversion;        /* 0xFC0B_0122 -> 0xFC0B_0123 - Device Interface Version Number
    */
00843     vudword dccparams;        /* 0xFC0B_0124 -> 0xFC0B_0127 - Device Capability Parameters
    */
00844     vubyte pack04[24];        /* 0xFC0B_0128 -> 0xFC0B_013F - RESERVED
    */
00845     vudword usbcmd;           /* 0xFC0B_0140 -> 0xFC0B_0143 - USB Command
    */
00846     vudword usbsts;           /* 0xFC0B_0144 -> 0xFC0B_0147 - USB Status
    */
00847     vudword usbintr;          /* 0xFC0B_0148 -> 0xFC0B_014B - USB Interrupt Enable
    */
00848     vudword frindex;          /* 0xFC0B_014C -> 0xFC0B_014F - USB Frame Index
    */
00849     vubyte pack05[4];         /* 0xFC0B_0150 -> 0xFC0B_0153 - RESERVED
    */
00850     vudword periodiclstbase; /* 0xFC0B_0154 -> 0xFC0B_0157 - (Host Mode)   Periodic Frame List Base
    Address
00851                                                            - (Device Mode) Device Address
    */
00852     vudword asynclistaddr;    /* 0xFC0B_0158 -> 0xFC0B_015B - (Host Mode)   Current Asynchronous List
    Address
00853                                                            - (Device Mode) Address at Endpoint List
    */
00854     vudword ttctrl;           /* 0xFC0B_015C -> 0xFC0B_015F - Host TT Asynchronous Buffer Control
    */
00855     vudword burstsize;        /* 0xFC0B_0160 -> 0xFC0B_0163 - Master Interface Data Burst Size
```

```
00856     vudword txfilltuning;     /* 0xFC0B_0164 -> 0xFC0B_0167 - Host Transmit FIFO Tuning Control
00857     */
00857     vubyte pack06[8];         /* 0xFC0B_0168 -> 0xFC0B_016F - RESERVED
00858     */
00858     vudword ulpi_viewport;    /* 0xFC0B_0170 -> 0xFC0B_0173 - ULPI Register Access
00859     */
00859     vubyte pack07[12];        /* 0xFC0B_0174 -> 0xFC0B_017F - RESERVED
00860     */
00860     vudword configflag;       /* 0xFC0B_0180 -> 0xFC0B_0183 - Configure Flag Register
00861     */
00861     vudword portsc1;          /* 0xFC0B_0184 -> 0xFC0B_0187 - Port Status/Control
00862     */
00862     vubyte pack08[28];        /* 0xFC0B_0188 -> 0xFC0B_01A3 - RESERVED
00863     */
00863     vudword otgsc;            /* 0xFC0B_01A4 -> 0xFC0B_01A7 - On-the-Go Status and Control
00864     */
00864     vudword mode;             /* 0xFC0B_01A8 -> 0xFC0B_01AB - USB Mode Register
00865     */
00865     vudword epsetupsr;        /* 0xFC0B_01AC -> 0xFC0B_01AF - Endpoint Setup Status Register
00866     */
00866     vudword epprime;          /* 0xFC0B_01B0 -> 0xFC0B_01B3 - Endpoint Initialization
00867     */
00867     vudword epflush;          /* 0xFC0B_01B4 -> 0xFC0B_01B7 - Endpoint De-initialize
00868     */
00868     vudword epsr;             /* 0xFC0B_01B8 -> 0xFC0B_01BB - Endpoint Status Register
00869     */
00869     vudword epcomplete;       /* 0xFC0B_01BC -> 0xFC0B_01BF - Endpoint Complete
00870     */
00870     vudword epcr0;            /* 0xFC0B_01C0 -> 0xFC0B_01C3 - Endpoint Control Register 0
00871     */
00871     vudword epcr1;            /* 0xFC0B_01C4 -> 0xFC0B_01C7 - Endpoint Control Register 1
00872     */
00872     vudword epcr2;            /* 0xFC0B_01C8 -> 0xFC0B_01CB - Endpoint Control Register 2
00873     */
00873     vudword epcr3;            /* 0xFC0B_01CC -> 0xFC0B_01CF - Endpoint Control Register 3
00874     */
00874 } usb_otgstruct;
00875
00876 /*
00877  * USB HOST CONTROLLER
00878  */
00879 typedef struct
00880 {
00881     vudword id;               /* 0xFC0B_4000 -> 0xFC0B_4003 - Identification Register
00882     */
00882     vudword hwgeneral;        /* 0xFC0B_4004 -> 0xFC0B_4007 - General Hardware Parameters
00883     */
00883     vudword hwhost;           /* 0xFC0B_4008 -> 0xFC0B_400B - Host Hardware Parameters
00884     */
00884     vubyte pack00[4];         /* 0xFC0B_400C -> 0xFC0B_400F - RESERVED
00885     */
00885     vudword hwtxbuf;          /* 0xFC0B_4010 -> 0xFC0B_4013 - Tx Buffer Hardware Parameters
00886     */
00886     vudword hwrxbuf;          /* 0xFC0B_4014 -> 0xFC0B_4017 - Rx Buffer Hardware Parameters
00887     */
00887     vubyte pack01[232];       /* 0xFC0B_4018 -> 0xFC0B_40FF - RESERVED
00888     */
00888     vuword hciversion;        /* 0xFC0B_4100 -> 0xFC0B_4101 - Host Interface Version Number
00889     */
00889     vubyte pack02;            /* 0xFC0B_4102 -> 0xFC0B_4102 - RESERVED
00890     */
00890     vubyte caplength;         /* 0xFC0B_4103 -> 0xFC0B_4103 - Capability Register Length
00891     */
00891     vudword hcsparams;        /* 0xFC0B_4104 -> 0xFC0B_4107 - Host Structural Parameters
00892     */
00892     vudword hccparams;        /* 0xFC0B_4108 -> 0xFC0B_410B - Host Capability Parameters
00893     */
00893     vubyte pack03[52];        /* 0xFC0B_410C -> 0xFC0B_413F - RESERVED
00894     */
00894     vudword usbcmd;           /* 0xFC0B_4140 -> 0xFC0B_4143 - USB Command
00895     */
00895     vudword usbsts;           /* 0xFC0B_4144 -> 0xFC0B_4147 - USB Status
00896     */
00896     vudword usbintr;          /* 0xFC0B_4148 -> 0xFC0B_414B - USB Interrupt Enable
00897     */
00897     vudword frindex;          /* 0xFC0B_414C -> 0xFC0B_414F - USB Frame Index
00898     */
00898     vubyte pack04[4];         /* 0xFC0B_4150 -> 0xFC0B_4153 - RESERVED
00899     */
00899     vudword periodiclstbase;  /* 0xFC0B_4154 -> 0xFC0B_4157 - Periodic Frame List Base Address
00900     */
00900     vudword asynclistaddr;    /* 0xFC0B_4158 -> 0xFC0B_415B - Current Asynchronous List Address
00901     */
00901     vudword ttctrl;           /* 0xFC0B_415C -> 0xFC0B_415F - Host TT Asynchronous Buffer Control
00902     */
00902     vudword burstsize;        /* 0xFC0B_4160 -> 0xFC0B_4163 - Master Interface Data Burst Size
```

```
    */
00903     vudword txfilltuning;     /* 0xFC0B_4164 -> 0xFC0B_4167 - Host Transmit FIFO Tuning Control
    */
00904     vubyte pack05[24];        /* 0xFC0B_4168 -> 0xFC0B_417F - RESERVED
    */
00905     vudword configflag;       /* 0xFC0B_4180 -> 0xFC0B_4183 - Configure Flag Register
    */
00906     vudword portsc1;          /* 0xFC0B_4184 -> 0xFC0B_4187 - Port Status/Control
    */
00907     vubyte pack06[32];        /* 0xFC0B_4188 -> 0xFC0B_41A7 - RESERVED
    */
00908     vudword mode;             /* 0xFC0B_41A8 -> 0xFC0B_41AB - USB Mode Register
    */
00909 } usb_hoststruct;
00910
00911 /*
00912  * DDR1/2 SDRAM MEMORY CONTROLLER
00913  */
00914 typedef struct
00915 {
00916     vudword cr[64];     /* 0xFC0B_8000 -> 0xFC0B_80FF - Control Register 0-63 (46-52, 54, 61-63
    reserved)       */
00917     vubyte pack00[172]; /* 0xFC0B_8100 -> 0xFC0B_81AB - RESERVED
    */
00918     vudword padcr;      /* 0xFC0B_81AC -> 0xFC0B_81AF - I/O Pad Control Register
    */
00919 } ddrmcstruct;
00920
00921 /*
00922  * SYNCHRONOUS SERIAL INTERFACE 0
00923  */
00924 typedef struct
00925 {
00926     vudword tx0;      /* 0xFC0B_C000 -> 0xFC0B_C003 - Transmit Data Register 0
    */
00927     vudword tx1;      /* 0xFC0B_C004 -> 0xFC0B_C007 - Transmit Data Register 1
    */
00928     vudword rx0;      /* 0xFC0B_C008 -> 0xFC0B_C00B - Receive Data Register 0
    */
00929     vudword rx1;      /* 0xFC0B_C00C -> 0xFC0B_C00F - Receive Data Register 1
    */
00930     vudword cr;       /* 0xFC0B_C010 -> 0xFC0B_C013 - Control Register
    */
00931     vudword isr;      /* 0xFC0B_C014 -> 0xFC0B_C017 - Interrupt Status Register
    */
00932     vudword ier;      /* 0xFC0B_C018 -> 0xFC0B_C01B - Interrupt Enable Register
    */
00933     vudword tcr;      /* 0xFC0B_C01C -> 0xFC0B_C01F - Transmit Configuration Register
    */
00934     vudword rcr;      /* 0xFC0B_C020 -> 0xFC0B_C023 - Receive Configuration Register
    */
00935     vudword ccr;      /* 0xFC0B_C024 -> 0xFC0B_C027 - Clock Control Register
    */
00936     vubyte pack00[4]; /* 0xFC0B_C028 -> 0xFC0B_C02B - RESERVED
    */
00937     vudword fcsr;     /* 0xFC0B_C02C -> 0xFC0B_C02F - FIFO Control/Status Register
    */
00938     vubyte pack01[8]; /* 0xFC0B_C030 -> 0xFC0B_C037 - RESERVED
    */
00939     vudword acr;      /* 0xFC0B_C038 -> 0xFC0B_C03B - AC97 Control Register
    */
00940     vudword acadd;    /* 0xFC0B_C03C -> 0xFC0B_C03F - AC97 Command Address Register
    */
00941     vudword acdat;    /* 0xFC0B_C040 -> 0xFC0B_C043 - AC97 Command Data Register
    */
00942     vudword atag;     /* 0xFC0B_C044 -> 0xFC0B_C047 - AC97 Tag Register
    */
00943     vudword tmask;    /* 0xFC0B_C048 -> 0xFC0B_C04B - Transmit Time Slot Mask Register
    */
00944     vudword rmask;    /* 0xFC0B_C04C -> 0xFC0B_C04F - Receive Time Slot Mask Register
    */
00945     vudword accsr;    /* 0xFC0B_C050 -> 0xFC0B_C053 - AC97 Channel Status Register
    */
00946     vudword accen;    /* 0xFC0B_C054 -> 0xFC0B_C057 - AC97 Channel Enable Register
    */
00947     vudword accdis;   /* 0xFC0B_C058 -> 0xFC0B_C05B - AC97 Channel Disable Register
    */
00948 } ssi0struct;
00949
00950 /*
00951  * CLOCK MODULE (PHASE-LOCKED LOOP)
00952  */
00953 typedef struct
00954 {
00955     vudword pll_cr; /* 0xFC0C_0000 -> 0xFC0C_0003 - PLL Control Register
    */
00956     vudword pll_dr; /* 0xFC0C_0004 -> 0xFC0C_0007 - PLL Divider Register
```

```
    */
00957     vudword pll_sr; /* 0xFC0C_0008 -> 0xFC0C_000B - PLL Status Register
    */
00958 } clockstruct;
00959
00960 /*
00961  * RANDOM NUMBER GENERATOR
00962  */
00963 typedef struct
00964 {
00965     vudword ver; /* 0xFC0C_4000 -> 0xFC0C_4003 - Version ID Register
    */
00966     vudword cmd; /* 0xFC0C_4004 -> 0xFC0C_4007 - Command Register
    */
00967     vudword cr;  /* 0xFC0C_4008 -> 0xFC0C_400B - Control Register
    */
00968     vudword sr;  /* 0xFC0C_400C -> 0xFC0C_400F - Status Register
    */
00969     vudword esr; /* 0xFC0C_4010 -> 0xFC0C_4013 - Error Status Register
    */
00970     vudword out; /* 0xFC0C_4014 -> 0xFC0C_4017 - Output FIFO
    */
00971     vudword er;  /* 0xFC0C_4018 -> 0xFC0C_401B - Entropy Register
    */
00972 } rngstruct;
00973
00974 /*
00975  * SYNCHRONOUS SERIAL INTERFACE 1
00976  */
00977 typedef struct
00978 {
00979     vudword tx0;      /* 0xFC0C_8000 -> 0xFC0C_8003 - Transmit Data Register 0
    */
00980     vudword tx1;      /* 0xFC0C_8004 -> 0xFC0C_8007 - Transmit Data Register 1
    */
00981     vudword rx0;      /* 0xFC0C_8008 -> 0xFC0C_800B - Receive Data Register 0
    */
00982     vudword rx1;      /* 0xFC0C_800C -> 0xFC0C_800F - Receive Data Register 1
    */
00983     vudword cr;       /* 0xFC0C_8010 -> 0xFC0C_8013 - Control Register
    */
00984     vudword isr;      /* 0xFC0C_8014 -> 0xFC0C_8017 - Interrupt Status Register
    */
00985     vudword ier;      /* 0xFC0C_8018 -> 0xFC0C_801B - Interrupt Enable Register
    */
00986     vudword tcr;      /* 0xFC0C_801C -> 0xFC0C_801F - Transmit Configuration Register
    */
00987     vudword rcr;      /* 0xFC0C_8020 -> 0xFC0C_8023 - Receive Configuration Register
    */
00988     vudword ccr;      /* 0xFC0C_8024 -> 0xFC0C_8027 - Clock Control Register
    */
00989     vubyte pack00[4]; /* 0xFC0C_8028 -> 0xFC0C_802B - RESERVED
    */
00990     vudword fcsr;     /* 0xFC0C_802C -> 0xFC0C_802F - FIFO Control/Status Register
    */
00991     vubyte pack01[8]; /* 0xFC0C_8030 -> 0xFC0C_8037 - RESERVED
    */
00992     vudword acr;      /* 0xFC0C_8038 -> 0xFC0C_803B - AC97 Control Register
    */
00993     vudword acadd;    /* 0xFC0C_803C -> 0xFC0C_803F - AC97 Command Address Register
    */
00994     vudword acdat;    /* 0xFC0C_8040 -> 0xFC0C_8043 - AC97 Command Data Register
    */
00995     vudword atag;     /* 0xFC0C_8044 -> 0xFC0C_8047 - AC97 Tag Register
    */
00996     vudword tmask;    /* 0xFC0C_8048 -> 0xFC0C_804B - Transmit Time Slot Mask Register
    */
00997     vudword rmask;    /* 0xFC0C_804C -> 0xFC0C_804F - Receive Time Slot Mask Register
    */
00998     vudword accsr;    /* 0xFC0C_8050 -> 0xFC0C_8053 - AC97 Channel Status Register
    */
00999     vudword accen;    /* 0xFC0C_8054 -> 0xFC0C_8057 - AC97 Channel Enable Register
    */
01000     vudword accdis;   /* 0xFC0C_8058 -> 0xFC0C_805B - AC97 Channel Disable Register
    */
01001 } ssi1struct;
01002
01003 /*
01004  * ENHANCED SECURE DIGITAL HOST CONTROLLER
01005  */
01006 typedef struct
01007 {
01008     vudword dsaddr;     /* 0xFC0C_C000 -> 0xFC0C_C003 - DMA System Address
    */
01009     vudword blkattr;    /* 0xFC0C_C004 -> 0xFC0C_C007 - Block Attributes
    */
01010     vudword cmdarg;     /* 0xFC0C_C008 -> 0xFC0C_C00B - Command Argument
```

```
    */
01011     vudword xfertyp;    /* 0xFC0C_C00C -> 0xFC0C_C00F - Command Transfer Type
    */
01012     vudword cmdrsp0;    /* 0xFC0C_C010 -> 0xFC0C_C013 - Command Response 0
    */
01013     vudword cmdrsp1;    /* 0xFC0C_C014 -> 0xFC0C_C017 - Command Response 1
    */
01014     vudword cmdrsp2;    /* 0xFC0C_C018 -> 0xFC0C_C01B - Command Response 2
    */
01015     vudword cmdrsp3;    /* 0xFC0C_C01C -> 0xFC0C_C01F - Command Response 3
    */
01016     vudword datport;    /* 0xFC0C_C020 -> 0xFC0C_C023 - Data Buffer Access Port
    */
01017     vudword prsstat;    /* 0xFC0C_C024 -> 0xFC0C_C027 - Present State
    */
01018     vudword proctl;     /* 0xFC0C_C028 -> 0xFC0C_C02B - Protocol Control
    */
01019     vudword sysctl;     /* 0xFC0C_C02C -> 0xFC0C_C02F - System Control
    */
01020     vudword irqstat;    /* 0xFC0C_C030 -> 0xFC0C_C033 - Interrupt Status
    */
01021     vudword irqstaten;  /* 0xFC0C_C034 -> 0xFC0C_C037 - Interrupt Status Enable
    */
01022     vudword irqsigen;   /* 0xFC0C_C038 -> 0xFC0C_C03B - Interrupt Signal Enable
    */
01023     vudword autoc12err; /* 0xFC0C_C03C -> 0xFC0C_C03F - Auto CMD12 Status
    */
01024     vudword hostcapblt; /* 0xFC0C_C040 -> 0xFC0C_C043 - Host Controller Capabilities
    */
01025     vudword wml;        /* 0xFC0C_C044 -> 0xFC0C_C047 - Watermark Level
    */
01026     vubyte pack00[8];   /* 0xFC0C_C048 -> 0xFC0C_C04F - RESERVERED
    */
01027     vudword fevt;       /* 0xFC0C_C050 -> 0xFC0C_C053 - Force Event
    */
01028     vudword admaesr;    /* 0xFC0C_C054 -> 0xFC0C_C057 - ADMA Error Status
    */
01029     vudword admasar;    /* 0xFC0C_C058 -> 0xFC0C_C05B - ADMA System Address
    */
01030     vubyte pack01[160]; /* 0xFC0C_C05C -> 0xFC0C_C0FB - RESERVED
    */
01031     vudword hostver;    /* 0xFC0C_C0FC -> 0xFC0C_C0FF - Host Controller Version
    */
01032 } sdhcstruct;
01033
01034 /*
01035  * MIB COUNTER BLOCK RMON TRANSMIT
01036  */
01037 typedef struct
01038 {
01039     vudword drop;        /*        0x00 ->        0x03 - Count of Frames Not Counted Correctly (Not
    Implemented) */
01040     vudword packets;     /*        0x04 ->        0x07 - RMON Tx Packet Count
    */
01041     vudword bc_pkt;      /*        0x08 ->        0x0B - RMON Tx Broadcast Packets
    */
01042     vudword mc_pkt;      /*        0x0C ->        0x0F - RMON Tx Multicast Packets
    */
01043     vudword crc_align;   /*        0x10 ->        0x13 - RMON Tx Packets with CRC/Align Error
    */
01044     vudword undersize;   /*        0x14 ->        0x17 - RMON Tx Packets < 64 Bytes, Good CRC
    */
01045     vudword oversize;    /*        0x18 ->        0x1B - RMON Tx Packets > MAX_FL Bytes, Good CRC
    */
01046     vudword frag;        /*        0x1C ->        0x1F - RMON Tx Packets < 64 Bytes, Bad CRC
    */
01047     vudword jab;         /*        0x20 ->        0x23 - RMON Tx Packets > MAX_FL Bytes, Bad CRC
    */
01048     vudword col;         /*        0x24 ->        0x27 - RMON Tx Collision Count
    */
01049     vudword p64;         /*        0x28 ->        0x2B - RMON Tx 64 Byte Packets
    */
01050     vudword p65to127;    /*        0x2C ->        0x2F - RMON Tx 65 to 127 Byte Packets
    */
01051     vudword p128to255;   /*        0x30 ->        0x33 - RMON Tx 128 to 255 Byte Packets
    */
01052     vudword p256to511;   /*        0x34 ->        0x37 - RMON Tx 256 to 511 Byte Packets
    */
01053     vudword p512to1023;  /*        0x38 ->        0x3B - RMON Tx 512 to 1023 Byte Packets
    */
01054     vudword p1024to2047; /*        0x3C ->        0x3F - RMON Tx 1024 to 2047 Byte Packets
    */
01055     vudword p_gte2048;   /*        0x40 ->        0x43 - RMON Tx Packets with > 2048 Bytes
    */
01056     vudword octets;      /*        0x44 ->        0x47 - RMON Tx Octets
    */
01057 } rmon_tstruct;
```

```
01058
01059 /*
01060  * MIB COUNTER BLOCK IEEE TRANSMIT
01061  */
01062 typedef struct
01063 {
01064     vudword drop;       /*        0x00 ->        0x03 - Count of Frames Not Counted Correctly (Not
      Implemented) */
01065     vudword frame_ok;   /*        0x04 ->        0x07 - Frames Transmitted OK
      */
01066     vudword scol;       /*        0x08 ->        0x0B - Frames Transmitted with Single Collision
      */
01067     vudword mcol;       /*        0x0C ->        0x0F - Frames Transmitted with Multiple Collisions
      */
01068     vudword def;        /*        0x10 ->        0x13 - Frames Transmitted after Deferral Delay
      */
01069     vudword lcol;       /*        0x14 ->        0x17 - Frames Transmitted with Late Collision
      */
01070     vudword excol;      /*        0x18 ->        0x1B - Frames Transmitted with Excessive Collisions
      */
01071     vudword macerr;     /*        0x1C ->        0x1F - Frames Transmitted with Tx FIFO Underrun
      */
01072     vudword cserr;      /*        0x20 ->        0x23 - Frames Transmitted with Carrier Sense Error
      */
01073     vudword sqe;        /*        0x24 ->        0x27 - Frames Transmitted with SQE Error (Not
      Implemented)     */
01074     vudword fdxfc;      /*        0x28 ->        0x2B - Flow Control Pause Frames Transmitted
      */
01075     vudword octets_ok; /*        0x2C ->        0x2F - Octet Count for Frames Transmitted without
      Error        */
01076 } ieee_tstruct;
01077
01078 /*
01079  * MIB COUNTER BLOCK RMON RECEIVE
01080  */
01081 typedef struct
01082 {
01083     vubyte pack00[4];     /*        0x00 ->        0x03 - RESERVED
      */
01084     vudword packets;      /*        0x04 ->        0x07 - RMON Rx Packet Count
      */
01085     vudword bc_pkt;       /*        0x08 ->        0x0B - RMON Rx Broadcast Packets
      */
01086     vudword mc_pkt;       /*        0x0C ->        0x0F - RMON Rx Multicast Packets
      */
01087     vudword crc_align;    /*        0x10 ->        0x13 - RMON Rx Packets with CRC/Align Error
      */
01088     vudword undersize;    /*        0x14 ->        0x17 - RMON Rx Packets < 64 Bytes, Good CRC
      */
01089     vudword oversize;     /*        0x18 ->        0x1B - RMON Rx > MAX_FL Bytes, Good CRC
      */
01090     vudword frag;         /*        0x1C ->        0x1F - RMON Rx packets < 64 Bytes, Bad CRC
      */
01091     vudword jab;          /*        0x20 ->        0x23 - RMON Rx Packets > MAX_FL Bytes, Bad CRC
      */
01092     vudword resvd_0;      /*        0x24 ->        0x27 - RESERVED
      */
01093     vudword p64;          /*        0x28 ->        0x2B - RMON Rx 64 Byte Packets
      */
01094     vudword p65to127;     /*        0x2C ->        0x2F - RMON Rx 65 to 127 Byte Packets
      */
01095     vudword p128to255;    /*        0x30 ->        0x33 - RMON Rx 128 to 255 Byte Packets
      */
01096     vudword p256to511;    /*        0x34 ->        0x37 - RMON Rx 256 to 511 Byte Packets
      */
01097     vudword p512to1023;   /*        0x38 ->        0x3B - RMON Rx 512 to 1023 Byte Packets
      */
01098     vudword p1024to2047; /*        0x3C ->        0x3F - RMON Rx 1024 to 2047 Byte Packets
      */
01099     vudword p_gte2048;    /*        0x40 ->        0x43 - RMON Rx Packets with > 2048 Bytes
      */
01100     vudword octets;       /*        0x44 ->        0x47 - RMON Rx Octets
      */
01101 } rmon_rstruct;
01102
01103 /*
01104  * MIB COUNTER BLOCK IEEE RECEIVE
01105  */
01106 typedef struct
01107 {
01108     vudword drop;       /*        0x00 ->        0x03 - Count of Frames Not Counted Correctly
      */
01109     vudword frame_ok;   /*        0x04 ->        0x07 - Frames Received OK
      */
01110     vudword crc;        /*        0x08 ->        0x0B - Frames Received with CRC Error
      */
01111     vudword align;      /*        0x0C ->        0x0F - Frames Received with Alignment Error
```

```
    */
01112     vudword macerr;     /*        0x10 ->        0x13 - Receive FIFO Overflow Count
    */
01113     vudword fdxfc;      /*        0x14 ->        0x17 - Flow Control Pause Frames Received
    */
01114     vudword octets_ok; /*        0x18 ->        0x1B - Octet Count for Frames Received without Error
    */
01115 } ieee_rstruct;
01116
01117 /*
01118  * 10/100 MBPS ETHERNET MAC-NET CORE 0-1 (fec[2] = 0xFC0D_4000 -> 0xFC0D_BFFF)
01119  */
01120 typedef struct
01121 {
01122     vudword l;
01123     vudword u;
01124 } sumac;
01125 typedef struct
01126 {
01127     vubyte pack00[4];         /*      0x4000 ->      0x4003 - RESERVED
    */
01128     vudword eir;              /*      0x4004 ->      0x4007 - Interrupt Event Register
    */
01129     vudword eimr;             /*      0x4008 ->      0x400B - Interrupt Mask Register
    */
01130     vubyte pack01[4];         /*      0x400C ->      0x400F - RESERVED
    */
01131     vudword rdar;             /*      0x4010 ->      0x4013 - Receive Descriptor Active Register
    */
01132     vudword tdar;             /*      0x4014 ->      0x4017 - Transmit Descriptor Active Register
    */
01133     vubyte pack02[12];        /*      0x4018 ->      0x4023 - RESERVED
    */
01134     vudword ecr;              /*      0x4024 ->      0x4027 - Ethernet Control Register
    */
01135     vubyte pack03[24];        /*      0x4028 ->      0x403F - RESERVED
    */
01136     vudword mdata;            /*      0x4040 ->      0x4043 - MII Data Register (MII Management Frame
    Register)       */
01137     vudword mscr;             /*      0x4044 ->      0x4047 - MII Speed Control Register
    */
01138     vubyte pack04[28];        /*      0x4048 ->      0x4063 - RESERVED
    */
01139     vudword mibc;             /*      0x4064 ->      0x4067 - MIB Control/Status Register
    */
01140     vubyte pack05[28];        /*      0x4068 ->      0x4083 - RESERVED
    */
01141     vudword rcr;              /*      0x4084 ->      0x4087 - Receive Control Register
    */
01142     vubyte pack06[60];        /*      0x4088 ->      0x40C3 - RESERVED
    */
01143     vudword tcr;              /*      0x40C4 ->      0x40C7 - Transmit Control Register
    */
01144     vubyte pack07[28];        /*      0x40C8 ->      0x40E3 - RESERVED
    */
01145     vudword palr;             /*      0x40E4 ->      0x40E7 - Physical Address Low Register
    */
01146     vudword paur;             /*      0x40E8 ->      0x40EB - Physical Address High Register
    */
01147     vudword opd;              /*      0x40EC ->      0x40EF - Opcode/Pause Duration Register
    */
01148     vubyte pack08[40];        /*      0x40F0 ->      0x4117 - RESERVED
    */
01149     vudword iaur;             /*      0x4118 ->      0x411B - Descriptor Individual Upper Address
    Register            */
01150     vudword ialr;             /*      0x411C ->      0x411F - Descriptor Individual Lower Address
    Register            */
01151     vudword gaur;             /*      0x4120 ->      0x4123 - Descriptor Group Upper Address Register
    */
01152     vudword galr;             /*      0x4124 ->      0x4127 - Descriptor Group Lower Address Register
    */
01153     vubyte pack09[28];        /*      0x4128 ->      0x4143 - RESERVED
    */
01154     vudword tfwr;             /*      0x4144 ->      0x4147 - Transmit FIFO Watermark and Store/Foward
    Control        */
01155     vubyte pack10[4];         /*      0x4148 ->      0x414B - RESERVED
    */
01156     vudword frbr;             /*      0x414C ->      0x414F - FIFO Receive Bound Register (Not
    Implemented)            */
01157     vudword frsr;             /*      0x4150 ->      0x4153 - FIFO Receive Start Register (Not
    Implemented)            */
01158     vubyte pack11[44];        /*      0x4154 ->      0x417F - RESERVED
    */
01159     vudword erdsr;            /*      0x4180 ->      0x4183 - Receive Descriptor Ring Start Register
    */
01160     vudword etdsr;            /*      0x4184 ->      0x4187 - Transmit Descriptor Ring Start Register
    */
```

```
01161     vudword emrbr;           /*      0x4188 ->      0x418B - Maximum Receive Buffer Size
      */
01162     vubyte pack12[4];        /*      0x418C ->      0x418F - RESERVED
      */
01163     vudword rsfl;            /*      0x4190 ->      0x4193 - Receive FIFO Section Full Threshold
      */
01164     vudword rsem;            /*      0x4194 ->      0x4197 - Receive FIFO Section Empty Threshold
      */
01165     vudword raem;            /*      0x4198 ->      0x419B - Receive FIFO Almost Empty Threshold
      */
01166     vudword rafl;            /*      0x419C ->      0x419F - Receive FIFO Almost Full Threshold
      */
01167     vudword tsem;            /*      0x41A0 ->      0x41A3 - Transmit FIFO Section Empty Threshold
      */
01168     vudword taem;            /*      0x41A4 ->      0x41A7 - Transmit FIFO Almost Empty Threshold
      */
01169     vudword tafl;            /*      0x41A8 ->      0x41AB - Transmit FIFO Almost Full Threshold
      */
01170     vudword tipg;            /*      0x41AC ->      0x41AF - Transmit Inter-Packet Gap
      */
01171     vudword ftrl;            /*      0x41B0 ->      0x41B3 - Frame Truncation Length
      */
01172     vubyte pack13[12];       /*      0x41B4 ->      0x41BF - RESERVED
      */
01173     vudword tacc;            /*      0x41C0 ->      0x41C3 - Transmit Accelerator Function
      Configuration             */
01174     vudword racc;            /*      0x41C4 ->      0x41C7 - Receive Accelerator Function
      Configuration              */
01175     vubyte pack14[56];       /*      0x41C8 ->      0x41FF - RESERVED
      */
01176     rmon_tstruct fec_rmon_t; /*      0x4200 ->      0x4247 - MIB Counter Block RMON Transmit
      */
01177     ieee_tstruct fec_ieee_t; /*      0x4248 ->      0x4277 - MIB Counter Block IEEE Transmit
      */
01178     vubyte pack15[8];        /*      0x4278 ->      0x427F - RESERVED
      */
01179     rmon_rstruct fec_rmon_r; /*      0x4280 ->      0x42C7 - MIB Counter Block RMON Receive
      */
01180     ieee_rstruct fec_ieee_r; /*      0x42C8 ->      0x42E3 - MIB Counter Block IEEE Receive
      */
01181     vubyte pack16[284];      /*      0x42E4 ->      0x43FF - RESERVED
      */
01182     vudword atcr;            /*      0x4400 ->      0x4403 - Timer Control Register
      */
01183     vudword atvr;            /*      0x4404 ->      0x4407 - Timer Value Register
      */
01184     vudword atoff;           /*      0x4408 ->      0x440B - Offset Value for One-Shot Event
      Generation               */
01185     vudword atper;           /*      0x440C ->      0x440F - Timer Period
      */
01186     vudword atcor;           /*      0x4410 ->      0x4413 - Correction Counter Wrap-Around Value
      */
01187     vudword atinc;           /*      0x4414 ->      0x4417 - Timestamp Clock Period and Correction
      Increment         */
01188     vudword atstmp;          /*      0x4418 ->      0x441B - Timestamp of Last Transmitted Frame
      */
01189     vubyte pack17[228];      /*      0x441C ->      0x44FF - RESERVED
      */
01190     volatile sumac smac[4];  /*      0x4500 ->      0x451F - Supplemental MAC Address 0-3
      */
01191     vubyte pack18[15072];    /*      0x4520 ->      0x7FFF - RESERVED
      */
01192 } fecstruct;
01193
01194 typedef struct
01195 {
01196     vudword low;
01197     vudword high;
01198 } mactabentry;
01199
01200 /*
01201  * ETHERNET SWITCH
01202  */
01203 typedef struct
01204 {
01205     vudword rev;                /* 0xFC0D_C000 -> 0xFC0D_C003 - Revision
      */
01206     vudword scr;                /* 0xFC0D_C004 -> 0xFC0D_C007 - Scratch Register
      */
01207     vudword per;                /* 0xFC0D_C008 -> 0xFC0D_C00B - Port Enable Register
      */
01208     vubyte pack00[4];           /* 0xFC0D_C00C -> 0xFC0D_C00F - RESERVED
      */
01209     vudword vlanv;              /* 0xFC0D_C010 -> 0xFC0D_C013 - VLAN Verify
      */
01210     vudword dbcr;               /* 0xFC0D_C014 -> 0xFC0D_C017 - Default Broadcast Resolution
      */
```

```
01211     vudword dmcr;               /* 0xFC0D_C018 -> 0xFC0D_C01B - Default Multicast Resolution
      */
01212     vudword bklr;               /* 0xFC0D_C01C -> 0xFC0D_C01F - Blocking and Learning Enable
      */
01213     vudword bmpc;               /* 0xFC0D_C020 -> 0xFC0D_C023 - Bridge Management Port Configuration
      */
01214     vudword mode;               /* 0xFC0D_C024 -> 0xFC0D_C027 - Mode Configuration
      */
01215     vudword vimsel;             /* 0xFC0D_C028 -> 0xFC0D_C02B - VLAN Input Manipulation Select
      */
01216     vudword vomsel;             /* 0xFC0D_C02C -> 0xFC0D_C02F - VLAN Output Manipulation Select
      */
01217     vudword vimen;              /* 0xFC0D_C030 -> 0xFC0D_C033 - VLAN Input Manipulation Enable
      */
01218     vudword vid;                /* 0xFC0D_C034 -> 0xFC0D_C037 - VLAN Tag ID
      */
01219     vubyte pack01[8];           /* 0xFC0D_C038 -> 0xFC0D_C03F - RESERVED
      */
01220     vudword mcr;                /* 0xFC0D_C040 -> 0xFC0D_C043 - Mirror Control Register
      */
01221     vudword egmap;              /* 0xFC0D_C044 -> 0xFC0D_C047 - Egress Port Definitions
      */
01222     vudword ingmap;             /* 0xFC0D_C048 -> 0xFC0D_C04B - Ingress Port Definitions
      */
01223     vudword ingsal;             /* 0xFC0D_C04C -> 0xFC0D_C04F - Ingress Source MAC Address Low
      */
01224     vudword ingsah;             /* 0xFC0D_C050 -> 0xFC0D_C053 - Ingress Source MAC Address High
      */
01225     vudword ingdal;             /* 0xFC0D_C054 -> 0xFC0D_C057 - Ingress Destination MAC Address Low
      */
01226     vudword ingdah;             /* 0xFC0D_C058 -> 0xFC0D_C05B - Ingress Destination MAC Address High
      */
01227     vudword egsal;              /* 0xFC0D_C05C -> 0xFC0D_C05F - Egress Source MAC Address Low
      */
01228     vudword egsah;              /* 0xFC0D_C060 -> 0xFC0D_C063 - Egress Source MAC Address High
      */
01229     vudword egdal;              /* 0xFC0D_C064 -> 0xFC0D_C067 - Egress Destination MAC Address Low
      */
01230     vudword egdah;              /* 0xFC0D_C068 -> 0xFC0D_C06B - Egress Destination MAC Address High
      */
01231     vudword mcval;              /* 0xFC0D_C06C -> 0xFC0D_C06F - Mirror Count Value
      */
01232     vubyte pack02[16];          /* 0xFC0D_C070 -> 0xFC0D_C07F - RESERVED
      */
01233     vudword mmsr;               /* 0xFC0D_C080 -> 0xFC0D_C083 - Memory Manager Status
      */
01234     vudword lmt;                /* 0xFC0D_C084 -> 0xFC0D_C087 - Low Memory Threshold
      */
01235     vudword lfc;                /* 0xFC0D_C088 -> 0xFC0D_C08B - Lowest Number of Free Cells
      */
01236     vudword pcsr;               /* 0xFC0D_C08C -> 0xFC0D_C08F - Port Congestion Status
      */
01237     vudword iosr;               /* 0xFC0D_C090 -> 0xFC0D_C093 - Switch Input and Output Interface
      Status                */
01238     vudword qwt;                /* 0xFC0D_C094 -> 0xFC0D_C097 - Queue Weights
      */
01239     vubyte pack03[4];           /* 0xFC0D_C098 -> 0xFC0D_C09B - RESERVED
      */
01240     vudword p0bct;              /* 0xFC0D_C09C -> 0xFC0D_C09F - Port 0 Backpressure Congestion
      Threshold               */
01241     vubyte pack04[28];          /* 0xFC0D_C0A0 -> 0xFC0D_C0BB - RESERVED
      */
01242     vudword ffen;               /* 0xFC0D_C0BC -> 0xFC0D_C0BF - Port 0 Forced Forwarding Enable
      */
01243     vudword psnp1;              /* 0xFC0D_C0C0 -> 0xFC0D_C0C3 - Port Snooping Register 1
      */
01244     vudword psnp2;              /* 0xFC0D_C0C4 -> 0xFC0D_C0C7 - Port Snooping Register 2
      */
01245     vudword psnp3;              /* 0xFC0D_C0C8 -> 0xFC0D_C0CB - Port Snooping Register 3
      */
01246     vudword psnp4;              /* 0xFC0D_C0CC -> 0xFC0D_C0CF - Port Snooping Register 4
      */
01247     vudword psnp5;              /* 0xFC0D_C0D0 -> 0xFC0D_C0D3 - Port Snooping Register 5
      */
01248     vudword psnp6;              /* 0xFC0D_C0D4 -> 0xFC0D_C0D7 - Port Snooping Register 6
      */
01249     vudword psnp7;              /* 0xFC0D_C0D8 -> 0xFC0D_C0DB - Port Snooping Register 7
      */
01250     vudword psnp8;              /* 0xFC0D_C0DC -> 0xFC0D_C0DF - Port Snooping Register 8
      */
01251     vudword ipsnp1;             /* 0xFC0D_C0E0 -> 0xFC0D_C0E3 - IP Snooping Register 1
      */
01252     vudword ipsnp2;             /* 0xFC0D_C0E4 -> 0xFC0D_C0E7 - IP Snooping Register 2
      */
01253     vudword ipsnp3;             /* 0xFC0D_C0E8 -> 0xFC0D_C0EB - IP Snooping Register 3
      */
01254     vudword ipsnp4;             /* 0xFC0D_C0EC -> 0xFC0D_C0EF - IP Snooping Register 4
```

```
      */
01255     vudword ipsnp5;             /* 0xFC0D_C0F0 -> 0xFC0D_C0F3 - IP Snooping Register 5
      */
01256     vudword ipsnp6;             /* 0xFC0D_C0F4 -> 0xFC0D_C0F7 - IP Snooping Register 6
      */
01257     vudword ipsnp7;             /* 0xFC0D_C0F8 -> 0xFC0D_C0FB - IP Snooping Register 7
      */
01258     vudword ipsnp8;             /* 0xFC0D_C0FC -> 0xFC0D_C0FF - IP Snooping Register 8
      */
01259     vudword p0vres;             /* 0xFC0D_C100 -> 0xFC0D_C103 - Port 0 VLAN Priority Resolution Map
      */
01260     vudword p1vres;             /* 0xFC0D_C104 -> 0xFC0D_C107 - Port 1 VLAN Priority Resolution Map
      */
01261     vudword p2vres;             /* 0xFC0D_C108 -> 0xFC0D_C10B - Port 2 VLAN Priority Resolution Map
      */
01262     vubyte pack05[52];          /* 0xFC0D_C10C -> 0xFC0D_C13F - RESERVED
      */
01263     vudword ipres;              /* 0xFC0D_C140 -> 0xFC0D_C143 - IPv4/v6 Priority Resolution Table
      */
01264     vubyte pack06[60];          /* 0xFC0D_C144 -> 0xFC0D_C17F - RESERVED
      */
01265     vudword p0res;              /* 0xFC0D_C180 -> 0xFC0D_C183 - Port 0 Priority Resolution
      Configuration                */
01266     vudword p1res;              /* 0xFC0D_C184 -> 0xFC0D_C187 - Port 1 Priority Resolution
      Configuration                */
01267     vudword p2res;              /* 0xFC0D_C188 -> 0xFC0D_C18B - Port 2 Priority Resolution
      Configuration                */
01268     vubyte pack07[116];         /* 0xFC0D_C18C -> 0xFC0D_C1FF - RESERVED
      */
01269     vudword p0id;               /* 0xFC0D_C200 -> 0xFC0D_C203 - Port 0 VLAN ID
      */
01270     vudword p1id;               /* 0xFC0D_C204 -> 0xFC0D_C207 - Port 1 VLAN ID
      */
01271     vudword p2id;               /* 0xFC0D_C208 -> 0xFC0D_C20B - Port 2 VLAN ID
      */
01272     vubyte pack08[116];         /* 0xFC0D_C20C -> 0xFC0D_C27F - RESERVED
      */
01273     vudword vres[32];           /* 0xFC0D_C280 -> 0xFC0D_C2FF - VLAN Domain Resolution Entry 0-31
      */
01274     vudword discn;              /* 0xFC0D_C300 -> 0xFC0D_C303 - Number of Discarded Frames
      */
01275     vudword discb;              /* 0xFC0D_C304 -> 0xFC0D_C307 - Bytes of Discarded Frames
      */
01276     vudword ndiscn;             /* 0xFC0D_C308 -> 0xFC0D_C30B - Number of Non-Discarded Frames
      */
01277     vudword ndiscb;             /* 0xFC0D_C30C -> 0xFC0D_C30F - Bytes of Non-Discarded Frames
      */
01278     vudword p0oqc;              /* 0xFC0D_C310 -> 0xFC0D_C313 - Port 0 Output Queue Congestion
      */
01279     vudword p0mvid;             /* 0xFC0D_C314 -> 0xFC0D_C317 - Port 0 Mismatching VLAN ID
      */
01280     vudword p0mvtag;            /* 0xFC0D_C318 -> 0xFC0D_C31B - Port 0 Missing VLAN Tag
      */
01281     vudword p0bl;               /* 0xFC0D_C31C -> 0xFC0D_C31F - Port 0 Blocked
      */
01282     vudword p1oqc;              /* 0xFC0D_C320 -> 0xFC0D_C323 - Port 1 Output Queue Congestion
      */
01283     vudword p1mvid;             /* 0xFC0D_C324 -> 0xFC0D_C327 - Port 1 Mismatching VLAN ID
      */
01284     vudword p1mvtag;            /* 0xFC0D_C328 -> 0xFC0D_C32B - Port 1 Missing VLAN Tag
      */
01285     vudword p1bl;               /* 0xFC0D_C32C -> 0xFC0D_C32F - Port 1 Blocked
      */
01286     vudword p2oqc;              /* 0xFC0D_C330 -> 0xFC0D_C333 - Port 2 Output Queue Congestion
      */
01287     vudword p2mvid;             /* 0xFC0D_C334 -> 0xFC0D_C337 - Port 2 Mismatching VLAN ID
      */
01288     vudword p2mvtag;            /* 0xFC0D_C338 -> 0xFC0D_C33B - Port 2 Missing VLAN Tag
      */
01289     vudword p2bl;               /* 0xFC0D_C33C -> 0xFC0D_C33F - Port 2 Blocked
      */
01290     vubyte pack09[192];         /* 0xFC0D_C340 -> 0xFC0D_C3FF - RESERVED
      */
01291     vudword isr;                /* 0xFC0D_C400 -> 0xFC0D_C403 - Interrupt Status Register
      */
01292     vudword imr;                /* 0xFC0D_C404 -> 0xFC0D_C407 - Interrupt Mask Register
      */
01293     vudword rdsr;               /* 0xFC0D_C408 -> 0xFC0D_C40B - Receive Descriptor Ring Pointer
      */
01294     vudword tdsr;               /* 0xFC0D_C40C -> 0xFC0D_C40F - Transmit Descriptor Ring Pointer
      */
01295     vudword mrbr;               /* 0xFC0D_C410 -> 0xFC0D_C413 - Maximum Receive Buffer Size
      */
01296     vudword rdar;               /* 0xFC0D_C414 -> 0xFC0D_C417 - Receive Descriptor Active
      */
01297     vudword tdar;               /* 0xFC0D_C418 -> 0xFC0D_C41B - Transmit Descriptor Active
      */
```

```
01298     vubyte pack10[228];         /* 0xFC0D_C41C -> 0xFC0D_C4FF - RESERVED
      */
01299     vudword lrec0;              /* 0xFC0D_C500 -> 0xFC0D_C503 - Learning Records A0 & B1
      */
01300     vudword lrec1;              /* 0xFC0D_C504 -> 0xFC0D_C507 - Learning Record B1
      */
01301     vudword lsr;                /* 0xFC0D_C508 -> 0xFC0D_C50B - Learning Data Available Status
      */
01302     vubyte pack11[15092];       /* 0xFC0D_C50C -> 0xFC0D_FFFF - RESERVED
      */
01303     mactabentry mactable[2048]; /* 0xFC0E_0000 -> 0xFC0E_3FFF - MAC Address Lookup Table
      */
01304 } eswstruct;
01305
01306 /*
01307  * NAND FLASH CONTROLLER
01308  */
01309 typedef struct
01310 {
01311     vudword sramb0[576]; /* 0xFC0F_C000 -> 0xFC0F_C8FF - SRAM Buffer 0
      */
01312     vubyte pack00[1792]; /* 0xFC0F_C900 -> 0xFC0F_CFFF - RESERVED
      */
01313     vudword sramb1[576]; /* 0xFC0F_D000 -> 0xFC0F_D8FF - SRAM Buffer 1
      */
01314     vubyte pack01[1792]; /* 0xFC0F_D900 -> 0xFC0F_DFFF - RESERVED
      */
01315     vudword sramb2[576]; /* 0xFC0F_E000 -> 0xFC0F_E8FF - SRAM Buffer 2
      */
01316     vubyte pack02[1792]; /* 0xFC0F_E900 -> 0xFC0F_EFFF - RESERVED
      */
01317     vudword sramb3[576]; /* 0xFC0F_F000 -> 0xFC0F_F8FF - SRAM Buffer 3
      */
01318     vubyte pack03[1536]; /* 0xFC0F_F900 -> 0xFC0F_FEFF - RESERVED
      */
01319     vudword cmd1;        /* 0xFC0F_FF00 -> 0xFC0F_FF03 - Flash Command 1
      */
01320     vudword cmd2;        /* 0xFC0F_FF04 -> 0xFC0F_FF07 - Flash Command 2
      */
01321     vudword car;         /* 0xFC0F_FF08 -> 0xFC0F_FF0B - Column Address
      */
01322     vudword rar;         /* 0xFC0F_FF0C -> 0xFC0F_FF0F - Row Address
      */
01323     vudword rpt;         /* 0xFC0F_FF10 -> 0xFC0F_FF13 - Flash Command Repeat
      */
01324     vudword rai;         /* 0xFC0F_FF14 -> 0xFC0F_FF17 - Row Address Increment
      */
01325     vudword sr1;         /* 0xFC0F_FF18 -> 0xFC0F_FF1B - Flash Status 1
      */
01326     vudword sr2;         /* 0xFC0F_FF1C -> 0xFC0F_FF1F - Flash Status 2
      */
01327     vudword dma1;        /* 0xFC0F_FF20 -> 0xFC0F_FF23 - DMA 1 Address Register
      */
01328     vudword dmacfg;      /* 0xFC0F_FF24 -> 0xFC0F_FF27 - DMA Configuration Register
      */
01329     vudword swap;        /* 0xFC0F_FF28 -> 0xFC0F_FF2B - Cache Swap Register
      */
01330     vudword secsz;       /* 0xFC0F_FF2C -> 0xFC0F_FF2F - Sector Size Register
      */
01331     vudword cfg;         /* 0xFC0F_FF30 -> 0xFC0F_FF33 - Flash Configuration Register
      */
01332     vudword dma2;        /* 0xFC0F_FF34 -> 0xFC0F_FF37 - DMA 2 Address Register
      */
01333     vudword isr;         /* 0xFC0F_FF38 -> 0xFC0F_FF3B - Interrupt Status Register
      */
01334 } nfcstruct;
01335
01336 /*
01337  * MCF54418 MAIN STRUCT 1
01338  */
01339 typedef struct
01340 {
01341     owstruct ow; /* 0xEC00_8000 -> 0xEC00_801B - 1-Wire Module                              */
01342
01343     vubyte pack01[32740]; /* 0xEC00_801C -> 0xEC00_FFFF
      */
01344
01345     i2cstruct i2c25[4]; /* 0xEC01_0000 -> 0xEC01_FFFF - I2C Module 2-5                           */
01346
01347     vubyte pack02[98304]; /* 0xEC02_0000 -> 0xEC03_7FFF
      */
01348
01349     dspistruct dspi2; /* 0xEC03_8000 -> 0xEC03_80BB - DMA Serial Peripheral Interface 2        */
01350
01351     vubyte pack03[16196]; /* 0xEC03_80BC -> 0xEC03_BFFF
      */
01352
```

```
01353     dspistruct dspi3; /* 0xEC03_C000 -> 0xEC03_C0BB - DMA Serial Peripheral Interface 3           */
01354
01355     vubyte pack04[147268]; /* 0xEC03_C0BC -> 0xEC05_FFFF
      */
01356
01357     uartstruct uarts[6]; /* 0xEC06_0000 -> 0xEC07_7FFF - UART Module 4-9
      */
01358
01359     vubyte pack05[65536]; /* 0xEC07_8000 -> 0xEC08_7FFF
      */
01360
01361     mcpwmstruct mcpwm; /* 0xEC08_8000 -> 0xEC08_8153 - Motor Control Pulse-Width Modulator         */
01362
01363     vubyte pack06[32428]; /* 0xEC08_8154 -> 0xEC08_FFFF
      */
01364
01365     resetstruct reset; /* 0xEC09_0000 -> 0xEC09_0003 - Reset Controller                             */
01366
01367     ccmstruct ccm; /* 0xEC09_0004 -> 0xEC09_0027 - Chip Configuration Module                    */
01368
01369     vubyte pack07[16344]; /* 0xEC09_0028 -> 0xEC09_3FFF
      */
01370
01371     gpiostruct gpio; /* 0xEC09_4000 -> 0xEC09_4087 - Pin-Multiplexing and Control                 */
01372
01373 } mcf54418_1;
01374
01375 /*
01376  * MCF54418 MAIN STRUCT 2
01377  */
01378 typedef struct
01379 {
01380     xbsstruct xbs; /* 0xFC00_4000 -> 0xFC00_4713 - Crossbar Switch (XBS)                        */
01381
01382     vubyte pack09[14572]; /* 0xFC00_4714 -> 0xFC00_7FFF
      */
01383
01384     csstruct cs[6]; /* 0xFC00_8000 -> 0xFC00_8047 - Chip Select 0-5                               */
01385
01386     vubyte pack10[98232]; /* 0xFC00_8048 -> 0xFC01_FFFF
      */
01387
01388     canstruct can[2]; /* 0xFC02_0000 -> 0xFC02_7FFF - Controller Area Network (FlexCAN) 0-1       */
01389
01390     vubyte pack11[65536]; /* 0xFC02_8000 -> 0xFC03_7FFF
      */
01391
01392     i2cstruct i2c1; /* 0xFC03_8000 -> 0xFC03_8013 - I2C Module 1                                  */
01393
01394     //   vubyte           pack12[16364];      /* 0xFC03_8014 -> 0xFC03_BFFF
      */
01395
01396     dspistruct dspi1; /* 0xFC03_C000 -> 0xFC03_C0BB - DMA Serial Peripheral Interface 1           */
01397
01398     vubyte pack13[16212]; /* 0xFC03_C0BC -> 0xFC04_000F
      */
01399
01400     scmstruct scm; /* 0xFC04_0010 -> 0xFC04_007F - System Control Module and Power Management  */
01401
01402     vubyte pack14[16256]; /* 0xFC04_0080 -> 0xFC04_3FFF
      */
01403
01404     edmastruct edma; /* 0xFC04_4000 -> 0xFC04_57FF - Enhanced Direct Memory Access Controller    */
01405
01406     vubyte pack15[10240]; /* 0xFC04_5800 -> 0xFC04_7FFF
      */
01407
01408     intcstruct intc[3]; /* 0xFC04_8000 -> 0xFC05_3FFF - Interrupt Controller 0-2                    */
01409
01410     vubyte pack16[224]; /* 0xFC05_4000 -> 0xFC05_40DF                                              */
01411
01412     intc_iackstruct intc_iack; /* 0xFC05_40E0 -> 0xFC05_40FF - Global Interrupt Acknowledge Cycles
      */
01413
01414     vubyte pack17[16128]; /* 0xFC05_4100 -> 0xFC05_7FFF
      */
01415
01416     i2cstruct i2c0; /* 0xFC05_8000 -> 0xFC05_8013 - I2C Module 0                                  */
01417
01418     //   vubyte           pack18[16364];      /* 0xFC05_8014 -> 0xFC05_BFFF
      */
01419
01420     dspistruct dspi0; /* 0xFC05_C000 -> 0xFC05_C0BB - DMA Serial Peripheral Interface 0           */
01421
01422     vubyte pack19[16196]; /* 0xFC05_C0BC -> 0xFC05_FFFF
      */
01423
```

```
01424     uartstruct uarts[4]; /* 0xFC06_0000 -> 0xFC06_FFFF - UART Module 0-3
      */
01425
01426     timerstruct timer[4]; /* 0xFC07_0000 -> 0xFC07_FFFF - DMA Timer Module 0-3
      */
01427
01428     pitstruct pit[4]; /* 0xFC08_0000 -> 0xFC08_FFFF - Programmable Interrupt Timer Module 0-3     */
01429
01430     eportstruct eport; /* 0xFC09_0000 -> 0xFC09_0007 - Edge Port Module                            */
01431
01432     vubyte pack20[16376]; /* 0xFC09_0008 -> 0xFC09_3FFF
      */
01433
01434     adcstruct adc; /* 0xFC09_4000 -> 0xFC09_405B - Analog-to-Digital Converter                 */
01435
01436     vubyte pack21[16292]; /* 0xFC09_405C -> 0xFC09_7FFF
      */
01437
01438     dacstruct dac[2]; /* 0xFC09_8000 -> 0xFC09_FFFF - Digital-to-Analog Converter                */
01439
01440     vubyte pack22[32]; /* 0xFC0A_0000 -> 0xFC0A_001F                                             */
01441
01442     sbfstruct sbf; /* 0xFC0A_0020 -> 0xFC0A_0023 - Serial Boot Facility                         */
01443
01444     vubyte pack23[32732]; /* 0xFC0A_0024 -> 0xFC0A_7FFF
      */
01445
01446     rtcstruct rtc; /* 0xFC0A_8000 -> 0xFC0A_883F - Real-Time Clock                              */
01447
01448     vubyte pack24[14272]; /* 0xFC0A_8840 -> 0xFC0A_BFFF
      */
01449
01450     simstruct sim; /* 0xFC0A_C000 -> 0xFC0A_C07B - Subscriber Identification Module            */
01451
01452     vubyte pack25[16260]; /* 0xFC0A_C07C -> 0xFC0A_FFFF
      */
01453
01454     usb_otgstruct usb_otg; /* 0xFC0B_0000 -> 0xFC0B_01CF - USB On-the-Go
      */
01455
01456     vubyte pack26[15920]; /* 0xFC0B_01D0 -> 0xFC0B_3FFF
      */
01457
01458     usb_hoststruct usb_host; /* 0xFC0B_4000 -> 0xFC0B_41AB - USB Host Controller
      */
01459
01460     vubyte pack27[15956]; /* 0xFC0B_41AC -> 0xFC0B_7FFF
      */
01461
01462     ddrmcstruct ddrmc; /* 0xFC0B_8000 -> 0xFC0B_81AF - DDR1/2 SDRAM Memory Controller              */
01463
01464     vubyte pack28[15952]; /* 0xFC0B_81B0 -> 0xFC0B_BFFF
      */
01465
01466     ssi0struct ssi0; /* 0xFC0B_C000 -> 0xFC0B_C05B - Synchronous Serial Interface 0              */
01467
01468     vubyte pack29[16292]; /* 0xFC0B_C05C -> 0xFC0B_FFFF
      */
01469
01470     clockstruct clock; /* 0xFC0C_0000 -> 0xFC0C_000B - Clock Module (Phase-Locked Loop)           */
01471
01472     vubyte pack30[16372]; /* 0xFC0C_000C -> 0xFC0C_3FFF
      */
01473
01474     rngstruct rng; /* 0xFC0C_4000 -> 0xFC0C_401B - Random Number Generator                      */
01475
01476     vubyte pack31[16356]; /* 0xFC0C_401C -> 0xFC0C_7FFF
      */
01477
01478     ssi1struct ssi1; /* 0xFC0C_8000 -> 0xFC0C_805B - Synchronous Serial Interface 1              */
01479
01480     vubyte pack32[16292]; /* 0xFC0C_805C -> 0xFC0C_BFFF
      */
01481
01482     sdhcstruct sdhc; /* 0xFC0C_C000 -> 0xFC0C_C0FF - Enhanced Secure Digital Host Controller     */
01483
01484     vubyte pack33[32512]; /* 0xFC0C_C100 -> 0xFC0D_3FFF
      */
01485
01486     fecstruct fec[2]; /* 0xFC0D_4000 -> 0xFC0D_BFFF - 10/100 Mbps Ethernet MAC-NET Core 0-1       */
01487
01488     eswstruct esw; /* 0xFC0D_C000 -> 0xFC0E_3FFF - Ethernet Switch                              */
01489
01490     vubyte pack36[98304]; /* 0xFC0E_4000 -> 0xFC0F_BFFF
      */
01491
01492     nfcstruct nfc; /* 0xFC0F_C000 -> 0xFC0F_FF3B - NAND Flash Controller                        */
```

```
01493
01494 } mcf54418_2;
01495
01496 extern vudword flash_mirror[0x80000];
01497 extern volatile rgpiostruct sim_rgpio;
01498 extern volatile mcf54418_1 sim1;
01499 extern volatile mcf54418_2 sim2;
01500
01501 typedef struct
01502 {
01503     unsigned long table[256];
01504 } vectors;
01505
01506 extern vectors vector_base;
01507
01508 typedef struct
01509 {
01510     unsigned short flags;
01511     unsigned short length;
01512     unsigned long address;
01513 } Legacy_EtherBD;
01514
01515 typedef struct
01516 {
01517     unsigned short flags;
01518     unsigned short length;
01519     unsigned long address;
01520     unsigned long flags2;
01521     unsigned short hLenAndProt;
01522     unsigned short payCSum;
01523     unsigned long bdu;
01524     unsigned long _1588_ts;
01525     unsigned long pad[2];
01526 } Enhanced_EtherBD;
01527
01528 #ifdef ENHANCED_ETHER_BD
01529 typedef Enhanced_EtherBD EtherBD;
01530 #else
01531 typedef Legacy_EtherBD EtherBD;
01532 #endif
01533
01534 #define RXBD_Flag_Empty (0x8000)
01535 #define RXBD_Flag_SW1b (0x4000)
01536 #define RXBD_Flag_Wrap (0x2000)
01537 #define RXBD_Flag_SW2b (0x1000)
01538 #define RXBD_Flag_Last (0x0800)
01539 #define RXBD_Flag_Miss (0x0100)
01540 #define RXBD_Flag_BroadCast (0x0080)
01541 #define RXBD_Flag_MultiCast (0x0040)
01542 #define RXBD_Flag_LengthErr (0x0020)
01543 #define RXBD_Flag_Align_Err (0x0010)
01544 #define RXBD_Flag_ShortErr (0x0008)
01545 #define RXBD_Flag_CRC_Err (0x0004)
01546 #define RXBD_Flag_OverErr (0x0002)
01547 #define RXBD_Flag_TruncErr (0x0001)
01548 #define RXBD_Error_Mask \
01549     (RXBD_Flag_LengthErr | RXBD_Flag_Align_Err | RXBD_Flag_ShortErr | RXBD_Flag_CRC_Err |
      RXBD_Flag_OverErr | RXBD_Flag_TruncErr)
01550 #define ERXBD_Flag2_MAC_Err (0x80000000)
01551 #define ERXBD_Flag2_PHY_Err (0x04000000)
01552 #define ERXBD_Flag2_Collision (0x02000000)
01553 #define ERXBD_Flag2_Unicast (0x01000000)
01554 #define ERXBD_Flag2_IRQ (0x00800000)
01555 #define ERXBD_Flag2_IP_CSum (0x00000020)
01556 #define ERXBD_Flag2_Prot_CSum (0x00000010)
01557 #define ERXBD_Flag2_VLAN (0x00000004)
01558 #define ERXBD_Flag2_IPv6 (0x00000002)
01559 #define ERXBD_Flag2_Frag (0x00000001)
01560
01561 #define TXBD_Flag_Ready (0x8000)
01562 #define TXBD_Flag_SW1b (0x4000)
01563 #define TXBD_Flag_Wrap (0x2000)
01564 #define TXBD_Flag_SW2b (0x1000)
01565 #define TXBD_Flag_Last (0x0800)
01566 #define TXBD_Flag_TxCRC (0x0400)
01567 #define TXBD_Flag_Defered (0x0200)
01568 #define TXBD_Flag_HB_Error (0x0100)
01569 #define TXBD_Flag_LC_Error (0x0080)
01570 #define TXBD_Flag_RT_Error (0x0040)
01571 #define TXBD_Flag_RC_B3 (0x0020)
01572 #define TXBD_Flag_RC_B2 (0x0010)
01573 #define TXBD_Flag_RC_B1 (0x0008)
01574 #define TXBD_Flag_RC_B0 (0x0004)
01575 #define TXBD_Flag_UN_Error (0x0002)
01576 #define TXBD_Flag_CSL_Error (0x0001)
01577 #define TXBD_Error_Mask (TXBD_Flag_UN_Error | TXBD_Flag_RT_Error | TXBD_Flag_LC_Error |
      TXBD_Flag_HB_Error)
```

```
01578 #define TXBD_Flag_NormalSend (TXBD_Flag_TxCRC | TXBD_Flag_Ready | TXBD_Flag_Last)
01579 #define ETXBD_Flag2_IRQ (0x40000000)
01580 #define ETXBD_Flag2_TS_RQ (0x20000000)
01581 #define ETXBD_Flag2_Insert_Prot (0x10000000)
01582 #define ETXBD_Flag2_Insert_IP (0x08000000)
01583 #define ETXBD_Flag2_TX_Err (0x00008000)
01584 #define ETXBD_Flag2_UFlow_Err (0x00002000)
01585 #define ETXBD_Flag2_Coll_Err (0x00001000)
01586 #define ETXBD_Flag2_Frame_Err (0x00008000)
01587 #define ETXBD_Flag2_Late_Coll (0x00004000)
01588 #define ETXBD_Flag2_OFlow_Err (0x00002000)
01589 #define ETXBD_Flag2_TS_Err (0x00001000)
01590
01591 #define FEC_ISR_MASK_HBERR (0x80000000)
01592 #define FEC_ISR_MASK_BABR (0x40000000)
01593 #define FEC_ISR_MASK_BABT (0x20000000)
01594 #define FEC_ISR_MASK_GRA (0x10000000)
01595 #define FEC_ISR_MASK_TXF (0x08000000)
01596 #define FEC_ISR_MASK_TXB (0x04000000)
01597 #define FEC_ISR_MASK_RXF (0x02000000)
01598 #define FEC_ISR_MASK_RXB (0x01000000)
01599 #define FEC_ISR_MASK_MII (0x00800000)
01600 #define FEC_ISR_MASK_EBERR (0x00400000)
01601 #define FEC_ISR_MASK_LC (0x00200000)
01602 #define FEC_ISR_MASK_RL (0x00100000)
01603 #define FEC_ISR_MASK_UN (0x00080000)
01604
01605 #define ESW_ISR_MASK_RXF (0x00000004)
01606 #define ESW_ISR_MASK_RXB (0x00000002)
01607 #define ESW_ISR_MASK_TXF (0x00000010)
01608 #define ESW_ISR_MASK_TXB (0x00000008)
01609 #define ESW_ISR_MASK_LRN (0x00000200)
01610
01611 #endif /* _SIM54418_H_ */
```

24.30 cfinter.h File Reference

ColdFire Interrupt Macro.

24.30.1 Detailed Description

ColdFire Interrupt Macro.

24.31 cfinter.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00031 #ifndef _COLDFIRE_INTERRUPT_H
00032 #define _COLDFIRE_INTERRUPT_H
00033
00034 #ifdef __cplusplus
00035
00036 #ifdef _NBRTOS_H
00037
00055 #define INTERRUPT(x, y)                                  \
00056     extern "C"                                           \
00057     {                                                    \
00058         void real_##x();                                 \
00059         void x();                                        \
00060     }                                                    \
00061     void fake_##x()                                      \
00062     {                                                    \
00063         __asm__(".global " #x);                          \
00064         __asm__(#x ":");                                 \
00065         __asm__("move.w #0x2700,%sr ");                  \
00066         __asm__("lea      -60(%a7),%a7 ");               \
00067         __asm__("movem.l  %d0-%d7/%a0-%a6,(%a7) ");      \
00068         __asm__("move.l (OSISRLevel32),%d0 ");           \
00069         __asm__("move.l %d0,-(%sp) ");                   \
00070         __asm__("move.l (OSIntNesting),%d0");            \
00071         __asm__("addq.l #1,%d0");                        \
00072         __asm__("move.l %d0,(OSIntNesting)");            \
00073         __asm__("move.l #" #y ",%d0 ");                  \
00074         __asm__("move.w %d0,%sr ");                      \
00075         __asm__("move.l %d0,(OSISRLevel32)");            \
00076         __asm__("jsr real_" #x);                         \
```

```
00077         __asm__("move.l (%sp)+,%d0 ");              \
00078         __asm__("move.l %d0,(OSISRLevel32)");       \
00079         __asm__(" jsr      OSIntExit  ");           \
00080         __asm__("movem.l  (%a7),%d0-%d7/%a0-%a6 "); \
00081         __asm__("lea    60(%a7),%a7 ");             \
00082         __asm__("rte");                             \
00083     }                                               \
00084     void real_##x()
00085
00086 #else
00087 #error NBRTOS must be included
00088 #endif
00089
00090 #else /*Do C version */
00091
00092 #ifdef _NBRTOS_H
00093 #define INTERRUPT(x, y)                             \
00094     void x();                                       \
00095     void fake_##x()                                 \
00096     {                                               \
00097         __asm__(".global " #x);                     \
00098         __asm__(#x ":");                            \
00099         __asm__("move.w #0x2700,%sr ");             \
00100         __asm__("lea     -60(%a7),%a7 ");           \
00101         __asm__("movem.l %d0-%d7/%a0-%a6,(%a7) ");  \
00102         __asm__("move.l (OSISRLevel32),%d0 ");      \
00103         __asm__("move.l %d0,-(%sp) ");              \
00104         __asm__("move.l (OSIntNesting),%d0");       \
00105         __asm__("addq.l #1,%d0");                   \
00106         __asm__("move.l %d0,(OSIntNesting)");       \
00107         __asm__("move.l #" #y ",%d0 ");             \
00108         __asm__("move.w %d0,%sr ");                 \
00109         __asm__("move.l %d0,(OSISRLevel32)");       \
00110         __asm__("jsr real_" #x);                    \
00111         __asm__("move.l (%sp)+,%d0 ");              \
00112         __asm__("move.l %d0,(OSISRLevel32)");       \
00113         __asm__(" jsr      OSIntExit  ");           \
00114         __asm__("movem.l  (%a7),%d0-%d7/%a0-%a6 "); \
00115         __asm__("lea    60(%a7),%a7 ");             \
00116         __asm__("rte");                             \
00117     }                                               \
00118     void real_##x()
00119 #else
00120 #error NBRTOS must be included
00121 #endif
00122 #endif /*C Version */
00123
00124 /*-------------------------------------------------------------------------------------------------------
00125  * The following definitions define the vector offset for various predefined vectors on the coldfire.
00126
    *-------------------------------------------------------------------------------------------------------*/
00127 #define CF_ACCESSERROR_VECTOR 2
00128 #define CF_ADDRESSERROR_VECTOR 3
00129 #define CF_ILLEGAL_INSTRUCTION_VECTOR 4
00130 #define CF_PRIVLEDGE_VIOLATION_VECTOR 8
00131 #define CF_TRACE_VECTOR 9
00132 #define CF_UNIMPLMENTED_A_VECTOR 10
00133 #define CF_UNIMPLMENTED_F_VECTOR 11
00134 #define CF_DEBUG_INTERRUPT 12
00135 #define CF_FORMAT_ERR_VECTOR 14
00136 #define CF_UNINITIALIZED_VECTOR 15
00137 #define CF_SPURIOUS_INT_VECTOR 24
00138 #define CF_AUTOVECTOR_IRQ1 25
00139 #define CF_AUTOVECTOR_IRQ2 26
00140 #define CF_AUTOVECTOR_IRQ3 27
00141 #define CF_AUTOVECTOR_IRQ4 28
00142 #define CF_AUTOVECTOR_IRQ5 29
00143 #define CF_AUTOVECTOR_IRQ6 30
00144 #define CF_AUTOVECTOR_IRQ7 31
00145 #define CF_TRAP0_VECTOR 32
00146 #define CF_TRAP1_VECTOR 33
00147 #define CF_TRAP2_VECTOR 34
00148 #define CF_TRAP3_VECTOR 35
00149 #define CF_TRAP4_VECTOR 36
00150 #define CF_TRAP5_VECTOR 37
00151 #define CF_TRAP6_VECTOR 38
00152 #define CF_TRAP7_VECTOR 39
00153 #define CF_TRAP8_VECTOR 40
00154 #define CF_TRAP9_VECTOR 41
00155 #define CF_TRAP10_VECTOR 42
00156 #define CF_TRAP11_VECTOR 43
00157 #define CF_TRAP12_VECTOR 44
00158 #define CF_TRAP13_VECTOR 45
00159 #define CF_TRAP14_VECTOR 46
00160 #define CF_TRAP15_VECTOR 47
00161 #define CF_USER_BASE_VECTOR 64
00162
```

```
00163 extern "C"
00164 {
00165     /*-------------------------------------------------------------
00166      * This gets the current interrupt mask level in the SR register.
00167      * You must have supervisor privileges to use this macro.
00168      *-------------------------------------------------------------*/
00169     uint16_t GetSR_IntLevel();
00170
00171     /*-------------------------------------------------------------
00172      * The following macro sets the interrupt mask level in the SR.
00173      * You must have supervisor privileges to use this macro.
00174      *-------------------------------------------------------------*/
00175     void SetSR_IntLevel(uint16_t sv);
00176 }
00177
00178 #endif
00179
```

24.32 arch/coldfire/include/PlatformHeader.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NB_PLATFORM_H
00006 #define NB_PLATFORM_H
00007
00008 #include <basictypes.h>
00009
00010 #define NBUP_MSFUS_XTRA_RECORD_PAD (16)
00011 extern const int EXTRA_RECORD_PAD;
00012 struct PlatformFlashHeaderStruct
00013 {
00014     // these values will be in the correct order when transmit in the image file
00015     // they do not need to be converted between endianess
00016     uint32_t U32BlockRamStart;
00017     uint32_t U32ExecutionAddr;
00018     uint32_t U32BlockSize;
00019     uint32_t U32SourceBlockSize;
00020     uint32_t U32BlockSum;
00021     uint32_t U32StructSum;
00022     uint8_t ExtraData[NBUP_MSFUS_XTRA_RECORD_PAD];
00023     bool VerifyCorrect();
00024     uint32_t SizeWithoutPad() { return sizeof(PlatformFlashHeaderStruct) - NBUP_MSFUS_XTRA_RECORD_PAD;
    };
00025     uint32_t CompleteRecordSize() { return SizeWithoutPad() + U32SourceBlockSize; };
00026 } __attribute__((packed));
00027
00028 #endif
```

24.33 platform/MODM7AE70/include/PlatformHeader.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NB_PLATFORM_H
00006 #define NB_PLATFORM_H
00007 #include <basictypes.h>
00008
00009 #define NBUP_MSFUS_XTRA_RECORD_PAD (16)
00010 extern const int EXTRA_RECORD_PAD;
00011 struct PlatformFlashHeaderStruct
00012 {
00013     // these values will be in the correct order when transmit in the image file
00014     // they do not need to be converted between endianess
00015     uint32_t U32BlockRamStart;
00016     uint32_t U32ExecutionAddr;
00017     uint32_t U32BlockSize;
00018     uint32_t U32SourceBlockSize;
00019     uint32_t U32BlockSum;
00020     uint32_t U32StructSum;
00021     uint8_t ExtraData[NBUP_MSFUS_XTRA_RECORD_PAD];
00022     bool VerifyCorrect();
00023     uint32_t SizeWithoutPad() { return sizeof(PlatformFlashHeaderStruct) - NBUP_MSFUS_XTRA_RECORD_PAD;
    };
00024     uint32_t CompleteRecordSize() { return SizeWithoutPad() + U32SourceBlockSize; };
00025 } __attribute__((packed));
00026
00027 #endif
```

24.34 platform/SBE70LC/include/PlatformHeader.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NB_PLATFORM_H
00006 #define NB_PLATFORM_H
00007
00008 #define NBUP_MSFUS_XTRA_RECORD_PAD (16)
00009 extern const int EXTRA_RECORD_PAD;
00010 struct PlatformFlashHeaderStruct
00011 {
00012     // these values will be in the correct order when transmit in the image file
00013     // they do not need to be converted between endianess
00014     uint32_t U32BlockRamStart;
00015     uint32_t U32ExecutionAddr;
00016     uint32_t U32BlockSize;
00017     uint32_t U32SourceBlockSize;
00018     uint32_t U32BlockSum;
00019     uint32_t U32StructSum;
00020     uint8_t ExtraData[NBUP_MSFUS_XTRA_RECORD_PAD];
00021     bool VerifyCorrect();
00022     uint32_t SizeWithoutPad() { return sizeof(PlatformFlashHeaderStruct) - NBUP_MSFUS_XTRA_RECORD_PAD;
    };
00023     uint32_t CompleteRecordSize() { return SizeWithoutPad() + U32SourceBlockSize; };
00024 } __attribute__((packed));
00025
00026 #endif
```

24.35 platform/SOMRT1061/include/PlatformHeader.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef NB_PLATFORM_H
00005 #define NB_PLATFORM_H
00006
00007 #include <basictypes.h>
00008
00009 enum eCompress_t : uint8_t {
00010     eComp_None      = 0,
00011     eComp_ZSTD      = 1,
00012     eComp_LZMA      = 2,
00013     eComp_DEFLATE   = 3,
00014     eComp_LZ4       = 4,
00015 };
00016
00017 enum eDigest_t : uint8_t {
00018     eDigest_CSUM16  = 1,
00019     eDigest_CSUM32  = 2,
00020     eDigest_MD5     = 3,
00021     eDigest_SHA1    = 4,
00022     eDigest_SHA256  = 5,
00023     eDigest_SHA384  = 6,
00024     eDigest_SHA512  = 7,
00025     eDigest_XXH32   = 8,
00026     eDigest_XXH64   = 9,
00027     eDigest_None    = 0xFE,
00028     eDigest_MAXLEN  = 0xFF
00029 };
00030
00031 #define HDR_DIGEST_MAX_LEN 64
00032 constexpr uint32_t GetDigestLen(eDigest_t eDigest)
00033 {
00034     switch (eDigest) {
00035         default:                return 0;
00036         case eDigest_None:      return 0;
00037         case eDigest_CSUM16:    return 2;
00038         case eDigest_CSUM32:    return 4;
00039         case eDigest_MD5:       return 16;
00040         case eDigest_SHA1:      return 20;
00041         case eDigest_SHA256:    return 32;
00042         case eDigest_SHA384:    return 48;
00043         case eDigest_SHA512:    return 64;
00044         case eDigest_XXH32:     return 4;
00045         case eDigest_XXH64:     return 8;
00046
00047         case eDigest_MAXLEN:    return HDR_DIGEST_MAX_LEN;
00048     }
00049 }
00050
00051 struct StartUpStruct
00052 {
```

```
00053     uint32_t    dwBlockRamStart ;  //Where the code is to be loaded
00054     uint32_t    dwExecutionAddr ;  //Where execution is supposed to start
00055     uint32_t    dwBlockSize     ;  //Decompressed size
00056     uint32_t    dwSrcBlockSize  ;  //size of compressed source
00057     uint32_t    dwSrcBlockOffset;  //Offset from begining of this structure to find source
00058     eCompress_t eCompression    ;  //Type of compression
00059     eDigest_t   eSrcBlockDigest ;  //Digest of the compressed source image
00060     eDigest_t   eStructDigest   ;  //digest of this structure
00061     uint8_t     _reserved1      ;  //Padding...
00062     uint8_t     digestData[]    ;  //Digest data, structurefirst then Src digest
00063
00064     inline uint8_t *GetSrcDigest()     { return digestData; }
00065     inline uint32_t GetSrcDigestLen()  { return GetDigestLen(eSrcBlockDigest); }
00066     inline uint8_t *GetHdrDigest()     { return GetSrcDigest() + GetSrcDigestLen(); }
00067     inline uint32_t GetHdrDigestLen()  { return GetDigestLen(eStructDigest); }
00068     inline uint32_t GetHdrSize()
00069     { return sizeof(StartUpStruct) + GetSrcDigestLen() + GetHdrDigestLen(); }
00070     constexpr static inline uint32_t MaxSize()
00071     { return sizeof(StartUpStruct) + 2*GetDigestLen(eDigest_MAXLEN); }
00072     bool            HdrValid();
00073     bool            SrcValid();
00074 }__attribute__((packed));
00075
00076
00077 #define NBUP_MSFUS_XTRA_RECORD_PAD (16)
00078 extern const int EXTRA_RECORD_PAD;
00079 struct PlatformFlashHeaderStruct
00080 {
00081     // these values will be in the correct order when transmit in the image file
00082     // they do not need to be converted between endianess
00083     uint32_t U32BlockRamStart ;
00084     uint32_t U32ExecutionAddr ;
00085     uint32_t U32BlockSize     ;
00086     uint32_t U32SourceBlockSize;
00087     uint32_t U32BlockSum      ;
00088     uint32_t U32StructSum     ;
00089     uint8_t ExtraData[128+NBUP_MSFUS_XTRA_RECORD_PAD];
00090     bool     VerifyCorrect();
00091     uint32_t SizeWithoutPad()     { return
      sizeof(PlatformFlashHeaderStruct)-NBUP_MSFUS_XTRA_RECORD_PAD; };
00092     uint32_t CompleteRecordSize();
00093 }__attribute__((packed));
00094 #endif
```

24.36 cm_core_config.h File Reference

Macros

- #define __CM7_REV 0x0000

  Configuration of the Cortex-M7 Processor and Core Peripherals.
- #define __MPU_PRESENT 1
- #define __NVIC_PRIO_BITS 3
- #define __FPU_PRESENT 1
- #define __FPU_DP 1
- #define __ICACHE_PRESENT 1
- #define __DCACHE_PRESENT 1
- #define __DTCM_PRESENT 1
- #define __ITCM_PRESENT 1
- #define __Vendor_SysTickConfig 0

Typedefs

- typedef enum IRQn IRQn_Type

  Interrupt Number Definitions.

Enumerations

- enum IRQn {
  NonMaskableInt_IRQn = -14 , HardFault_IRQn = -13 , MemoryManagement_IRQn = -12 , BusFault_IRQn = -11 ,
  UsageFault_IRQn = -10 , SVCall_IRQn = -5 , DebugMonitor_IRQn = -4 , PendSV_IRQn = -2 ,

SysTick_IRQn = -1 , SUPC_IRQn = 0 , RSTC_IRQn = 1 , RTC_IRQn = 2 ,
RTT_IRQn = 3 , WDT_IRQn = 4 , PMC_IRQn = 5 , EFC_IRQn = 6 ,
UART0_IRQn = 7 , UART1_IRQn = 8 , PIOA_IRQn = 10 , PIOB_IRQn = 11 ,
PIOC_IRQn = 12 , USART0_IRQn = 13 , USART1_IRQn = 14 , USART2_IRQn = 15 ,
PIOD_IRQn = 16 , PIOE_IRQn = 17 , HSMCI_IRQn = 18 , TWIHS0_IRQn = 19 ,
TWIHS1_IRQn = 20 , SPI0_IRQn = 21 , SSC_IRQn = 22 , TC0_IRQn = 23 ,
TC1_IRQn = 24 , TC2_IRQn = 25 , TC3_IRQn = 26 , TC4_IRQn = 27 ,
TC5_IRQn = 28 , AFEC0_IRQn = 29 , DACC_IRQn = 30 , PWM0_IRQn = 31 ,
ICM_IRQn = 32 , ACC_IRQn = 33 , USBHS_IRQn = 34 , MCAN0_IRQn = 35 ,
MCAN1_IRQn = 37 , GMAC_IRQn = 39 , AFEC1_IRQn = 40 , TWIHS2_IRQn = 41 ,
SPI1_IRQn = 42 , QSPI_IRQn = 43 , UART2_IRQn = 44 , UART3_IRQn = 45 ,
UART4_IRQn = 46 , TC6_IRQn = 47 , TC7_IRQn = 48 , TC8_IRQn = 49 ,
TC9_IRQn = 50 , TC10_IRQn = 51 , TC11_IRQn = 52 , **SW_IRQn** = 53 ,
AES_IRQn = 56 , TRNG_IRQn = 57 , XDMAC_IRQn = 58 , ISI_IRQn = 59 ,
PWM1_IRQn = 60 , SDRAMC_IRQn = 62 , RSWDT_IRQn = 63 , PERIPH_COUNT_IRQn = 64 }

Interrupt Number Definitions.

24.36.1 Detailed Description

24.36.2 Macro Definition Documentation

24.36.2.1 __CM7_REV

```
#define __CM7_REV 0x0000
```

Configuration of the Cortex-M7 Processor and Core Peripherals.
SAME70Q21 core revision number ([15:8] revision number, [7:0] patch number)

24.36.2.2 __DCACHE_PRESENT

```
#define __DCACHE_PRESENT 1
```

SAME70Q21 does provide a Data Cache

24.36.2.3 __DTCM_PRESENT

```
#define __DTCM_PRESENT 1
```
SAME70Q21 does provide a Data TCM

24.36.2.4 __FPU_DP

```
#define __FPU_DP 1
```
SAME70Q21 Double precision FPU

24.36.2.5 __FPU_PRESENT

```
#define __FPU_PRESENT 1
```
SAME70Q21 does provide a FPU

24.36.2.6 __ICACHE_PRESENT

```
#define __ICACHE_PRESENT 1
```
SAME70Q21 does provide an Instruction Cache

24.36.2.7 __ITCM_PRESENT

```
#define __ITCM_PRESENT 1
```
SAME70Q21 does provide an Instruction TCM

24.36.2.8 __MPU_PRESENT

```
#define __MPU_PRESENT 1
```
SAME70Q21 does provide a MPU

24.36.2.9 __NVIC_PRIO_BITS

```
#define __NVIC_PRIO_BITS 3
```
SAME70Q21 uses 3 Bits for the Priority Levels

24.36.2.10 __Vendor_SysTickConfig

```
#define __Vendor_SysTickConfig 0
```
Set to 1 if different SysTick Config is used

24.36.3 Typedef Documentation

24.36.3.1 IRQn_Type

```
typedef enum IRQn IRQn_Type
```
Interrupt Number Definitions.

24.36.4 Enumeration Type Documentation

24.36.4.1 IRQn

enum IRQn
Interrupt Number Definitions.

Enumerator

NonMaskableInt_IRQn	2 Non Maskable Interrupt
HardFault_IRQn	3 HardFault Interrupt
MemoryManagement_IRQn	4 Cortex-M7 Memory Management Interrupt
BusFault_IRQn	5 Cortex-M7 Bus Fault Interrupt
UsageFault_IRQn	6 Cortex-M7 Usage Fault Interrupt
SVCall_IRQn	11 Cortex-M7 SV Call Interrupt
DebugMonitor_IRQn	12 Cortex-M7 Debug Monitor Interrupt
PendSV_IRQn	14 Cortex-M7 Pend SV Interrupt
SysTick_IRQn	15 Cortex-M7 System Tick Interrupt
SUPC_IRQn	0 SAME70Q21 Supply Controller (SUPC)
RSTC_IRQn	1 SAME70Q21 Reset Controller (RSTC)
RTC_IRQn	2 SAME70Q21 Real Time Clock (RTC)
RTT_IRQn	3 SAME70Q21 Real Time Timer (RTT)
WDT_IRQn	4 SAME70Q21 Watchdog Timer (WDT)
PMC_IRQn	5 SAME70Q21 Power Management Controller (PMC)
EFC_IRQn	6 SAME70Q21 Enhanced Embedded Flash Controller (EFC)
UART0_IRQn	7 SAME70Q21 UART 0 (UART0)
UART1_IRQn	8 SAME70Q21 UART 1 (UART1)
PIOA_IRQn	10 SAME70Q21 Parallel I/O Controller A (PIOA)
PIOB_IRQn	11 SAME70Q21 Parallel I/O Controller B (PIOB)
PIOC_IRQn	12 SAME70Q21 Parallel I/O Controller C (PIOC)
USART0_IRQn	13 SAME70Q21 USART 0 (USART0)
USART1_IRQn	14 SAME70Q21 USART 1 (USART1)
USART2_IRQn	15 SAME70Q21 USART 2 (USART2)
PIOD_IRQn	16 SAME70Q21 Parallel I/O Controller D (PIOD)
PIOE_IRQn	17 SAME70Q21 Parallel I/O Controller E (PIOE)
HSMCI_IRQn	18 SAME70Q21 Multimedia Card Interface (HSMCI)
TWIHS0_IRQn	19 SAME70Q21 Two Wire Interface 0 HS (TWIHS0)
TWIHS1_IRQn	20 SAME70Q21 Two Wire Interface 1 HS (TWIHS1)
SPI0_IRQn	21 SAME70Q21 Serial Peripheral Interface 0 (SPI0)
SSC_IRQn	22 SAME70Q21 Synchronous Serial Controller (SSC)
TC0_IRQn	23 SAME70Q21 Timer/Counter 0 (TC0)
TC1_IRQn	24 SAME70Q21 Timer/Counter 1 (TC1)
TC2_IRQn	25 SAME70Q21 Timer/Counter 2 (TC2)
TC3_IRQn	26 SAME70Q21 Timer/Counter 3 (TC3)
TC4_IRQn	27 SAME70Q21 Timer/Counter 4 (TC4)
TC5_IRQn	28 SAME70Q21 Timer/Counter 5 (TC5)
AFEC0_IRQn	29 SAME70Q21 Analog Front End 0 (AFEC0)
DACC_IRQn	30 SAME70Q21 Digital To Analog Converter (DACC)
PWM0_IRQn	31 SAME70Q21 Pulse Width Modulation 0 (PWM0)
ICM_IRQn	32 SAME70Q21 Integrity Check Monitor (ICM)

Enumerator

ACC_IRQn	33 SAME70Q21 Analog Comparator (ACC)
USBHS_IRQn	34 SAME70Q21 USB Host / Device Controller (USBHS)
MCAN0_IRQn	35 SAME70Q21 MCAN Controller 0 (MCAN0)
MCAN1_IRQn	37 SAME70Q21 MCAN Controller 1 (MCAN1)
GMAC_IRQn	39 SAME70Q21 Ethernet MAC (GMAC)
AFEC1_IRQn	40 SAME70Q21 Analog Front End 1 (AFEC1)
TWIHS2_IRQn	41 SAME70Q21 Two Wire Interface 2 HS (TWIHS2)
SPI1_IRQn	42 SAME70Q21 Serial Peripheral Interface 1 (SPI1)
QSPI_IRQn	43 SAME70Q21 Quad I/O Serial Peripheral Interface (QSPI)
UART2_IRQn	44 SAME70Q21 UART 2 (UART2)
UART3_IRQn	45 SAME70Q21 UART 3 (UART3)
UART4_IRQn	46 SAME70Q21 UART 4 (UART4)
TC6_IRQn	47 SAME70Q21 Timer/Counter 6 (TC6)
TC7_IRQn	48 SAME70Q21 Timer/Counter 7 (TC7)
TC8_IRQn	49 SAME70Q21 Timer/Counter 8 (TC8)
TC9_IRQn	50 SAME70Q21 Timer/Counter 9 (TC9)
TC10_IRQn	51 SAME70Q21 Timer/Counter 10 (TC10)
TC11_IRQn	52 SAME70Q21 Timer/Counter 11 (TC11)
AES_IRQn	56 SAME70Q21 AES (AES)
TRNG_IRQn	57 SAME70Q21 True Random Generator (TRNG)
XDMAC_IRQn	58 SAME70Q21 DMA (XDMAC)
ISI_IRQn	59 SAME70Q21 Camera Interface (ISI)
PWM1_IRQn	60 SAME70Q21 Pulse Width Modulation 1 (PWM1)
SDRAMC_IRQn	62 SAME70Q21 SDRAM Controller (SDRAMC)
RSWDT_IRQn	63 SAME70Q21 Reinforced Secure Watchdog Timer (RSWDT)
PERIPH_COUNT_IRQn	Number of peripheral IDs

24.37 cm_core_config.h

Go to the documentation of this file.

```
00001 #ifndef  __CM_CORE_CONFIG_H
00002 #define  __CM_CORE_CONFIG_H
00036 /*
00037  * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel Support</a>
00038  */
00039
00044 #define __CM7_REV              0x0000
00045 #define __MPU_PRESENT          1
00046 #define __NVIC_PRIO_BITS       3
00047 #define __FPU_PRESENT          1
00048 #define __FPU_DP               1
00049 #define __ICACHE_PRESENT       1
00050 #define __DCACHE_PRESENT       1
00051 #define __DTCM_PRESENT         1
00052 #define __ITCM_PRESENT         1
00053 #define __Vendor_SysTickConfig 0
00059 typedef enum IRQn
00060 {
00061 /******  Cortex-M7 Processor Exceptions Numbers ***************************/
00062   NonMaskableInt_IRQn   = -14,
00063   HardFault_IRQn        = -13,
00064   MemoryManagement_IRQn = -12,
00065   BusFault_IRQn         = -11,
00066   UsageFault_IRQn       = -10,
00067   SVCall_IRQn           = -5,
00068   DebugMonitor_IRQn     = -4,
00069   PendSV_IRQn           = -2,
00070   SysTick_IRQn          = -1,
00071 /******  SAME70Q21 specific Interrupt Numbers *********************************/
```

```
00072
00073   SUPC_IRQn             =  0,
00074   RSTC_IRQn             =  1,
00075   RTC_IRQn              =  2,
00076   RTT_IRQn              =  3,
00077   WDT_IRQn              =  4,
00078   PMC_IRQn              =  5,
00079   EFC_IRQn              =  6,
00080   UART0_IRQn            =  7,
00081   UART1_IRQn            =  8,
00082   PIOA_IRQn             = 10,
00083   PIOB_IRQn             = 11,
00084   PIOC_IRQn             = 12,
00085   USART0_IRQn           = 13,
00086   USART1_IRQn           = 14,
00087   USART2_IRQn           = 15,
00088   PIOD_IRQn             = 16,
00089   PIOE_IRQn             = 17,
00090   HSMCI_IRQn            = 18,
00091   TWIHS0_IRQn           = 19,
00092   TWIHS1_IRQn           = 20,
00093   SPI0_IRQn             = 21,
00094   SSC_IRQn              = 22,
00095   TC0_IRQn              = 23,
00096   TC1_IRQn              = 24,
00097   TC2_IRQn              = 25,
00098   TC3_IRQn              = 26,
00099   TC4_IRQn              = 27,
00100   TC5_IRQn              = 28,
00101   AFEC0_IRQn            = 29,
00102   DACC_IRQn             = 30,
00103   PWM0_IRQn             = 31,
00104   ICM_IRQn              = 32,
00105   ACC_IRQn              = 33,
00106   USBHS_IRQn            = 34,
00107   MCAN0_IRQn            = 35,
00108   MCAN1_IRQn            = 37,
00109   GMAC_IRQn             = 39,
00110   AFEC1_IRQn            = 40,
00111   TWIHS2_IRQn           = 41,
00112   SPI1_IRQn             = 42,
00113   QSPI_IRQn             = 43,
00114   UART2_IRQn            = 44,
00115   UART3_IRQn            = 45,
00116   UART4_IRQn            = 46,
00117   TC6_IRQn              = 47,
00118   TC7_IRQn              = 48,
00119   TC8_IRQn              = 49,
00120   TC9_IRQn              = 50,
00121   TC10_IRQn             = 51,
00122   TC11_IRQn             = 52,
00123   SW_IRQn               = 53,
00124   AES_IRQn              = 56,
00125   TRNG_IRQn             = 57,
00126   XDMAC_IRQn            = 58,
00127   ISI_IRQn              = 59,
00128   PWM1_IRQn             = 60,
00129   SDRAMC_IRQn           = 62,
00130   RSWDT_IRQn            = 63,
00132   PERIPH_COUNT_IRQn     = 64
00133 } IRQn_Type;
00134
00135
00136 #endif   /* ----- #ifndef __CM_CORE_CONFIG_H  ----- */
```

24.38 conf_mcan.h File Reference

SAM Control Area Network Driver Configuration Header.

Macros

- #define CONF_MCAN_ELEMENT_DATA_SIZE 8
- #define CONF_MCAN_NBTP_NBRP_VALUE 2
- #define CONF_MCAN_NBTP_NSJW_VALUE 3
- #define CONF_MCAN_NBTP_NTSEG1_VALUE 10
- #define CONF_MCAN_NBTP_NTSEG2_VALUE 7
- #define CONF_MCAN_FBTP_FBRP_VALUE 5
- #define CONF_MCAN_FBTP_FSJW_VALUE 3

- #define CONF_MCAN_FBTP_FTSEG1_VALUE 10
- #define CONF_MCAN_FBTP_FTSEG2_VALUE 3

24.38.1 Detailed Description

SAM Control Area Network Driver Configuration Header.

24.38.2 Macro Definition Documentation

24.38.2.1 CONF_MCAN_ELEMENT_DATA_SIZE

```
#define CONF_MCAN_ELEMENT_DATA_SIZE 8
```

The value should be 8/12/16/20/24/32/48/64.

24.38.2.2 CONF_MCAN_FBTP_FBRP_VALUE

```
#define CONF_MCAN_FBTP_FBRP_VALUE 5
```

Data bit Baud Rate Prescaler

24.38.2.3 CONF_MCAN_FBTP_FSJW_VALUE

```
#define CONF_MCAN_FBTP_FSJW_VALUE 3
```

Data bit (Re)Synchronization Jump Width

24.38.2.4 CONF_MCAN_FBTP_FTSEG1_VALUE

```
#define CONF_MCAN_FBTP_FTSEG1_VALUE 10
```

Data bit Time segment before sample point

24.38.2.5 CONF_MCAN_FBTP_FTSEG2_VALUE

```
#define CONF_MCAN_FBTP_FTSEG2_VALUE 3
```

Data bit Time segment after sample point

24.38.2.6 CONF_MCAN_NBTP_NBRP_VALUE

```
#define CONF_MCAN_NBTP_NBRP_VALUE 2
```

The setting of the nominal bit rate is based on the PCK5 which is 30M which you can change in the conf_clock.h. Below is the default configuration. The time quanta is 30MHz / (2+1) = 10MHz. And each bit is (1 + NTSEG1 + 1 + NTSEG2 + 1) = 20 time quanta which means the bit rate is 10MHz/20=500KHz. Nominal bit Baud Rate Prescaler

24.38.2.7 CONF_MCAN_NBTP_NSJW_VALUE

```
#define CONF_MCAN_NBTP_NSJW_VALUE 3
```

Nominal bit (Re)Synchronization Jump Width

24.38.2.8 CONF_MCAN_NBTP_NTSEG1_VALUE

```
#define CONF_MCAN_NBTP_NTSEG1_VALUE 10
```

Nominal bit Time segment before sample point

24.38.2.9 CONF_MCAN_NBTP_NTSEG2_VALUE

```
#define CONF_MCAN_NBTP_NTSEG2_VALUE 7
```

Nominal bit Time segment after sample point

24.39 conf_mcan.h

Go to the documentation of this file.

```
00001
00027 /*
00028  * Support and FAQ: visit <a href="https://www.microchip.com/support/">Microchip Support</a>
00029  */
00030 #ifndef CONF_MCAN_H_INCLUDED
00031 #define CONF_MCAN_H_INCLUDED
00032
00034 #define CONF_MCAN_ELEMENT_DATA_SIZE         8
00035
00043 #define CONF_MCAN_NBTP_NBRP_VALUE    2
00045 #define CONF_MCAN_NBTP_NSJW_VALUE    3
00047 #define CONF_MCAN_NBTP_NTSEG1_VALUE  10
00049 #define CONF_MCAN_NBTP_NTSEG2_VALUE  7
00050
00051 /*
00052  * The setting of the data bit rate is based on the GCLK_MCAN is 48M which you can
00053  * change in the conf_clock.h. Below is the default configuration. The
00054  * time quanta is 48MHz / (5+1) =  8MHz. And each bit is (1 + FTSEG1 + 1 + FTSEG2 + 1) = 16 time
00055  * quanta which means the bit rate is 8MHz/16=500KHz.
00056  */
00058 #define CONF_MCAN_FBTP_FBRP_VALUE    5
00060 #define CONF_MCAN_FBTP_FSJW_VALUE    3
00062 #define CONF_MCAN_FBTP_FTSEG1_VALUE  10
00064 #define CONF_MCAN_FBTP_FTSEG2_VALUE  3
00065
00066 #endif
```

24.40 core_ppb.h

```
00001 #ifndef  __CORE_PPB_H
00002 #define  __CORE_PPB_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 /************ NO UNIQUE CORE IMPLIMENTATION DETAILS ************/
00008
00009 #endif   /* ----- #ifndef __CORE_PPB_H  ----- */
```

24.41 cpu_hal.h

```
00001 #ifndef  __CPU_HAL_H
00002 #define  __CPU_HAL_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <predef.h>
00007
00008 void EnablePeriphClock(int id);
00009
00010 #endif   /* ----- #ifndef __CPU_HAL_H  ----- */
```

24.42 ebi.h File Reference

External Bus Interface (EBI) Header File.

```
#include <predef.h>
#include <stdint.h>
```

Classes

- struct EBI_CS_cfg_t

  Configuration structure for an External Bus Interface (EBI) chip select.

Enumerations

- enum EBI_CS_BusWidth_t { EBI_BUS_WIDTH_8 = 0 , EBI_BUS_WIDTH_16 = 1 }
- enum EBI_CS_ByteAccess_t { EBI_BYTE_ACCESS_SELECT = 0 , EBI_BYTE_ACCESS_WRITE = 1 }
- enum EBI_CS_NWait_t { EBI_NWAIT_DISABLED = 0 , EBI_NWAIT_FROZEN = 2 , EBI_NWAIT_READY = 3 }
- enum EBI_CS_WrMode_t
- enum EBI_CS_RdMode_t

Functions

- void ConfigureEBI_CSPin (int csNum)

  Configure the I/O pin for a given Chip Select for the external data bus.
- void **ConfigureEBI_NWRPin** ()

  Configure the I/O pin for the active low write/read (NWR) bus signal.
- void **ConfigureEBI_NRDPin** ()

  Configure the I/O pin for the active low read (NRD) bus signal.
- void ConfigureEBI_CS (uint32_t csNum, const EBI_CS_cfg_t &&cfg)

  Configure the given Chip Select for the external data bus.
- void ConfigureEBI_CS (uint32_t csNum, const EBI_CS_cfg_t &cfg)

  Configure the given Chip Select for the external data bus.

24.42.1 Detailed Description

External Bus Interface (EBI) Header File.

24.43 ebi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00088 #ifndef  __EBI_H
00089 #define  __EBI_H
00090 #include <predef.h>
00091 #include <stdint.h>
00092
00097 void ConfigureEBI_CSPin(int csNum);
00098
00102 void ConfigureEBI_NWRPin();
00103
00107 void ConfigureEBI_NRDPin();
00108
00112 enum EBI_CS_BusWidth_t {
00113     EBI_BUS_WIDTH_8     = 0,
00114     EBI_BUS_WIDTH_16    = 1,
00115 };
00116
00120 enum EBI_CS_ByteAccess_t {
00121     EBI_BYTE_ACCESS_SELECT = 0,
00122     EBI_BYTE_ACCESS_WRITE  = 1,
00123 };
00124
00135 enum EBI_CS_NWait_t {
00136     EBI_NWAIT_DISABLED      = 0,
00137     EBI_NWAIT_FROZEN        = 2,
00138     EBI_NWAIT_READY         = 3,
00139 };
00140
00144 enum EBI_CS_WrMode_t {
00145     EBI_WRITE_MODE_NCS  = 0,
```

```
00146     EBI_WRITE_MODE_NWE  = 1,
00147 };
00148
00152 enum EBI_CS_RdMode_t {
00153     EBI_READ_MODE_NCS   = 0,
00154     EBI_READ_MODE_NRD   = 1,
00155 };
00156
00208 struct EBI_CS_cfg_t {
00209     uint8_t ncs_rd_setup;
00210     uint8_t nrd_setup;
00211     uint8_t ncs_wr_setup;
00212     uint8_t nwe_setup;
00213
00214     uint8_t ncs_rd_pulse;
00215     uint8_t nrd_pulse;
00216     uint8_t ncs_wr_pulse;
00217     uint8_t nwe_pulse;
00218
00219     uint16_t nrd_cycles;
00220     uint16_t nwe_cycles;
00221
00222     uint8_t tdf_cycles;
00223     EBI_CS_BusWidth_t   busWidth;
00224     EBI_CS_ByteAccess_t byteAccess;
00225     EBI_CS_NWait_t      nWait;
00226     EBI_CS_WrMode_t     wrMode;
00227     EBI_CS_RdMode_t     rdMode;
00228 };
00229
00236 void ConfigureEBI_CS(uint32_t csNum, const EBI_CS_cfg_t &&cfg);
00237
00244 void ConfigureEBI_CS(uint32_t csNum, const EBI_CS_cfg_t &cfg);
00245
00246 #endif   /* ----- #ifndef __EBI_H  ----- */
00247
00248
```

24.44 i2c.h File Reference

NetBurner I2C API for ARM SAME70.

```
#include <predef.h>
#include <ctype.h>
#include <sim.h>
#include <nbrtos.h>
```

Classes

- class WireIntf

  Wire Interface Class for I2C.
- class I2C

  I2C Peripheral Class.
- class I2CDevice

  I2C Device Class (recommended)

24.44.1 Detailed Description

NetBurner I2C API for ARM SAME70.

24.45 i2c.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00024 #ifndef  __I2C_H
00025 #define  __I2C_H
00026
00027 #include <predef.h>
```

```
00028 #include <ctype.h>
00029 #include <sim.h>
00030 #include <nbrtos.h>
00031
00032
00033 #define TWIHS_SR_IOS (TWIHS_SR_SCL | TWIHS_SR_SDA)
00034
00035 struct I2CTxn_t {
00036     uint8_t *buf;
00037     uint32_t blen;
00038     uint8_t  regAddr[3];
00039     uint8_t  devAddr : 7;
00040     bool     RnW : 1;
00041     uint8_t  regAddrLen : 3;
00042     bool     bIssueStop : 1;
00043
00044     void dump(uint32_t line);
00045 };
00046
00047 class I2C;
00048
00049
00050 /*---------------------------------------------------------------------------------------------
00051  * Wire Interface Class
00052  *--------------------------------------------------------------------------------------------*/
00053
00065 class WireIntf {
00066     I2C     &i2c;            // I2C instance of this wire interface
00067     uint8_t txnBuf[128];    // Transaction buffer. Build up transaction before being flushed to the
      I2C bus
00068     uint8_t txnLen;         // Current transaction length
00069     uint8_t devAddr;        // I2C slave device adddress to be accessed
00070     uint8_t bytesRead;      // Number of bytes read, returned by requestFrom(). Provides number of
      bytes available to read()
00071     uint8_t readIdx;        // Read index. Location in the transaction buffer of next byte to be read
00072     bool    bTxnStarted;    // Whether or not the wire interface is in an active transaction
00073     bool    bBusStarted;    // Whether or not the bus is active (h/w signal state)
00074
00075 public:
00076     WireIntf(I2C &module);
00077
00083     void begin();
00084
00098     uint32_t requestFrom(uint8_t addr, uint32_t len, bool stop = true);
00099
00107     void beginTransmission(uint8_t addr);
00108
00119     void endTransmission(bool stop = true);
00120
00131     uint32_t write(uint8_t dat);
00132
00143     uint32_t write(char * str);
00144
00155     uint32_t write(uint8_t * buf, uint32_t len);
00156
00164     uint32_t available();
00165
00172     uint8_t read();
00173
00185     void flush(bool bIssueStop = false);
00186 };
00187
00188
00189 /*
00190  * I2C WireIntf objects instantiated by the system.
00191  *
00192  * Provides access to the all the I2C peripherals on the processor. The I2C peripherals available will
      vary by platform.
00193  * For example, Wire.read(); will exectute a read opeeration on I2C module 0.
00194  */
00195 extern WireIntf Wire;   // I2C module 0
00196 extern WireIntf Wire1;  // I2C module 1
00197 extern WireIntf Wire2;  // I2C module 2
00198
00199
00200
00201 /*---------------------------------------------------------------------------------------------
00202  * I2C Class
00203  *--------------------------------------------------------------------------------------------*/
00204
00216 class I2C {
00217 public:
00222     enum Result_t {
00223         I2C_RES_ACK,
00224         I2C_RES_NACK,
00225         I2C_RES_ARB_LST,
00226         I2C_RES_BUSY,
```

```
00227         I2C_RES_ARG
00228     };
00231 private:
00232     enum TxnStat {
00233         TXN_RDY,
00234         TXN_IN_PROGRESS,
00235         TXN_WAITING
00236     };
00237
00238     Twihs &twi;
00239     int modNum;
00240     TxnStat txnStatus;
00241     Result_t txnResult;
00242     uint32_t sticky_sr;
00243     uint8_t  iadrAddressSize;   // Number of address register bytes, 0 - 3
00244
00245     volatile I2CTxn_t *pTxn;
00246     OS_SEM txnSem;
00247
00248     void isr();
00249     void isr_rx(uint32_t sr);
00250     void isr_tx(uint32_t sr);
00251
00252     inline uint32_t getStatus()
00253         { uint32_t sr = twi.TWIHS_SR; sticky_sr |= sr; return sr; }
00254     inline uint32_t getStickyStatus()
00255         { uint32_t sr = twi.TWIHS_SR; sticky_sr |= sr; return sticky_sr; }
00256     inline void clrStickyStatus() { sticky_sr = 0; }
00257
00258     inline bool busBusy()
00259         { return ((getStatus() & TWIHS_SR_IOS) != TWIHS_SR_IOS); }
00260
00261     void start(uint8_t deviceAddr, bool rnw, uint8_t regAddrlen = 0, bool bIssueStop = false);
00262     void restart(uint8_t deviceAddr, bool rnw, uint8_t regAddrlen = 0, bool bIssueStop = false);
00263     void stop();
00264
00265     Result_t write8(uint8_t dat);
00266     Result_t read8(uint8_t &dat);
00267
00268 public:
00275     I2C(int module);
00276
00277     // Copy constructor
00278     I2C(const I2C & rhs)
00279         : twi(rhs.twi), modNum(rhs.modNum), txnStatus(rhs.txnStatus)
00280     {}
00281
00288     void setup(uint32_t busSpeed);
00289
00294     void resetBus();
00295
00307     Result_t DoTransaction(I2CTxn_t *pTransaction, bool bRepeatedStart = false);
00308
00323     inline void setNumAddressBytes(uint8_t numAddressBytes = 1) { iadrAddressSize = numAddressBytes; }
00324
00334     Result_t writeReg8(uint8_t devAddr, uint32_t reg, uint8_t data);
00335
00345     Result_t readReg8(uint8_t devAddr, uint32_t reg, uint8_t &data);
00346
00361     Result_t writeRegN(uint8_t devAddr, uint32_t reg, const uint8_t *buf, uint32_t blen);
00362
00373     Result_t readRegN(uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen);
00374
00375     // Static implementations of the C++ class that can be used if you prefer a C style interface
00376     static Result_t writeReg8(int module, uint8_t devAddr, uint32_t reg, uint8_t dat);
00377     static Result_t readReg8(int module, uint8_t devAddr, uint32_t reg, uint8_t &dat);
00378     static Result_t writeRegN(int module, uint8_t devAddr, uint32_t reg, const uint8_t *buf, uint32_t
      blen);
00379     static Result_t readRegN(int module, uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen);
00380
00381     // These functions are for internal use only.
00382     void dump(uint32_t line);
00383     friend void TWIHS0_Handler();
00384     friend void TWIHS1_Handler();
00385     friend void TWIHS2_Handler();
00386
00387     friend class WireIntf;
00388
00389 };  // end class I2C
00390
00391
00392 // I2C instantions exist for all I2C (TWIHS) peripherals. They are accessed as array: i2c[0], i2c[1],
      i2c[2]
00393 extern I2C i2c[];
00394
00395 inline I2C::Result_t I2C::writeReg8(int module, uint8_t devAddr, uint32_t reg, uint8_t dat)
00396 {
```

```
00397     if (module < 0) { return I2C_RES_NACK; }
00398     if (module > 2) { return I2C_RES_NACK; }
00399     return i2c[module].writeReg8(devAddr, reg, dat);
00400 }
00401
00402 inline I2C::Result_t I2C::readReg8(int module, uint8_t devAddr, uint32_t reg, uint8_t &dat)
00403 {
00404     if (module < 0) { return I2C_RES_NACK; }
00405     if (module > 2) { return I2C_RES_NACK; }
00406     return i2c[module].readReg8(devAddr, reg, dat);
00407 }
00408
00409 inline I2C::Result_t I2C::writeRegN(int module, uint8_t devAddr, uint32_t reg, const uint8_t *buf,
      uint32_t blen)
00410 {
00411     if (module < 0) { return I2C_RES_NACK; }
00412     if (module > 2) { return I2C_RES_NACK; }
00413     return i2c[module].writeRegN(devAddr, reg, buf, blen);
00414 }
00415 inline I2C::Result_t I2C::readRegN(int module, uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t
      blen)
00416 {
00417     if (module < 0) { return I2C_RES_NACK; }
00418     if (module > 2) { return I2C_RES_NACK; }
00419     return i2c[module].readRegN(devAddr, reg, buf, blen);
00420 }
00421
00422
00423 /*---------------------------------------------------------------------------------------------------
00424  * I2C Device Class
00425  *--------------------------------------------------------------------------------------------------*/
00442 class I2CDevice {
00443
00444 private:
00445     I2C     *pI2CInterface;     // Pointer to I2C class object for peripheral/interface
00446     uint8_t devAddress;         // I2C device address, 1 byte, 1 to 127
00447     uint8_t numRegAddrBytes;    // Number of register address bytes to send to device, 0 to 3
00448
00449 public:
00450
00465     inline I2CDevice(I2C & pInterface, uint8_t deviceAddress, uint8_t numAddressBytes = 1)
00466     {
00467         pI2CInterface =     &pInterface;        // Pointer to the I2C peripheral module object
00468         devAddress =        deviceAddress;      // Device I2C address
00469         numRegAddrBytes =   numAddressBytes;    // Number of bytes to send for register address: 0 - 3
00470     }
00471
00484     inline uint8_t getI2CAddress() { return devAddress; }
00485
00486
00493     inline void setup(uint32_t busSpeed);
00494
00499     inline void resetBus();
00500
00501
00510     inline I2C::Result_t writeReg8(uint32_t reg, uint8_t data)
00511     {
00512         pI2CInterface->setNumAddressBytes(numRegAddrBytes);
00513         return pI2CInterface->writeReg8(devAddress, reg, data);
00514     }
00515
00524     inline I2C::Result_t readReg8(uint32_t reg, uint8_t &data)
00525     {
00526         pI2CInterface->setNumAddressBytes(numRegAddrBytes);
00527         return pI2CInterface->readReg8(devAddress, reg, data);
00528     }
00529
00543     inline I2C::Result_t writeRegN(uint32_t reg, const uint8_t *buf, uint32_t blen)
00544     {
00545         pI2CInterface->setNumAddressBytes(numRegAddrBytes);
00546         return pI2CInterface->writeRegN(devAddress, reg, buf, blen);
00547     }
00548
00558     inline I2C::Result_t readRegN(uint32_t reg, uint8_t *buf, uint32_t blen)
00559     {
00560         pI2CInterface->setNumAddressBytes(numRegAddrBytes);
00561         return pI2CInterface->readRegN(devAddress, reg, buf, blen);
00562     }
00563 };
00564
00565
00566
00567 #endif   /* ----- #ifndef __I2C_H  ----- */
00568
```

24.46 mcan.h File Reference

SAM Control Area Network (MCAN) Low Level Driver.

```
#include <conf_mcan.h>
#include <mcan_internal.h>
```

Classes

- class mcanMODM7AE70::mcan_module

  MCAN Module Class.
- class mcanMODM7AE70::mcan_config

  MCAN configuration structure.
- class mcanMODM7AE70::CanRxMessage

  Class to hold received CAN messages.

Namespaces

- namespace mcanMODM7AE70

  mcanMODM7AE70 namespace

Macros

- #define **CAN_EXTENDED_ID_BIT** (0x20000000)

  The single bit used by this API to indicate an extended ID.
- #define **ExtToNbId**(id) (id | CAN_EXTENDED_ID_BIT)

  Convert an ID to an extended ID.
- #define **NormToNbId**(id) (id & 0x7ff)

  Convert a normal ID in the range of 0 to 2048 to the range of 0 to 1024.
- #define IsNBIdExt(id) ((id & CAN_EXTENDED_ID_BIT)!=0)

  Check for an extened MCAN ID.
- #define **NbToExtId**(id) (id & 0x1FFFFFFF)

  Remove the API extended flage from the ID.
- #define NbToNormId(id) (id & 0x7FF)

  Shift a Normal ID so it has a value from 0 to 1023.
- #define **CAN_DATA_STORE_SIZE** (512)

  Receive OS_FIFO Buffer Size.

Variables

- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_FIFO_0_NUM = 32
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_FIFO_1_NUM = 1
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_BUFFER_NUM = 1
- const uint32_t mcanMODM7AE70::CONF_MCAN_TX_BUFFER_NUM = 8
- const uint32_t mcanMODM7AE70::CONF_MCAN_TX_FIFO_QUEUE_NUM = 1
- const uint32_t mcanMODM7AE70::CONF_MCAN_TX_EVENT_FIFO = 8
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_STANDARD_ID_FILTER_NUM = 32
- const uint32_t mcanMODM7AE70::CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM = 32

24.46.1 Detailed Description

SAM Control Area Network (MCAN) Low Level Driver.
This driver derived from

24.47 mcan.h

Go to the documentation of this file.

```
00001
00031 /*
00032  * Support and FAQ: visit <a href="https://www.microchip.com/support/">Microchip Support</a>
00033  */
00034
00035 #ifndef MCAN_H_INCLUDED
00036 #define MCAN_H_INCLUDED
00037
00038 #include <conf_mcan.h>
00039 #include <mcan_internal.h>
00040
00041 #ifdef DOXYGEN_STUFF
00045 namespace mcanMODM7AE70
00046 {
00047 #endif
00048
00079 #define SAME70B 1
00080
00081
00089 const uint32_t CONF_MCAN_RX_FIFO_0_NUM              = 32; //16
00090
00092 const uint32_t CONF_MCAN_RX_FIFO_1_NUM              = 1;  //16
00093
00095 const uint32_t CONF_MCAN_RX_BUFFER_NUM              = 1;  //16
00096
00098 const uint32_t CONF_MCAN_TX_BUFFER_NUM              = 8;  // 4
00099
00101 const uint32_t CONF_MCAN_TX_FIFO_QUEUE_NUM          = 1;  // 4
00102
00104 const uint32_t CONF_MCAN_TX_EVENT_FIFO              = 8;  // 8
00105
00107 const uint32_t CONF_MCAN_RX_STANDARD_ID_FILTER_NUM  = 32; //32
00108
00110 const uint32_t CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM  = 32; //16
00111
00112 /****************WARNING **********************/
00113 /* This is added to
00114 CONF_MCAN_RX_STANDARD_ID_FILTER_NUM
00115 and
00116 CONF_MCAN_RX_EXTENDED_ID_FILTER_NUMc
00117 and
00118 CONF_MCAN_TX_BUFFER_NUM
00119
00120 and the limits above must be ok with the added values.
00121 */
00122 const uint32_t CONF_MCAN_RTR_CHANNELS           = 8;
00123
00134 class mcan_module
00135 {
00136     private:
00137         Mcan *hw;
00138         uint32_t TxBufferIndex;
00139         OS_SEM TxSem;
00140         volatile uint32_t standard_receive_index = 0;
00141         volatile uint32_t extended_receive_index = 0;
00142
```

```
00143         __attribute__((__aligned__(0x0800)))
00144         struct mcan_rx_element_buffer mcan_rx_buffer[CONF_MCAN_RX_BUFFER_NUM];
00145         struct mcan_rx_element_buffer mcan_rx_fifo_0[CONF_MCAN_RX_FIFO_0_NUM];
00146         struct mcan_rx_element_buffer mcan_rx_fifo_1[CONF_MCAN_RX_FIFO_1_NUM];
00147         struct mcan_tx_element mcan_tx_buffer[CONF_MCAN_TX_BUFFER_NUM +
      CONF_MCAN_TX_FIFO_QUEUE_NUM+CONF_MCAN_RTR_CHANNELS];
00148         struct mcan_tx_event_element
      mcan_tx_event_fifo[CONF_MCAN_TX_EVENT_FIFO+CONF_MCAN_RTR_CHANNELS];
00149         struct mcan_standard_message_filter_element
      mcan_rx_standard_filter[CONF_MCAN_RX_STANDARD_ID_FILTER_NUM+CONF_MCAN_RTR_CHANNELS];
00150         struct mcan_extended_message_filter_element
      mcan_rx_extended_filter[CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM+CONF_MCAN_RTR_CHANNELS];
00151
00152         tx_record mcan_tx_record[CONF_MCAN_TX_BUFFER_NUM];
00153
00154         OS_FIFO * Extended_Fifos[CONF_MCAN_RX_EXTENDED_ID_FILTER_NUM];
00155         OS_FIFO * Standard_Fifos[CONF_MCAN_RX_STANDARD_ID_FILTER_NUM];
00156
00157         void enable_peripheral_clock();
00158         void message_memory_init();
00159         void set_configuration(struct mcan_config *config);
00160         void clean_up_pending_tx();
00161
00162         void process_isr(void);
00163
00164         void PushToFifo(mcan_rx_element_buffer * pBuffer, OS_FIFO * pFifo);
00165         void ack_tx(uint32_t index);
00166         void process_rx(mcan_rx_element_buffer * pBuffer);
00167
00168         static mcan_module * this_Ref[2];
00169         static inline void dispatch_isr(int n) {if(this_Ref[n]) this_Ref[n]->process_isr(); };
00170         friend void MCAN0_Handler( void );
00171         friend void MCAN1_Handler( void );
00172
00173         void set_baudrate(uint32_t baudrate);
00174         void fd_set_baudrate(uint32_t baudrate);
00175         void start();
00176         void stop();
00177         void enable_fd_mode();
00178         void disable_fd_mode();
00179         void enable_restricted_operation_mode();
00180         void disable_restricted_operation_mode();
00181         void enable_bus_monitor_mode();
00182         void disable_bus_monitor_mode();
00183         void enable_sleep_mode();
00184         void disable_sleep_mode();
00185         void enable_test_mode();
00186         void disable_test_mode();
00187         enum status_code set_rx_standard_filter(struct mcan_standard_message_filter_element
      *sd_filter, uint32_t index);
00188         enum status_code set_rx_extended_filter(struct mcan_extended_message_filter_element
      *et_filter, uint32_t index);
00189         enum status_code get_rx_buffer_element (struct mcan_rx_element_buffer *rx_element, uint32_t
      index);
00190         enum status_code get_rx_fifo_0_element (struct mcan_rx_element_buffer *rx_element, uint32_t
      index);
00191         enum status_code get_rx_fifo_1_element (struct mcan_rx_element_buffer *rx_element, uint32_t
      index);
00192         enum status_code set_tx_buffer_element (struct mcan_tx_element *tx_element, uint32_t index);
00193         enum status_code get_tx_event_fifo_element(struct mcan_tx_event_element *tx_event_element,
      uint32_t index);
00194
00195         /*
00196          * \brief Can read timestamp count value.
00197          *
00198          * \param[in] module_inst  Pointer to the MCAN software instance struct
00199          *
00200          * \return Timestamp count value.
00201          */
00202          inline uint16_t read_timestamp_count_value(){return hw->MCAN_TSCV;};
00203
00204         /*
00205          * \brief Can read timeout count value.
00206          *
00207          * \param[in] module_inst  Pointer to the MCAN software instance struct
00208          *
00209          * \return Timeout count value.
00210          */
00211          inline uint16_t read_timeout_count_value(){return hw->MCAN_TOCV;};
00212
00213         /*
00214          * \brief Can read error count.
00215          *
00216          * \param[in] module_inst  Pointer to the MCAN software instance struct
00217          *
00218          * \return Error count value.
00219          */
```

```
00220          inline uint32_t read_error_count(){return hw->MCAN_ECR;};
00221
00222         /*
00223          * \brief Can read protocol status.
00224          *
00225          * \param[in] module_inst  Pointer to the MCAN software instance struct
00226          *
00227          * \return protocol status value.
00228          */
00229          inline uint32_t read_protocal_status(){ return hw->MCAN_PSR;};
00230
00231         /*
00232          * \brief Read high priority message status.
00233          *
00234          * \param[in] module_inst  Pointer to the MCAN software instance struct
00235          *
00236          * \return High priority message status value.
00237          */
00238         inline uint32_t read_high_priority_message_status(){return hw->MCAN_HPMS;}
00239
00240         /*
00241          * \brief Get Rx buffer status.
00242          *
00243          * \param[in] module_inst  Pointer to the MCAN software instance struct
00244          * \param[in] index  Index offset in Rx buffer
00245          *
00246          * \return Rx buffer status value.
00247          *
00248          *  \retval true Rx Buffer updated from new message.
00249          *  \retval false Rx Buffer not updated.
00250          */
00251         inline bool rx_get_buffer_status(uint32_t index)
00252         {
00253             if (index < 32)
00254             {
00255                 if (hw->MCAN_NDAT1 & (1 << index))
00256                 {
00257                     return true;
00258                 } else
00259                 {
00260                     return false;
00261                 }
00262             } else
00263             {
00264                 index -= 32;
00265                 if (hw->MCAN_NDAT2 & (1 << index))
00266                 {
00267                     return true;
00268                 } else
00269                 {
00270                     return false;
00271                 }
00272             }
00273         };
00274
00275         /*
00276          * \brief Clear Rx buffer status.
00277          *
00278          * \param[in] module_inst   Pointer to the MCAN software instance struct
00279          * \param[in] index         Index offset in Rx buffer
00280          */
00281         inline void rx_clear_buffer_status(uint32_t index)
00282         {
00283             if (index < 32)
00284             {
00285                 hw->MCAN_NDAT1 = (1 << index);
00286             } else
00287             {
00288                 index -= 32;
00289                 hw->MCAN_NDAT2 = (1 << index);
00290             }
00291         }
00292
00293         /*
00294          * \brief Get Rx FIFO status.
00295          *
00296          * \param[in] module_inst  Pointer to the MCAN software instance struct
00297          * \param[in] fifo_number  Rx FIFO 0 or 1
00298          *
00299          * \return Rx FIFO status value.
00300          */
00301         inline uint32_t rx_get_fifo_status(bool fifo_number)
00302         {
00303             if (!fifo_number)
00304             {
00305                 return hw->MCAN_RXF0S;
00306             } else
```

```
00307             {
00308                 return hw->MCAN_RXF1S;
00309             }
00310         }
00311
00312         /*
00313          * \brief Set Rx acknowledge.
00314          *
00315          * \param[in] module_inst  Pointer to the MCAN software instance struct
00316          * \param[in] fifo_number  Rx FIFO 0 or 1
00317          * \param[in] index  Index offset in FIFO
00318          */
00319         inline void rx_fifo_acknowledge(bool fifo_number, uint32_t index)
00320         {
00321             if (!fifo_number)
00322             {
00323                 hw->MCAN_RXF0A = MCAN_RXF0A_F0AI(index);
00324             } else
00325             {
00326                 hw->MCAN_RXF1A = MCAN_RXF1A_F1AI(index);
00327             }
00328         }
00329
00330         /*
00331          * \brief Get Tx FIFO/Queue status.
00332          *
00333          * \param[in] module_inst  Pointer to the MCAN software instance struct
00334          *
00335          * \return Tx FIFO/Queue status value.
00336          */
00337         inline uint32_t tx_get_fifo_queue_status()
00338         {
00339             return hw->MCAN_TXFQS;
00340         }
00341
00342         /*
00343          * \brief Get Tx buffer request pending status.
00344          *
00345          * \param[in] module_inst  Pointer to the MCAN software instance struct
00346          *
00347          * \return Bit mask of Tx buffer request pending status value.
00348          */
00349         inline uint32_t tx_get_pending_status()
00350         {
00351             return hw->MCAN_TXBRP;
00352         }
00353
00354         /*
00355          * \brief Tx buffer add transfer request.
00356          *
00357          * \param[in] module_inst  Pointer to the MCAN software instance struct
00358          * \param[in] trig_mask  The mask value to trigger transfer buffer
00359          *
00360          *  \return Status of the result.
00361          *
00362          *  \retval STATUS_OK   Set the transfer request.
00363          *  \retval STATUS_ERR_BUSY The module is in configuration.
00364          */
00365         inline enum status_code tx_transfer_request( uint32_t trig_mask)
00366         {
00367             if (hw->MCAN_CCCR & MCAN_CCCR_CCE)
00368             {
00369                 return ERR_BUSY;
00370             }
00371             hw->MCAN_TXBAR = trig_mask;
00372             return STATUS_OK;
00373         }
00374
00375         /*
00376          * \brief Set Tx Queue operation.
00377          *
00378          * \param[in] module_inst  Pointer to the MCAN software instance struct
00379          * \param[in] trig_mask  The mask value to cancel transfer buffer
00380          *
00381          *  \return Status of the result.
00382          *
00383          *  \retval STATUS_OK   Set the transfer request.
00384          *  \retval STATUS_BUSY The module is in configuration.
00385          */
00386         inline enum status_code tx_cancel_request( uint32_t trig_mask)
00387         {
00388             if (hw->MCAN_CCCR & MCAN_CCCR_CCE)
00389             {
00390                 return STATUS_ERR_BUSY;
00391             }
00392             hw->MCAN_TXBCR = trig_mask;
00393             return STATUS_OK;
```

```
00394         }
00395
00396         /*
00397          * \brief Get Tx transmission status.
00398          *
00399          * \param[in] module_inst  Pointer to the MCAN software instance struct
00400          *
00401          * \return Bit mask of Tx transmission status value.
00402          */
00403         inline uint32_t tx_get_transmission_status()
00404         {
00405             return hw->MCAN_TXBTO;
00406         }
00407
00408         /*
00409          * \brief Get Tx cancellation status.
00410          *
00411          * \param[in] module_inst  Pointer to the MCAN software instance struct
00412          *
00413          * \return Bit mask of Tx cancellation status value.
00414          */
00415         inline uint32_t tx_get_cancellation_status()
00416         {
00417             return hw->MCAN_TXBCF;
00418         }
00419
00420         /*
00421          * \brief Get Tx event FIFO status.
00422          *
00423          * \param[in] module_inst  Pointer to the MCAN software instance struct
00424          *
00425          * \return Tx event FIFO status value.
00426          */
00427         inline uint32_t tx_get_event_fifo_status()
00428         {
00429             return hw->MCAN_TXEFS;
00430         }
00431
00432         /*
00433          * \brief Set Tx Queue operation.
00434          *
00435          * \param[in] module_inst  Pointer to the MCAN software instance struct
00436          * \param[in] index  Index for the transfer FIFO
00437          */
00438         inline void tx_event_fifo_acknowledge( uint32_t index)
00439         {
00440             hw->MCAN_TXEFA = MCAN_TXEFA_EFAI(index);
00441         }
00442
00443         /*
00444          * \brief Enable MCAN interrupt.
00445          *
00446          * \param[in] module_inst  Pointer to the MCAN software instance struct
00447          * \param[in] source  Interrupt source type
00448          */
00449         inline void enable_interrupt(const enum mcan_interrupt_source source)
00450         {
00451             hw->MCAN_IE |= source;
00452         }
00453
00454         /*
00455          * \brief Disable MCAN interrupt.
00456          *
00457          * \param[in] module_inst  Pointer to the MCAN software instance struct
00458          * \param[in] source  Interrupt source type
00459          */
00460         inline void disable_interrupt(const enum mcan_interrupt_source source)
00461         {
00462             hw->MCAN_IE &= ~source;
00463         }
00464
00465         /*
00466          * \brief Get MCAN interrupt status.
00467          *
00468          * \param[in] module_inst  Pointer to the MCAN software instance struct
00469          */
00470         inline uint32_t read_interrupt_status()
00471         {
00472             return hw->MCAN_IR;
00473         }
00474
00475         /*
00476          * \brief Clear MCAN interrupt status.
00477          *
00478          * \param[in] module_inst  Pointer to the MCAN software instance struct
00479          * \param[in] source  Interrupt source type
00480          *
```

```
00481          * \return Bit mask of interrupt status value.
00482          */
00483         inline void clear_interrupt_status(const enum mcan_interrupt_source source)
00484         {
00485             hw->MCAN_IR = source;
00486         }
00487
00488
00489     public:
00497         mcan_module(Mcan *hw, uint32_t baud);
00498
00499         mcan_module(){};
00500
00510         void init(Mcan *hw, struct mcan_config *config, uint32_t baud);
00511
00521         void send_message(uint32_t id_value, uint8_t *data,uint32_t data_length, OS_SEM * pSem=0);
00522
00532         bool blocking_send_message(uint32_t id_value, uint8_t *data,uint32_t data_length,uint32_t
      TimeOut);
00533
00534
00544         int RegisterRxFifo(uint32_t composite_id, OS_FIFO *pFifo, int channel = -1 );
00545
00557         int RegisterRxFifoMask(uint32_t composite_id, uint32_t mask, OS_FIFO *pFifo, int channel = -1
      );
00558
00569         int RegisterRxFifoRange(uint32_t composite_id_low,uint32_t composite_id_hi, OS_FIFO *pFifo,
      int channel = -1 );
00570
00578         int UnRegisterFifo(int channel );
00579
00595         int MultiCanSetRTRMessage(uint32_t id, uint8_t *data, uint8_t len, int channel = -1 );
00596
00604         int MultiCanReplaceRTRMessage(int channel, uint8_t *data, uint8_t len );
00605
00615         int MultiCanStopRTRMessage(int channel );
00616
00617         // Toggle the TX line hi/low and print the result of the RX line.
00618         void IOTest();
00619
00620 };      // end class mcan_module
00621
00622
00631 #define CAN_EXTENDED_ID_BIT (0x20000000)
00632
00636 #define ExtToNbId(id ) (id | CAN_EXTENDED_ID_BIT)
00637
00641 #define NormToNbId(id) (id & 0x7ff)
00642
00648 #define IsNBIdExt(id) ( (id & CAN_EXTENDED_ID_BIT)!=0)
00649
00653 #define NbToExtId(id) (id & 0x1FFFFFFF)
00654
00662 #define NbToNormId(id) (id & 0x7FF)
00663
00664
00675 class mcan_config
00676 {
00677     public:
00679         bool run_in_standby;
00681         uint8_t watchdog_configuration;
00683         bool transmit_pause;
00685         bool edge_filtering;
00687         bool protocol_exception_handling;
00689         bool automatic_retransmission;
00691         bool clock_stop_request;
00693         bool clock_stop_acknowledge;
00695         uint8_t timestamp_prescaler;
00697         uint16_t timeout_period;
00699         enum mcan_timeout_mode timeout_mode;
00701         bool timeout_enable;
00703         bool tdc_enable;
00705         uint8_t delay_compensation_offset;
00707         uint8_t delay_compensation_filter_window_length;
00709         enum mcan_nonmatching_frames_action nonmatching_frames_action_standard;
00711         enum mcan_nonmatching_frames_action nonmatching_frames_action_extended;
00713         bool remote_frames_standard_reject;
00715         bool remote_frames_extended_reject;
00717         uint32_t extended_id_mask;
00719         bool rx_fifo_0_overwrite;
00721         uint8_t rx_fifo_0_watermark;
00723         bool rx_fifo_1_overwrite;
00725         uint8_t rx_fifo_1_watermark;
00727         bool tx_queue_mode;
00729         uint8_t tx_event_fifo_watermark;
00730
00766     inline void set_config_defaults()
```

```
00767     {
00768
00769         /* Default configuration values */
00770         run_in_standby = false;
00771         watchdog_configuration = 0x00;
00772         transmit_pause = true;
00773         edge_filtering = true;
00774         protocol_exception_handling = true;
00775         automatic_retransmission = true;  //was true
00776         clock_stop_request = false;
00777         clock_stop_acknowledge = false;
00778         timestamp_prescaler = 0;
00779         timeout_period = 0xFFFF;
00780         timeout_mode = MCAN_TIMEOUT_CONTINUES;
00781         timeout_enable = false;
00782         tdc_enable = false;
00783         delay_compensation_offset = 0;
00784         delay_compensation_filter_window_length = 0;
00785         nonmatching_frames_action_standard = MCAN_NONMATCHING_FRAMES_REJECT;
00786         nonmatching_frames_action_extended = MCAN_NONMATCHING_FRAMES_REJECT;
00787         remote_frames_standard_reject = false;
00788         remote_frames_extended_reject = false;
00789         extended_id_mask = 0x1FFFFFFF;
00790         rx_fifo_0_overwrite = true;
00791         rx_fifo_0_watermark = 0;
00792         rx_fifo_1_overwrite = true;
00793         rx_fifo_1_watermark = 0;
00794         tx_queue_mode = false;
00795         tx_event_fifo_watermark = 0;
00796     }
00797
00798 };      // end mcan_config class
00799
00800
00801
00806 #define CAN_DATA_STORE_SIZE (512)
00810 struct PrivateCanData;
00811
00815 class CanRxMessage
00816 {
00817    private:
00818       PrivateCanData *pData;
00819       /* Private constructor used for received frames */
00820       CanRxMessage( PrivateCanData *pData );
00821
00822    public:
00828       uint8_t GetLength();
00829
00838       uint8_t CopyData( uint8_t *buffer, uint8_t max_len );
00839
00846       const uint8_t * GetData();
00847
00853       uint32_t GetId();
00854
00875       uint16_t GetTimeStamp();
00876
00884       BOOL IsValid();
00885
00886       /* Constructors */
00887
00898       CanRxMessage( OS_FIFO *pFifo, uint32_t timeout = WAIT_FOREVER );
00899
00904       ~CanRxMessage();
00905
00916       BOOL GetNewMessage( OS_FIFO *pFifo, uint32_t timeout = WAIT_FOREVER );
00917 };
00918
00919
00922 #ifdef DOXYGEN_STUFF
00923 }   // namespace
00924 #endif
00925
00926
00927 #endif /* MCAN_H_INCLUDED */
00928
00929
00930
```

24.48 mcan_internal.h

```
00001
00002 #ifndef __MCAN_INTERNAL_H__
00003 #define __MCAN_INTERNAL_H__
00004
00005 #include <same70q21_sim.h>
```

```
00006
00007 /* -------- MCAN_RX_ELEMENT_R0 : (MCAN RX element: 0x00) (R/W 32) Rx Element R0 Configuration --------
      */
00008 typedef union
00009 {
00010     struct
00011     {
00012         /* bit:  0..28  Identifier */
00013         uint32_t ID:29;
00014         /* bit:  29     Remote Transmission Request */
00015         uint32_t RTR:1;
00016         /* bit:  30     Extended Identifier */
00017         uint32_t XTD:1;
00018         /* bit:  31     Error State Indicator */
00019         uint32_t ESI:1;
00020     } bit;
00021     /* Type used for register access */
00022     uint32_t reg;
00023 } MCAN_RX_ELEMENT_R0_Type;
00024
00025 #define MCAN_RX_ELEMENT_R0_ID_Pos          0
00026 #define MCAN_RX_ELEMENT_R0_ID_Msk          (0x1FFFFFFFul « MCAN_RX_ELEMENT_R0_ID_Pos)
00027 #define MCAN_RX_ELEMENT_R0_ID(value)       ((MCAN_RX_ELEMENT_R0_ID_Msk & ((value) «
      MCAN_RX_ELEMENT_R0_ID_Pos)))
00028 #define MCAN_RX_ELEMENT_R0_RTR_Pos         29
00029 #define MCAN_RX_ELEMENT_R0_RTR             (0x1ul « MCAN_RX_ELEMENT_R0_RTR_Pos)
00030 #define MCAN_RX_ELEMENT_R0_XTD_Pos         30
00031 #define MCAN_RX_ELEMENT_R0_XTD             (0x1ul « MCAN_RX_ELEMENT_R0_XTD_Pos)
00032 #define MCAN_RX_ELEMENT_R0_ESI_Pos         31
00033 #define MCAN_RX_ELEMENT_R0_ESI             (0x1ul « MCAN_RX_ELEMENT_R0_ESI_Pos)
00034
00035 /* -------- MCAN_RX_ELEMENT_R1 : (MCAN RX element: 0x01) (R/W 32) Rx Element R1 Configuration --------
      */
00036 typedef union
00037 {
00038     struct
00039     {
00040         /* bit: 0..15   Rx Timestamp */
00041         uint32_t RXTS:16;
00042         /* bit: 16..19  Data Length Code */
00043         uint32_t DLC:4;
00044         /* bit: 20      Bit Rate Switch */
00045         uint32_t BRS:1;
00046         /* bit: 21      FD Format */
00047         uint32_t EDL:1;
00048         /* bit: 22..23  Reserved */
00049         uint32_t :2;
00050         /* bit: 24..30  Filter Index */
00051         uint32_t FIDX:7;
00052         /* bit: 31      Accepted Non-matching Frame */
00053         uint32_t ANMF:1;
00054     } bit;
00055     /* Type used for register access */
00056     uint32_t reg;
00057 } MCAN_RX_ELEMENT_R1_Type;
00058
00059 #define MCAN_RX_ELEMENT_R1_RXTS_Pos        0
00060 #define MCAN_RX_ELEMENT_R1_RXTS_Msk        (0xFFFFul « MCAN_RX_ELEMENT_R1_RXTS_Pos)
00061 #define MCAN_RX_ELEMENT_R1_RXTS(value)     ((MCAN_RX_ELEMENT_R1_RXTS_Msk & ((value) «
      MCAN_RX_ELEMENT_R1_RXTS_Pos)))
00062 #define MCAN_RX_ELEMENT_R1_DLC_Pos         16
00063 #define MCAN_RX_ELEMENT_R1_DLC_Msk         (0xFul « MCAN_RX_ELEMENT_R1_DLC_Pos)
00064 #define MCAN_RX_ELEMENT_R1_DLC(value)      ((MCAN_RX_ELEMENT_R1_DLC_Msk & ((value) «
      MCAN_RX_ELEMENT_R1_DLC_Pos)))
00065 #define MCAN_RX_ELEMENT_R1_BRS_Pos         20
00066 #define MCAN_RX_ELEMENT_R1_BRS             (0x1ul « MCAN_RX_ELEMENT_R1_BRS_Pos)
00067 #define MCAN_RX_ELEMENT_R1_FDF_Pos         21
00068 #define MCAN_RX_ELEMENT_R1_FDF             (0x1ul « MCAN_RX_ELEMENT_R1_FDF_Pos)
00069 #define MCAN_RX_ELEMENT_R1_FIDX_Pos        24
00070 #define MCAN_RX_ELEMENT_R1_FIDX_Msk        (0x7Ful « MCAN_RX_ELEMENT_R1_FIDX_Pos)
00071 #define MCAN_RX_ELEMENT_R1_FIDX(value)     ((MCAN_RX_ELEMENT_R1_FIDX_Msk & ((value) «
      MCAN_RX_ELEMENT_R1_FIDX_Pos)))
00072 #define MCAN_RX_ELEMENT_R1_ANMF_Pos        31
00073 #define MCAN_RX_ELEMENT_R1_ANMF            (0x1ul « MCAN_RX_ELEMENT_R1_ANMF_Pos)
00074
00078 struct mcan_rx_element_buffer
00079 {
00080     __IO MCAN_RX_ELEMENT_R0_Type R0;
00081     __IO MCAN_RX_ELEMENT_R1_Type R1;
00082     uint8_t data[CONF_MCAN_ELEMENT_DATA_SIZE];
00083 };
00084
00085
00086 /* -------- MCAN_TX_ELEMENT_T0 : (MCAN TX element: 0x00) (R/W 32) Tx Element T0 Configuration --------
      */
00087 typedef union
00088 {
```

```
00089     struct
00090     {
00091         /* bit:  0..28  Identifier */
00092         uint32_t ID:29;
00093         /* bit:  29     Remote Transmission Request */
00094         uint32_t RTR:1;
00095         /* bit:  30     Extended Identifier */
00096         uint32_t XTD:1;
00097 #if (SAMV71B || SAME70B || SAMV70B)
00098         /* bit:  31     Error State Indicator */
00099         uint32_t ESI:1;
00100 #else
00101         /* bit:  31     Reserved */
00102         uint32_t :1;
00103 #endif
00104     } bit;
00105     /* Type used for register access */
00106     uint32_t reg;
00107 } MCAN_TX_ELEMENT_T0_Type;
00108
00109 #define MCAN_TX_ELEMENT_T0_EXTENDED_ID_Pos          0
00110 #define MCAN_TX_ELEMENT_T0_EXTENDED_ID_Msk          (0x1FFFFFFFul «
      MCAN_TX_ELEMENT_T0_EXTENDED_ID_Pos)
00111 #define MCAN_TX_ELEMENT_T0_EXTENDED_ID(value)       ((MCAN_TX_ELEMENT_T0_EXTENDED_ID_Msk & ((value) «
      MCAN_TX_ELEMENT_T0_EXTENDED_ID_Pos)))
00112 #define MCAN_TX_ELEMENT_T0_STANDARD_ID_Pos          18
00113 #define MCAN_TX_ELEMENT_T0_STANDARD_ID_Msk          (0x7FFul « MCAN_TX_ELEMENT_T0_STANDARD_ID_Pos)
00114 #define MCAN_TX_ELEMENT_T0_STANDARD_ID(value)       ((MCAN_TX_ELEMENT_T0_STANDARD_ID_Msk & ((value) «
      MCAN_TX_ELEMENT_T0_STANDARD_ID_Pos)))
00115 #define MCAN_TX_ELEMENT_T0_RTR_Pos          29
00116 #define MCAN_TX_ELEMENT_T0_RTR              (0x1ul « MCAN_TX_ELEMENT_T0_RTR_Pos)
00117 #define MCAN_TX_ELEMENT_T0_XTD_Pos          30
00118 #define MCAN_TX_ELEMENT_T0_XTD              (0x1ul « MCAN_TX_ELEMENT_T0_XTD_Pos)
00119 #if (SAMV71B || SAME70B || SAMV70B)
00120     #define MCAN_TX_ELEMENT_T0_ESI_Pos          31
00121     #define MCAN_TX_ELEMENT_T0_ESI              (0x1ul « MCAN_TX_ELEMENT_T0_ESI_Pos)
00122 #endif
00123
00124 /* -------- MCAN_TX_ELEMENT_T1 : (MCAN TX element: 0x01) (R/W 32) Tx Element T1 Configuration --------
      */
00125 typedef union
00126 {
00127     struct
00128     {
00129         /* bit: 0..15   Reserved */
00130         uint32_t :16;
00131         /* bit: 16..19  Data Length Code */
00132         uint32_t DLC:4;
00133 #if (SAMV71B || SAME70B || SAMV70B)
00134         /* bit: 20      Bit Rate Switch */
00135         uint32_t BRS:1;
00136         /* bit: 21      FD Format */
00137         uint32_t FDF:1;
00138         /* bit: 22  Reserved */
00139         uint32_t :1;
00140 #else
00141         /* bit: 20..22  Reserved */
00142         uint32_t :3;
00143 #endif
00144         /* bit: 23      Event FIFO Control */
00145         uint32_t EFCC:1;
00146         /* bit: 24..31  Message Marker */
00147         uint32_t MM:8;
00148     } bit;
00149     /* Type used for register access */
00150     uint32_t reg;
00151 } MCAN_TX_ELEMENT_T1_Type;
00152
00153 #define MCAN_TX_ELEMENT_T1_DLC_Pos          16
00154 #define MCAN_TX_ELEMENT_T1_DLC_Msk          (0xFul « MCAN_TX_ELEMENT_T1_DLC_Pos)
00155 #define MCAN_TX_ELEMENT_T1_DLC(value)       ((MCAN_TX_ELEMENT_T1_DLC_Msk & ((value) «
      MCAN_TX_ELEMENT_T1_DLC_Pos)))
00157 #define MCAN_TX_ELEMENT_T1_DLC_DATA8_Val        0x8ul
00159 #define MCAN_TX_ELEMENT_T1_DLC_DATA12_Val       0x9ul
00161 #define MCAN_TX_ELEMENT_T1_DLC_DATA16_Val       0xAul
00163 #define MCAN_TX_ELEMENT_T1_DLC_DATA20_Val       0xBul
00165 #define MCAN_TX_ELEMENT_T1_DLC_DATA24_Val       0xCul
00167 #define MCAN_TX_ELEMENT_T1_DLC_DATA32_Val       0xDul
00169 #define MCAN_TX_ELEMENT_T1_DLC_DATA48_Val       0xEul
00171 #define MCAN_TX_ELEMENT_T1_DLC_DATA64_Val       0xFul
00172 #if (SAMV71B || SAME70B || SAMV70B)
00173     #define MCAN_TX_ELEMENT_T1_BRS_Pos          20
00174     #define MCAN_TX_ELEMENT_T1_BRS              (0x1ul « MCAN_TX_ELEMENT_T1_BRS_Pos)
00175     #define MCAN_TX_ELEMENT_T1_FDF_Pos          21
00176     #define MCAN_TX_ELEMENT_T1_FDF              (0x1ul « MCAN_TX_ELEMENT_T1_FDF_Pos)
00177 #endif
00178 #define MCAN_TX_ELEMENT_T1_EFC_Pos          23
```

```
00179 #define MCAN_TX_ELEMENT_T1_EFC             (0x1ul << MCAN_TX_ELEMENT_T1_EFC_Pos)
00180 #define MCAN_TX_ELEMENT_T1_MM_Pos          24
00181 #define MCAN_TX_ELEMENT_T1_MM_Msk          (0xFFul << MCAN_TX_ELEMENT_T1_MM_Pos)
00182 #define MCAN_TX_ELEMENT_T1_MM(value)       ((MCAN_TX_ELEMENT_T1_MM_Msk & ((value) <<
      MCAN_TX_ELEMENT_T1_MM_Pos)))
00183
00189 struct mcan_tx_element
00190 {
00191     __IO MCAN_TX_ELEMENT_T0_Type T0;
00192     __IO MCAN_TX_ELEMENT_T1_Type T1;
00193     uint8_t data[CONF_MCAN_ELEMENT_DATA_SIZE];
00194 };
00195
00196 /* -------- MCAN_TX_EVENT_ELEMENT_E0 : (MCAN TX event element: 0x00) (R/W 32) Tx Event Element E0
      Configuration -------- */
00197 typedef union
00198 {
00199     struct
00200     {
00201         /* bit: 0..28  Identifier */
00202         uint32_t ID:29;
00203         /* bit: 29     Remote Transmission Request */
00204         uint32_t RTR:1;
00205         /* bit: 30     Extended Identifier */
00206         uint32_t XTD:1;
00207         /* bit: 31     Error State Indicator */
00208         uint32_t ESI:1;
00209     } bit;
00210     /* Type used for register access */
00211     uint32_t reg;
00212 } MCAN_TX_EVENT_ELEMENT_E0_Type;
00213
00214 #define MCAN_TX_EVENT_ELEMENT_E0_ID_Pos          0
00215 #define MCAN_TX_EVENT_ELEMENT_E0_ID_Msk          (0x1FFFFFFFul << MCAN_TX_EVENT_ELEMENT_E0_ID_Pos)
00216 #define MCAN_TX_EVENT_ELEMENT_E0_ID(value)       ((MCAN_TX_EVENT_ELEMENT_E0_ID_Msk & ((value) <<
      MCAN_TX_EVENT_ELEMENT_E0_ID_Pos)))
00217 #define MCAN_TX_EVENT_ELEMENT_E0_RTR_Pos         29
00218 #define MCAN_TX_EVENT_ELEMENT_E0_RTR             (0x1ul << MCAN_TX_EVENT_ELEMENT_E0_RTR_Pos)
00219 #define MCAN_TX_EVENT_ELEMENT_E0_XTD_Pos         30
00220 #define MCAN_TX_EVENT_ELEMENT_E0_XTD             (0x1ul << MCAN_TX_EVENT_ELEMENT_E0_XTD_Pos)
00221 #define MCAN_TX_EVENT_ELEMENT_E0_ESI_Pos         31
00222 #define MCAN_TX_EVENT_ELEMENT_E0_ESI             (0x1ul << MCAN_TX_EVENT_ELEMENT_E0_ESI_Pos)
00223
00224 /* -------- MCAN_TX_EVENT_ELEMENT_E1 : (MCAN TX event element: 0x01) (R/W 32) Tx Event Element E1
      Configuration -------- */
00225 typedef union
00226 {
00227     struct
00228     {
00229         /* bit: 0..15   Tx Timestamp */
00230         uint32_t TXTS:16;
00231         /* bit: 16..19  Data Length Code */
00232         uint32_t DLC:4;
00233         /* bit: 20      Bit Rate Switch */
00234         uint32_t BRS:1;
00235         /* bit: 21      FD Format */
00236         uint32_t EDL:1;
00237         /* bit: 22..23  Event Type */
00238         uint32_t ET:2;
00239         /* bit: 24..31  Message Marker */
00240         uint32_t MM:8;
00241     } bit;
00242     /* Type used for register access */
00243     uint32_t reg;
00244 } MCAN_TX_EVENT_ELEMENT_E1_Type;
00245
00246 #define MCAN_TX_EVENT_ELEMENT_E1_TXTS_Pos        0
00247 #define MCAN_TX_EVENT_ELEMENT_E1_TXTS_Msk        (0xFFFFul << MCAN_TX_EVENT_ELEMENT_E1_TXTS_Pos)
00248 #define MCAN_TX_EVENT_ELEMENT_E1_TXTS(value)     ((MCAN_TX_EVENT_ELEMENT_E1_TXTS_Msk & ((value) <<
      MCAN_TX_EVENT_ELEMENT_E1_TXTS_Pos)))
00249 #define MCAN_TX_EVENT_ELEMENT_E1_DLC_Pos         16
00250 #define MCAN_TX_EVENT_ELEMENT_E1_DLC_Msk         (0xFul << MCAN_TX_EVENT_ELEMENT_E1_DLC_Pos)
00251 #define MCAN_TX_EVENT_ELEMENT_E1_DLC(value)      ((MCAN_TX_EVENT_ELEMENT_E1_DLC_Msk & ((value) <<
      MCAN_TX_EVENT_ELEMENT_E1_DLC_Pos)))
00252 #define MCAN_TX_EVENT_ELEMENT_E1_BRS_Pos         20
00253 #define MCAN_TX_EVENT_ELEMENT_E1_BRS             (0x1ul << MCAN_TX_EVENT_ELEMENT_E1_BRS_Pos)
00254 #define MCAN_TX_EVENT_ELEMENT_E1_FDF_Pos         21
00255 #define MCAN_TX_EVENT_ELEMENT_E1_FDF             (0x1ul << MCAN_TX_EVENT_ELEMENT_E1_FDF_Pos)
00256 #define MCAN_TX_EVENT_ELEMENT_E1_ET_Pos          22
00257 #define MCAN_TX_EVENT_ELEMENT_E1_ET_Msk          (0x3ul << MCAN_TX_EVENT_ELEMENT_E1_ET_Pos)
00258 #define MCAN_TX_EVENT_ELEMENT_E1_ET(value)       ((MCAN_TX_EVENT_ELEMENT_E1_ET_Msk & ((value) <<
      MCAN_TX_EVENT_ELEMENT_E1_ET_Pos)))
00259 #define MCAN_TX_EVENT_ELEMENT_E1_MM_Pos          24
00260 #define MCAN_TX_EVENT_ELEMENT_E1_MM_Msk          (0xFFul << MCAN_TX_EVENT_ELEMENT_E1_MM_Pos)
00261 #define MCAN_TX_EVENT_ELEMENT_E1_MM(value)       ((MCAN_TX_EVENT_ELEMENT_E1_MM_Msk & ((value) <<
      MCAN_TX_EVENT_ELEMENT_E1_MM_Pos)))
00262
```

```
00268 struct mcan_tx_event_element
00269 {
00270     __IO MCAN_TX_EVENT_ELEMENT_E0_Type E0;
00271     __IO MCAN_TX_EVENT_ELEMENT_E1_Type E1;
00272 };
00273
00274 /* -------- MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0 : (MCAN standard message ID filter element: 0x00)
      (R/W 32) Standard Message ID Filter Element S0 Configuration -------- */
00275 typedef union
00276 {
00277     struct
00278     {
00279         /* bit: 0..10   Standard Filter ID 2 */
00280         uint32_t SFID2:11;
00281         /* bit: 11..15  Reserved */
00282         uint32_t :5;
00283         /* bit: 16..26  Standard Filter ID 1 */
00284         uint32_t SFID1:11;
00285         /* bit: 27..29  Standard Filter Element Configuration */
00286         uint32_t SFEC:3;
00287         /* bit: 30..31  Standard Filter Type */
00288         uint32_t SFT:2;
00289     } bit;
00290     /* Type used for register access */
00291     uint32_t reg;
00292 } MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_Type;
00293
00294 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID2_Pos          0
00295 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID2_Msk          (0x7FFul «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID2_Pos)
00296 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID2(value)
      ((MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID2_Msk & ((value) «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID2_Pos)))
00297 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID1_Pos          16
00298 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID1_Msk          (0x7FFul «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID1_Pos)
00299 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID1(value)
      ((MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID1_Msk & ((value) «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFID1_Pos)))
00300 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_Pos           27
00301 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_Msk           (0x7ul «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_Pos)
00302 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC(value)
      ((MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_Msk & ((value) «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_Pos)))
00303 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_DISABLE_Val     0
00304 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_STF0M_Val       1
00305 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_STF1M_Val       2
00306 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_REJECT_Val      3
00307 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_PRIORITY_Val    4
00308 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_PRIF0M_Val      5
00309 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_PRIF1M_Val      6
00310 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFEC_STRXBUF_Val     7
00311 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_Pos            30
00312 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_Msk            (0x3ul «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_Pos)
00313 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT(value)
      ((MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_Msk & ((value) «
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_Pos)))
00314 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_RANGE
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT(0)
00315 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_DUAL
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT(1)
00316 #define MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT_CLASSIC
      MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_SFT(2)
00317
00326 class mcan_module;
00327 class mcan_standard_message_filter_element
00328 {
00329 private:
00330         __IO MCAN_STANDARD_MESSAGE_FILTER_ELEMENT_S0_Type S0;
00331 public:
00332     mcan_standard_message_filter_element() {S0.reg=0;};
00333     mcan_standard_message_filter_element(int id, int mask=0x7FF)
00334     {
00335      set_id_mask(id,mask);
00336     }
00337     void set_id_mask(int id, int mask=0x7FF);
00338     void set_exact(int id1, int id2);
00339     inline void set_exact(int id1) {set_id_mask(id1); };
00340     friend mcan_module;
00341 };
00342
00343
00344
00345
00346
```

```
00347
00348
00349
00350
00351 /* -------- MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0 : (MCAN extended message ID filter element: 0x00)
      (R/W 32) Extended Message ID Filter Element F0 Configuration -------- */
00352 typedef union
00353 {
00354     struct
00355     {
00356         /* bit: 0..28   Extended Filter ID 1 */
00357         uint32_t EFID1:29;
00358         /* bit: 29..31  Extended Filter Element Configuration */
00359         uint32_t EFEC:3;
00360     } bit;
00361     /* Type used for register access */
00362     uint32_t reg;
00363 } MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_Type;
00364
00365 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFID1_Pos          0
00366 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFID1_Msk          (0x1FFFFFFFul «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFID1_Pos)
00367 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFID1(value)
      ((MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFID1_Msk & ((value) «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFID1_Pos)))
00368 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_Pos           29
00369 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_Msk           (0x7ul «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_Pos)
00370 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC(value)
      ((MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_Msk & ((value) «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_Pos)))
00371 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_DISABLE_Val       0
00372 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_STF0M_Val         1
00373 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_STF1M_Val         2
00374 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_REJECT_Val        3
00375 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_PRIORITY_Val      4
00376 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_PRIF0M_Val        5
00377 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_PRIF1M_Val        6
00378 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_EFEC_STRXBUF_Val       7
00379
00380 /* -------- MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1 : (MCAN extended message ID filter element: 0x01)
      (R/W 32) Extended Message ID Filter Element F1 Configuration -------- */
00381 typedef union
00382 {
00383     struct
00384     {
00385         /* bit: 0..28  Extended Filter ID 2 */
00386         uint32_t EFID2:29;
00387         /* bit: 29     Reserved */
00388         uint32_t :1;
00389         /* bit: 30..31 Extended Filter Type */
00390         uint32_t EFT:2;
00391     } bit;
00392     /* Type used for register access */
00393     uint32_t reg;
00394 } MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_Type;
00395
00396 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFID2_Pos          0
00397 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFID2_Msk          (0x1FFFFFFFul «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFID2_Pos)
00398 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFID2(value)
      ((MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFID2_Msk & ((value) «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFID2_Pos)))
00399 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_Pos            30
00400 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_Msk            (0x3ul «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_Pos)
00401 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT(value)
      ((MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_Msk & ((value) «
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_Pos)))
00402 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_RANGEM
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT(0)
00403 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_DUAL
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT(1)
00404 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_CLASSIC
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT(2)
00405 #define MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT_RANGE
      MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_EFT(3)
00406
00412 class mcan_extended_message_filter_element
00413 {
00414 private:
00415     __IO MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F0_Type F0;
00416     __IO MCAN_EXTENDED_MESSAGE_FILTER_ELEMENT_F1_Type F1;
00417 public:
00418     mcan_extended_message_filter_element() {F0.reg=0; F1.reg=0; };
00419     mcan_extended_message_filter_element(int id,int mask=0x1FFFFFFF)
00420     {
```

```
00421       set_id_mask(id,mask);
00422     }
00423     void set_id_mask(int id, int mask=0x1FFFFFFF);
00424     void set_exact(int id1, int id2);
00425     inline void set_exact(int id1) {set_id_mask(id1); };
00426  friend mcan_module;
00427 };
00428 /* @} */
00429
00430
00431 enum status_code {
00432     STATUS_OK               =  0,
00433     STATUS_ERR_BUSY         =  0x19,
00434     STATUS_ERR_DENIED       =  0x1C,
00435     STATUS_ERR_TIMEOUT      =  0x12,
00436     ERR_IO_ERROR            =  -1,
00437     ERR_FLUSHED             =  -2,
00438     ERR_TIMEOUT             =  -3,
00439     ERR_BAD_DATA            =  -4,
00440     ERR_PROTOCOL            =  -5,
00441     ERR_UNSUPPORTED_DEV     =  -6,
00442     ERR_NO_MEMORY           =  -7,
00443     ERR_INVALID_ARG         =  -8,
00444     ERR_BAD_ADDRESS         =  -9,
00445     ERR_BUSY                =  -10,
00446     ERR_BAD_FORMAT          =  -11,
00447     ERR_NO_TIMER            =  -12,
00448     ERR_TIMER_ALREADY_RUNNING   =  -13,
00449     ERR_TIMER_NOT_RUNNING   =  -14,
00450     ERR_ABORTED             =  -15,
00460     OPERATION_IN_PROGRESS   = -128,
00461 };
00462
00463
00464
00541 enum mcan_timeout_mode
00542 {
00544     MCAN_TIMEOUT_CONTINUES = MCAN_TOCC_TOS_CONTINUOUS,
00546     MCAN_TIMEOUT_TX_EVEN_FIFO = MCAN_TOCC_TOS_TX_EV_TIMEOUT,
00548     MCAN_TIMEOUT_RX_FIFO_0 = MCAN_TOCC_TOS_RX0_EV_TIMEOUT,
00550     MCAN_TIMEOUT_RX_FIFO_1 = MCAN_TOCC_TOS_RX1_EV_TIMEOUT,
00551 };
00552
00553
00557 enum mcan_nonmatching_frames_action
00558 {
00560     MCAN_NONMATCHING_FRAMES_FIFO_0,
00562     MCAN_NONMATCHING_FRAMES_FIFO_1,
00564     MCAN_NONMATCHING_FRAMES_REJECT,
00565 };
00566
00567
00568
00574 enum mcan_interrupt_source
00575 {
00577     MCAN_RX_FIFO_0_NEW_MESSAGE = MCAN_IE_RF0NE,
00579     MCAN_RX_FIFO_0_WATERMARK = MCAN_IE_RF0WE,
00581     MCAN_RX_FIFO_0_FULL = MCAN_IE_RF0FE,
00583     MCAN_RX_FIFO_0_LOST_MESSAGE = MCAN_IE_RF0LE,
00585     MCAN_RX_FIFO_1_NEW_MESSAGE = MCAN_IE_RF1NE,
00587     MCAN_RX_FIFO_1_WATERMARK = MCAN_IE_RF1WE,
00589     MCAN_RX_FIFO_1_FULL = MCAN_IE_RF1FE,
00591     MCAN_RX_FIFO_1_MESSAGE_LOST = MCAN_IE_RF1LE,
00593     MCAN_RX_HIGH_PRIORITY_MESSAGE = MCAN_IE_HPME,
00595     MCAN_TIMESTAMP_COMPLETE = MCAN_IE_TCE,
00597     MCAN_TX_CANCELLATION_FINISH = MCAN_IE_TCFE,
00599     MCAN_TX_FIFO_EMPTY = MCAN_IE_TFEE,
00601     MCAN_TX_EVENT_FIFO_NEW_ENTRY = MCAN_IE_TEFNE,
00603     MCAN_TX_EVENT_FIFO_WATERMARK = MCAN_IE_TEFWE,
00605     MCAN_TX_EVENT_FIFO_FULL = MCAN_IE_TEFFE,
00607     MCAN_TX_EVENT_FIFO_ELEMENT_LOST = MCAN_IE_TEFLE,
00609     MCAN_TIMESTAMP_WRAPAROUND = MCAN_IE_TSWE,
00611     MCAN_MESSAGE_RAM_ACCESS_FAILURE = MCAN_IE_MRAFE,
00613     MCAN_TIMEOUT_OCCURRED = MCAN_IE_TOOE,
00615     MCAN_RX_BUFFER_NEW_MESSAGE = MCAN_IE_DRXE,
00617     MCAN_ERROR_LOGGING_OVERFLOW = MCAN_IE_ELOE,
00619     MCAN_ERROR_PASSIVE = MCAN_IE_EPE,
00621     MCAN_WARNING_STATUS = MCAN_IE_EWE,
00623     MCAN_BUS_OFF = MCAN_IE_BOE,
00625     MCAN_WATCHDOG = MCAN_IE_WDIE,
00627     MCAN_CRC_ERROR = MCAN_IE_CRCEE,
00629     MCAN_BIT_ERROR = MCAN_IE_BEE,
00631     MCAN_ACKNOWLEDGE_ERROR = MCAN_IE_ACKEE,
00633     MCAN_FORMAT_ERROR = MCAN_IE_FOEE,
00635     MCAN_STUFF_ERROR = MCAN_IE_STEE
00636 };
00640 struct tx_record
```

```
00641 {
00642 OS_SEM * pSem;
00643 volatile uint32_t when;
00644 };
00645
00646 #endif // __MCAN_INTERNAL_H__
00647
```

24.49 pwm.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef  _PWM_H_INC
00006 #define  _PWM_H_INC
00007
00008 #include <predef.h>
00009 #include <stdio.h>
00010 #include <nbrtos.h>
00011 #include <sim.h>
00012
00013 #ifndef __cplusplus
00014 #error PWM driver is a C++ only library
00015 #endif
00016
00017 #define DEFAULT_PWM_MODULE 0
00018 #define PWM_MODULE_COUNT   2
00019 #define PWM_CHANNEL_COUNT  4
00020
00021 #define PWM_CLOCK_DIV_MAX  256
00022 #define PWM_CLOCK_PRE_MAX  11
00023
00024 #define __DEBUG_PWM
00025
00026 extern volatile uint32_t PERIPH_CLOCK;
00027
00028 // 49.5.3 PWM interrupt
00029
00030 enum pwmChannelNumber {
00031     PWM_CHANNEL_0        = 0,
00032     PWM_CHANNEL_1        = 1,
00033     PWM_CHANNEL_2        = 2,
00034     PWM_CHANNEL_3        = 3,
00035 };
00036
00037 enum pwmAlignment {
00038     PWM_ALIGN_LEFT       = 0,
00039     PWM_ALIGN_CENTER     = 1,
00040 };
00041
00042 enum pwmPolarity {
00043     PWM_LOW_LEVEL        = 0,
00044     PWM_HIGH_LEVEL       = 1,
00045 };
00046
00047 typedef struct
00048 {
00049     uint32_t m_reg_PWM_CMR;
00050     uint32_t m_frequency;
00051     uint32_t m_dutyCycle;
00052     uint32_t m_period;
00053     uint8_t  m_alignment;
00054     uint8_t  m_polarity;
00055     uint32_t m_clockSource; // clock A, clock B, or master clock/peripheral clock
00056     // counter event
00057     // deadtime
00058 } pwmChannelStruct;
00059
00060
00061 class PWMModule
00062 {
00063 public:
00064
00065 protected:
00066     static  PWMModule *lastCtxs[PWM_MODULE_COUNT];
00067     // static  spiDriverStruct driverCtx[PWM_MODULE_COUNT];
00068
00069     // PWM Module specific
00070     uint8_t  m_moduleNum;
00071     uint32_t m_reg_PWM_CLK;
00074     // polarity
00075
00076     // Channel Specific
00077     // static pwmChannelStruct pwmChannels[PWM_CHANNEL_COUNT];
```

```
00078     static pwmChannelStruct pwmChannels[PWM_MODULE_COUNT][PWM_CHANNEL_COUNT];
00079     // deadtime
00080
00081     /* Additional members pwm_channel_t (Microchip source): */
00082     // pwm_counter_event_t counter_event;  /** Channel counter event */
00083     // bool b_deadtime_generator;          /** Boolean of channel dead-time generator */
00084     // bool b_pwmh_output_inverted;        /** Boolean of channel dead-time PWMH output inverted */
00085     // bool b_pwml_output_inverted;        /** Boolean of channel dead-time PWML output inverted */
00086     // uint16_t us_deadtime_pwmh;          /** Dead-time Value for PWMH Output */
00087     // uint16_t us_deadtime_pwml;          /** Dead-time Value for PWML Output */
00088     // pwm_output_t output_selection;      /** Channel output */
00089     // bool b_sync_ch;                     /** Boolean of Synchronous Channel */
00090     // pwm_fault_id_t fault_id;            /** Fault ID of the channel */
00091     // pwm_level_t ul_fault_output_pwmh;   /** Channel PWMH output level in fault protection */
00092     // pwm_level_t ul_fault_output_pwml;   /** Channel PWML output level in fault protection */
00093
00094     // uint32_t ul_spread;                                      /** Spread Spectrum Value */
00095     // pwm_spread_spectrum_mode_t spread_spectrum_mode;         /** Spread Spectrum Mode */
00096     // uint32_t ul_leading_edge_delay;                          /** Leading Edge Value */
00097     // pwm_leading_edge_blanking_mode_t leading_edge_blanking_mode;    /** Leading Edge Mode */
00098     // uint32_t ul_ppm_mode;                                    /** PPM Mode in Channel mode */
00099     /* End Additional members pwm_channel_t (Microchip source): */
00100
00101
00102
00103     // uint32_t    m_regMR;
00104     // uint32_t    m_regCSR;
00105     // OS_SEM      *m_finishedSem;
00106     // uint32_t    m_busSpeed;
00107     // uint8_t     m_CSNum;
00108     // volatile bool m_inProgress;
00109
00110     // void ReadyHW();
00111     // spiDriverStruct *getCtx() { return driverCtx + m_moduleNum; }
00112     inline Pwm   * pwm() { return PWM0 + m_moduleNum; }
00113     inline pwmChannelStruct * getChannelStruct(uint8_t channelNum) { return
      &(pwmChannels[m_moduleNum][channelNum]); }
00114
00115 public:
00116
00117 #ifdef __DEBUG_PWM
00118     void dumpRegs();
00119     void dumpChannelStruct(uint8_t channelNum);
00120 #endif
00121
00122      PWMModule( uint8_t PWMModule = DEFAULT_PWM_MODULE);
00123
00124      uint8_t Init();
00125      uint32_t ConfigureChannel(uint8_t channelNum, uint32_t frequency,
00126         uint8_t alignment = PWM_ALIGN_LEFT, uint8_t polarity = PWM_LOW_LEVEL,
00127         uint32_t period = 100, uint32_t dutyCycle = 50);
00128      uint8_t EnableChannel(uint8_t channelNum);
00129      uint8_t DisableChannel(uint8_t channelNum);
00130      uint32_t SetFrequency(uint8_t channelNum, uint32_t frequency);
00131      uint32_t GetActualFrequency(uint8_t channelNum);
00132      bool ChannelIsActive(uint8_t channelNum);
00133
00134 // EnableDeadtime(uint8_t channelNum)
00135 // DisableDeadtime(uint8_t channelNum)
00136 // uint8_t GetAvailableChannel() ?
00137  };
00138
00139
00140  #endif   /* ----- #ifndef _PWM_H_INC  ----- */
```

24.50 quadspi.h File Reference

NetBurner DMA Quad SPI (QSPI) API for ARM SAME70 (MODM7AE70)

```
#include <nbrtos.h>
#include <basictypes.h>
#include <xdmac.h>
#include <dspi.h>
```

Classes

- class SPI_QSPI

  The Single-Bit SPI mode QSPI Peripheral Class.

Macros

- #define **DEFAULT_QUADSPI_MODULE** 0

  Default QUADSPI module.
- #define **QUADSPI_MODULE_COUNT** 1

  Number of modules: 0, 1.

24.50.1 Detailed Description

NetBurner DMA Quad SPI (QSPI) API for ARM SAME70 (MODM7AE70)

24.51 quadspi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00023 #ifndef  _DMA_QUADSPI_H_INC
00024 #define  _DMA_QUADSPI_H_INC
00025
00026 #include <nbrtos.h>
00027 #include <basictypes.h>
00028 #include <xdmac.h>
00029 #include <dspi.h>
00030
00031 #ifndef __cplusplus
00032 #error QuadSPI driver is a C++ only library
00033 #endif
00034
00035 // #define __DEBUG_QUADSPI     1
00036
00037
00043 #define DEFAULT_QUADSPI_MODULE 0
00044 #define QUADSPI_MODULE_COUNT   1
00053 class SPI_QSPI : public SPI_SPI
00054 {
00055 protected:
00056     static  SPI_QSPI *lastQuadSPICtxs[QUADSPI_MODULE_COUNT];
00057     static  spiDriverStruct QuadSPIDriverCtx[QUADSPI_MODULE_COUNT];
00058
00059 private:
00060     uint32_t    m_regSCR;
00061
00062     virtual void ReadyHW() override;
00063     virtual spiDriverStruct *getCtx() override { return QuadSPIDriverCtx + m_moduleNum; }
00064     inline Qspi   * quadSpi() { return QSPI + m_moduleNum; }
00065
00066 public:
00067
00068 #ifdef __DEBUG_QUADSPI
00069     virtual void    dumpRegs() override;
00070 #endif
00071
00072
00081     SPI_QSPI( uint8_t QuadSPIModule = DEFAULT_QUADSPI_MODULE);
00082
00103     SPI_QSPI( uint8_t QuadSPIModule, uint32_t baudRateInBps,
00104             uint8_t transferSizeInBits = 8, uint8_t peripheralChipSelects = 0x00,
00105             uint8_t chipSelectPolarity = 0x0F, uint8_t clockPolarity = 0,
00106             uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00107             uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 );
00108
00127     virtual uint8_t Init( uint32_t baudRateInBps = 2000000,
00128             uint8_t transferSizeInBits = 8, uint8_t peripheralChipSelects = 0x00,
00129             uint8_t chipSelectPolarity = 0x0F, uint8_t clockPolarity = 0,
00130             uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00131             uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 ) override;
00132
00142     virtual uint32_t SetBusSpeed(uint32_t maxSpeed) override;
00143
00159     virtual uint8_t Start( uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr,
00160                 uint32_t byteCount, int csReturnToInactive = DEASSERT_AFTER_LAST ) override;
00161
00171     virtual inline uint8_t Tx( uint8_t *transmitBufferPtr, uint32_t byteCount,
00172                     int csReturnToInactive = DEASSERT_AFTER_LAST ) override
00173     { return Start(transmitBufferPtr, NULL, byteCount, csReturnToInactive); }
00174
00184     inline uint8_t Rx( uint8_t *receiveBufferPtr, uint32_t byteCount, int csReturnToInactive =
      DEASSERT_AFTER_LAST )
```

```
00185     { return Start(NULL, receiveBufferPtr, byteCount, csReturnToInactive); }
00186
00187     // The SAME70 always uses DMA. These functions kept here for reference to other platforms
00188     //    bool EnableDMA(bool enableDMA = true);
00189     //    inline bool DisableDMA() { return EnableDMA(false); }
00190
00191     /*
00192      * @brief Register a semaphore for the SPI module.
00193      *
00194      * The SPI module will post to this semaphore when a transaction is complete
00195      *
00196      * @param finishedSem  Pointer to the semaphore
00197      *
00198      * @return true if the registration was successful, false if a SPI transaction is in progress
00199      */
00200     // bool RegisterSem( OS_SEM *finishedSem );
00201
00202     /*
00203      * @brief Clear a semaphore registration
00204      *
00205      * @return true if the clear was successful, false if a SPI transaction is in progress
00206      */
00207     // inline bool ClrSem() { return RegisterSem(NULL); }
00208
00209     /*
00210      * @brief Obtain a pointer to the SPI finished semaphore
00211      *
00212      * @return A pointer to the semaphore
00213      */
00214     // inline OS_SEM * GetSem() { return m_finishedSem; }
00215
00216     /*
00217      * @brief Function to check SPI status
00218      *
00219      * Called as a class method on a specific SPI object. For example: MySpi.Done()
00220      *
00221      * @return true if SPI is finished, false if active
00222      */
00223     // inline bool Done() { return !m_inProgress; }
00224     // virtual inline bool Done() { return !m_inProgress; }
00225
00226     /*
00227      * @brief Returns the active baud rate
00228      *
00229      * The baud rate will be set to the value specified when the SPI module is initialized.
00230      * If that value is not possible, the next lowest baud rate will be chosen.
00231      *
00232      * @return m_busSpeed The actual SPI module baud rate
00233      */
00234     // inline uint32_t GetActualBaudrate() { return m_busSpeed; }
00235
00246     virtual inline bool SetCS( uint8_t CS ) override
00247     {
00248         OSLockObj lock;
00249         return true;
00250     }
00251
00252     // The ISR used by the QuadSPI driver. Internal use only (qspi.cpp).
00253     friend void QuadSPI_DMA_Isr(XdmaCh_t *dma, int module);
00254 };
00255
00256 #endif   /* ----- #ifndef _DMA_QUADSPI_H_INC  ----- */
00257
```

24.52 same70.h File Reference

24.52.1 Detailed Description

Copyright (c) 2015 Atmel Corporation. All rights reserved.
License
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.

4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

24.53 same70.h

Go to the documentation of this file.

```
00001
00036 /*
00037  * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel Support</a>
00038  */
00039 #ifndef _SAME70_
00040 #define _SAME70_
00041
00042 #if defined (__SAME70J19__)
00043 #include "same70j19.h"
00044 #elif defined (__SAME70J20__)
00045 #include "same70j20.h"
00046 #elif defined (__SAME70J21__)
00047 #include "same70j21.h"
00048 #elif defined (__SAME70N19__)
00049 #include "same70n19.h"
00050 #elif defined (__SAME70N20__)
00051 #include "same70n20.h"
00052 #elif defined (__SAME70N21__)
00053 #include "same70n21.h"
00054 #elif defined (__SAME70Q19__)
00055 #include "same70q19.h"
00056 #elif defined (__SAME70Q20__)
00057 #include "same70q20.h"
00058 #elif defined (__SAME70Q21__)
00059 #include "same70q21.h"
00060 #else
00061   #error Library does not support the specified device.
00062 #endif
00063
00064 #define CPU_MAX_IRQ                 (8)
00065
00066 #define MPU_REGION_NULL_POINTERS    (15)
00067 #define MPU_REGION_BUFF_POOL        (14)
00068 #define MPU_REGION_FAST_BUFF_POOL   (13)
00069 #define MPU_REGION_NO_CACHE_SRAM    (12)
00070 #define MPU_REGION_DRAM             (11)
00071 #define MPU_REGION_EBI_PAGING       (7)
00072 #define MPU_REGION_EBI_WT           (6)
00073 #define MPU_REGION_DRAM_UNUSED      (5)
00074 #define MPU_REGION_DRAM_CACHED      (4)
00075 #define MPU_REGION_FLASH_ORDER      (3)
00076
00077 #endif /* _SAME70_ */
```

24.54 same70_serial.h

```
00001 #ifndef  __SERIAL_H
00002 #define  __SERIAL_H
00003
00004 #include <stdint.h>
00005
00006 #define USART_DMA_RX_BUF_LEN 256
00007 #define UART_DMA_RX_BUF_LEN  256
00008
00009 #define DMA_RX_MAX_LATENCY_MS 14
00010
00011 // Do not configure this particular pin. Typically used for signals that can
00012 // come out in multiple locations that need to be specified by the developer
```

```
00013 #define SER_IO_NO_CONF 0xFF
00014
00015 struct configMap {
00016     enum mode_t {
00017         MODE_A = 0,
00018         MODE_B = 1,
00019         MODE_C = 2,
00020         MODE_D = 3,
00021         MODE_GPIO = 4
00022     };
00023     uint8_t pio;            // pinio module index
00024     uint8_t io;             // pinio number
00025     mode_t mode;            // peripheral function, A, B, C, D, GPIO
00026     uint8_t out;            // set to 1 if signal is an output
00027 };
00028
00029
00030 void ser_putchar(char c);
00031 char ser_getchar();
00032 void ser_putstring(char *s);
00033 void ser_putbyte(uint8_t b);
00034 void ser_putshort(uint16_t hw);
00035 void ser_putword(uint32_t w);
00036 void ser_dump(uint32_t *p, int n);
00037
00038 #endif   /* ----- #ifndef __SERIAL_H  ----- */
```

24.55 same70_wdt.h

```
00001 #ifndef  __SAME70_WDT_H
00002 #define  __SAME70_WDT_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 void ServiceBootWatchdog();
00008
00009 void DisableWatchdog();
00010
00011 #endif   /* ----- #ifndef __SAME70_WDT_H  ----- */
```

24.56 same70q21_sim.h File Reference

```
#include "component/acc.h"
#include "component/aes.h"
#include "component/afec.h"
#include "component/chipid.h"
#include "component/dacc.h"
#include "component/efc.h"
#include "component/gmac.h"
#include "component/gpbr.h"
#include "component/hsmci.h"
#include "component/icm.h"
#include "component/isi.h"
#include "component/matrix.h"
#include "component/mcan.h"
#include "component/pio.h"
#include "component/pmc.h"
#include "component/pwm.h"
#include "component/qspi.h"
#include "component/rstc.h"
#include "component/rswdt.h"
#include "component/rtc.h"
#include "component/rtt.h"
#include "component/sdramc.h"
#include "component/smc.h"
#include "component/spi.h"
#include "component/ssc.h"
#include "component/supc.h"
```

```
#include "component/tc.h"
#include "component/trng.h"
#include "component/twihs.h"
#include "component/uart.h"
#include "component/usart.h"
#include "component/usbhs.h"
#include "component/utmi.h"
#include "component/wdt.h"
#include "component/xdmac.h"
#include "instance/hsmci.h"
#include "instance/ssc.h"
#include "instance/spi0.h"
#include "instance/tc0.h"
#include "instance/tc1.h"
#include "instance/tc2.h"
#include "instance/twihs0.h"
#include "instance/twihs1.h"
#include "instance/pwm0.h"
#include "instance/usart0.h"
#include "instance/usart1.h"
#include "instance/usart2.h"
#include "instance/mcan0.h"
#include "instance/mcan1.h"
#include "instance/usbhs.h"
#include "instance/afec0.h"
#include "instance/dacc.h"
#include "instance/acc.h"
#include "instance/icm.h"
#include "instance/isi.h"
#include "instance/gmac.h"
#include "instance/tc3.h"
#include "instance/spi1.h"
#include "instance/pwm1.h"
#include "instance/twihs2.h"
#include "instance/afec1.h"
#include "instance/aes.h"
#include "instance/trng.h"
#include "instance/xdmac.h"
#include "instance/qspi.h"
#include "instance/smc.h"
#include "instance/sdramc.h"
#include "instance/matrix.h"
#include "instance/utmi.h"
#include "instance/pmc.h"
#include "instance/uart0.h"
#include "instance/chipid.h"
#include "instance/uart1.h"
#include "instance/efc.h"
#include "instance/pioa.h"
#include "instance/piob.h"
#include "instance/pioc.h"
#include "instance/piod.h"
#include "instance/pioe.h"
#include "instance/rstc.h"
#include "instance/supc.h"
#include "instance/rtt.h"
#include "instance/wdt.h"
#include "instance/rtc.h"
```

```
#include "instance/gpbr.h"
#include "instance/rswdt.h"
#include "instance/uart2.h"
#include "instance/uart3.h"
#include "instance/uart4.h"
#include "pio/same70q21.h"
```

Macros

- #define **HSMCI** ((Hsmci *)0x40000000U)
  (HSMCI) Base Address
- #define **SSC** ((Ssc *)0x40004000U)
  (SSC) Base Address
- #define **SPI0** ((Spi *)0x40008000U)
  (SPI0) Base Address
- #define **TC0** ((Tc *)0x4000C000U)
  (TC0) Base Address
- #define **TC1** ((Tc *)0x40010000U)
  (TC1) Base Address
- #define **TC2** ((Tc *)0x40014000U)
  (TC2) Base Address
- #define **TWIHS0** ((Twihs *)0x40018000U)
  (TWIHS0) Base Address
- #define **TWIHS1** ((Twihs *)0x4001C000U)
  (TWIHS1) Base Address
- #define **PWM0** ((Pwm *)0x40020000U)
  (PWM0) Base Address
- #define **USART0** ((Usart *)0x40024000U)
  (USART0) Base Address
- #define **USART1** ((Usart *)0x40028000U)
  (USART1) Base Address
- #define **USART2** ((Usart *)0x4002C000U)
  (USART2) Base Address
- #define **MCAN0** ((Mcan *)0x40030000U)
  (MCAN0) Base Address
- #define **MCAN1** ((Mcan *)0x40034000U)
  (MCAN1) Base Address
- #define **USBHS** ((Usbhs *)0x40038000U)
  (USBHS) Base Address
- #define **AFEC0** ((Afec *)0x4003C000U)
  (AFEC0) Base Address
- #define **DACC** ((Dacc *)0x40040000U)
  (DACC) Base Address
- #define **ACC** ((Acc *)0x40044000U)
  (ACC) Base Address
- #define **ICM** ((Icm *)0x40048000U)
  (ICM) Base Address
- #define **ISI** ((Isi *)0x4004C000U)
  (ISI) Base Address
- #define **GMAC** ((Gmac *)0x40050000U)
  (GMAC) Base Address

- #define **TC3** ((Tc ∗)0x40054000U)
  (TC3) Base Address
- #define **SPI1** ((Spi ∗)0x40058000U)
  (SPI1) Base Address
- #define **PWM1** ((Pwm ∗)0x4005C000U)
  (PWM1) Base Address
- #define **TWIHS2** ((Twihs ∗)0x40060000U)
  (TWIHS2) Base Address
- #define **AFEC1** ((Afec ∗)0x40064000U)
  (AFEC1) Base Address
- #define **AES** ((Aes ∗)0x4006C000U)
  (AES) Base Address
- #define **TRNG** ((Trng ∗)0x40070000U)
  (TRNG) Base Address
- #define **XDMAC** ((Xdmac ∗)0x40078000U)
  (XDMAC) Base Address
- #define **QSPI** ((Qspi ∗)0x4007C000U)
  (QSPI) Base Address
- #define **SMC** ((Smc ∗)0x40080000U)
  (SMC) Base Address
- #define **SDRAMC** ((Sdramc ∗)0x40084000U)
  (SDRAMC) Base Address
- #define **MATRIX** ((Matrix ∗)0x40088000U)
  (MATRIX) Base Address
- #define **UTMI** ((Utmi ∗)0x400E0400U)
  (UTMI) Base Address
- #define **PMC** ((Pmc ∗)0x400E0600U)
  (PMC) Base Address
- #define **UART0** ((Uart ∗)0x400E0800U)
  (UART0) Base Address
- #define **CHIPID** ((Chipid ∗)0x400E0940U)
  (CHIPID) Base Address
- #define **UART1** ((Uart ∗)0x400E0A00U)
  (UART1) Base Address
- #define **EFC** ((Efc ∗)0x400E0C00U)
  (EFC) Base Address
- #define **PIO** ((Pio[5])0x400E0E00U)
  (PIO) Base Address
- #define **PIOA** ((Pio ∗)0x400E0E00U)
  (PIOA) Base Address
- #define **PIOB** ((Pio ∗)0x400E1000U)
  (PIOB) Base Address
- #define **PIOC** ((Pio ∗)0x400E1200U)
  (PIOC) Base Address
- #define **PIOD** ((Pio ∗)0x400E1400U)
  (PIOD) Base Address
- #define **PIOE** ((Pio ∗)0x400E1600U)
  (PIOE) Base Address
- #define **RSTC** ((Rstc ∗)0x400E1800U)
  (RSTC) Base Address
- #define **SUPC** ((Supc ∗)0x400E1810U)

(SUPC) Base Address

- #define **RTT** ((Rtt ∗)0x400E1830U)

  (RTT) Base Address
- #define **WDT** ((Wdt ∗)0x400E1850U)

  (WDT) Base Address
- #define **RTC** ((Rtc ∗)0x400E1860U)

  (RTC) Base Address
- #define **GPBR** ((Gpbr ∗)0x400E1890U)

  (GPBR) Base Address
- #define **RSWDT** ((Rswdt ∗)0x400E1900U)

  (RSWDT) Base Address
- #define **UART2** ((Uart ∗)0x400E1A00U)

  (UART2) Base Address
- #define **UART3** ((Uart ∗)0x400E1C00U)

  (UART3) Base Address
- #define **UART4** ((Uart ∗)0x400E1E00U)

  (UART4) Base Address
- #define **ADDR_HSMCI** (0x40000000U)

  (HSMCI) Base Address
- #define **ADDR_SSC** (0x40004000U)

  (SSC) Base Address
- #define **ADDR_SPI0** (0x40008000U)

  (SPI0) Base Address
- #define **ADDR_TC0** (0x4000C000U)

  (TC0) Base Address
- #define **ADDR_TC1** (0x40010000U)

  (TC1) Base Address
- #define **ADDR_TC2** (0x40014000U)

  (TC2) Base Address
- #define **ADDR_TWIHS0** (0x40018000U)

  (TWIHS0) Base Address
- #define **ADDR_TWIHS1** (0x4001C000U)

  (TWIHS1) Base Address
- #define **ADDR_PWM0** (0x40020000U)

  (PWM0) Base Address
- #define **ADDR_USART0** (0x40024000U)

  (USART0) Base Address
- #define **ADDR_USART1** (0x40028000U)

  (USART1) Base Address
- #define **ADDR_USART2** (0x4002C000U)

  (USART2) Base Address
- #define **ADDR_MCAN0** (0x40030000U)

  (MCAN0) Base Address
- #define **ADDR_MCAN1** (0x40034000U)

  (MCAN1) Base Address
- #define **ADDR_USBHS** (0x40038000U)

  (USBHS) Base Address
- #define **ADDR_AFEC0** (0x4003C000U)

  (AFEC0) Base Address
- #define **ADDR_DACC** (0x40040000U)

  (DACC) Base Address

- #define **ADDR_ACC** (0x40044000U)
  (ACC) Base Address
- #define **ADDR_ICM** (0x40048000U)
  (ICM) Base Address
- #define **ADDR_ISI** (0x4004C000U)
  (ISI) Base Address
- #define **ADDR_GMAC** (0x40050000U)
  (GMAC) Base Address
- #define **ADDR_TC3** (0x40054000U)
  (TC3) Base Address
- #define **ADDR_SPI1** (0x40058000U)
  (SPI1) Base Address
- #define **ADDR_PWM1** (0x4005C000U)
  (PWM1) Base Address
- #define **ADDR_TWIHS2** (0x40060000U)
  (TWIHS2) Base Address
- #define **ADDR_AFEC1** (0x40064000U)
  (AFEC1) Base Address
- #define **ADDR_AES** (0x4006C000U)
  (AES) Base Address
- #define **ADDR_TRNG** (0x40070000U)
  (TRNG) Base Address
- #define **ADDR_XDMAC** (0x40078000U)
  (XDMAC) Base Address
- #define **ADDR_QSPI** (0x4007C000U)
  (QSPI) Base Address
- #define **ADDR_SMC** (0x40080000U)
  (SMC) Base Address
- #define **ADDR_SDRAMC** (0x40084000U)
  (SDRAMC) Base Address
- #define **ADDR_MATRIX** (0x40088000U)
  (MATRIX) Base Address
- #define **ADDR_UTMI** (0x400E0400U)
  (UTMI) Base Address
- #define **ADDR_PMC** (0x400E0600U)
  (PMC) Base Address
- #define **ADDR_UART0** (0x400E0800U)
  (UART0) Base Address
- #define **ADDR_CHIPID** (0x400E0940U)
  (CHIPID) Base Address
- #define **ADDR_UART1** (0x400E0A00U)
  (UART1) Base Address
- #define **ADDR_EFC** (0x400E0C00U)
  (EFC) Base Address
- #define **ADDR_PIOA** (0x400E0E00U)
  (PIOA) Base Address
- #define **ADDR_PIOB** (0x400E1000U)
  (PIOB) Base Address
- #define **ADDR_PIOC** (0x400E1200U)
  (PIOC) Base Address
- #define **ADDR_PIOD** (0x400E1400U)

(PIOD) Base Address

- #define **ADDR_PIOE** (0x400E1600U)

  (PIOE) Base Address
- #define **ADDR_RSTC** (0x400E1800U)

  (RSTC) Base Address
- #define **ADDR_SUPC** (0x400E1810U)

  (SUPC) Base Address
- #define **ADDR_RTT** (0x400E1830U)

  (RTT) Base Address
- #define **ADDR_WDT** (0x400E1850U)

  (WDT) Base Address
- #define **ADDR_RTC** (0x400E1860U)

  (RTC) Base Address
- #define **ADDR_GPBR** (0x400E1890U)

  (GPBR) Base Address
- #define **ADDR_RSWDT** (0x400E1900U)

  (RSWDT) Base Address
- #define **ADDR_UART2** (0x400E1A00U)

  (UART2) Base Address
- #define **ADDR_UART3** (0x400E1C00U)

  (UART3) Base Address
- #define **ADDR_UART4** (0x400E1E00U)

  (UART4) Base Address

24.56.1 Detailed Description

24.57 same70q21_sim.h

Go to the documentation of this file.

```
00001 #ifndef  __SAME70Q21_SIM_H
00002 #define  __SAME70Q21_SIM_H
00037 /*
00038  * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel Support</a>
00039  */
00040
00041
00042
00043 /*
00044  *  SOFTWARE PERIPHERAL API DEFINITION FOR SAME70Q21
00045  */
00046
00047 /* \addtogroup SAME70Q21_api Peripheral Software API
00048  * @{
00049  */
00050 #include "component/acc.h"
00051 #include "component/aes.h"
00052 #include "component/afec.h"
00053 #include "component/chipid.h"
00054 #include "component/dacc.h"
00055 #include "component/efc.h"
00056 #include "component/gmac.h"
00057 #include "component/gpbr.h"
00058 #include "component/hsmci.h"
00059 #include "component/icm.h"
00060 #include "component/isi.h"
00061 #include "component/matrix.h"
00062 #include "component/mcan.h"
00063 #include "component/pio.h"
00064 #include "component/pmc.h"
00065 #include "component/pwm.h"
00066 #include "component/qspi.h"
00067 #include "component/rstc.h"
00068 #include "component/rswdt.h"
00069 #include "component/rtc.h"
00070 #include "component/rtt.h"
00071 #include "component/sdramc.h"
00072 #include "component/smc.h"
00073 #include "component/spi.h"
00074 #include "component/ssc.h"
00075 #include "component/supc.h"
00076 #include "component/tc.h"
00077 #include "component/trng.h"
00078 #include "component/twihs.h"
00079 #include "component/uart.h"
00080 #include "component/usart.h"
00081 #include "component/usbhs.h"
00082 #include "component/utmi.h"
00083 #include "component/wdt.h"
00084 #include "component/xdmac.h"
00085 /* @} */
00086
00087 /*
00088  *   REGISTER ACCESS DEFINITIONS FOR SAME70Q21
00089  */
00090
00091 /* \addtogroup SAME70Q21_reg Registers Access Definitions
00092  * @{
00093  */
00094 #include "instance/hsmci.h"
00095 #include "instance/ssc.h"
00096 #include "instance/spi0.h"
00097 #include "instance/tc0.h"
00098 #include "instance/tc1.h"
00099 #include "instance/tc2.h"
00100 #include "instance/twihs0.h"
00101 #include "instance/twihs1.h"
00102 #include "instance/pwm0.h"
00103 #include "instance/usart0.h"
00104 #include "instance/usart1.h"
00105 #include "instance/usart2.h"
00106 #include "instance/mcan0.h"
00107 #include "instance/mcan1.h"
00108 #include "instance/usbhs.h"
00109 #include "instance/afec0.h"
00110 #include "instance/dacc.h"
00111 #include "instance/acc.h"
00112 #include "instance/icm.h"
00113 #include "instance/isi.h"
00114 #include "instance/gmac.h"
00115 #include "instance/tc3.h"
00116 #include "instance/spi1.h"
00117 #include "instance/pwm1.h"
```

```
00118 #include "instance/twihs2.h"
00119 #include "instance/afec1.h"
00120 #include "instance/aes.h"
00121 #include "instance/trng.h"
00122 #include "instance/xdmac.h"
00123 #include "instance/qspi.h"
00124 #include "instance/smc.h"
00125 #include "instance/sdramc.h"
00126 #include "instance/matrix.h"
00127 #include "instance/utmi.h"
00128 #include "instance/pmc.h"
00129 #include "instance/uart0.h"
00130 #include "instance/chipid.h"
00131 #include "instance/uart1.h"
00132 #include "instance/efc.h"
00133 #include "instance/pioa.h"
00134 #include "instance/piob.h"
00135 #include "instance/pioc.h"
00136 #include "instance/piod.h"
00137 #include "instance/pioe.h"
00138 #include "instance/rstc.h"
00139 #include "instance/supc.h"
00140 #include "instance/rtt.h"
00141 #include "instance/wdt.h"
00142 #include "instance/rtc.h"
00143 #include "instance/gpbr.h"
00144 #include "instance/rswdt.h"
00145 #include "instance/uart2.h"
00146 #include "instance/uart3.h"
00147 #include "instance/uart4.h"
00148 /* @} */
00149
00150 /*
00151  *   BASE ADDRESS DEFINITIONS FOR SAME70Q21
00152  */
00153
00154 /* \addtogroup SAME70Q21_base Peripheral Base Address Definitions
00155  * @{
00156  */
00157 #if (defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_ASM__))
00158 #define HSMCI   (0x40000000U)
00159 #define SSC     (0x40004000U)
00160 #define SPI0    (0x40008000U)
00161 #define TC0     (0x4000C000U)
00162 #define TC1     (0x40010000U)
00163 #define TC2     (0x40014000U)
00164 #define TWIHS0  (0x40018000U)
00165 #define TWIHS1  (0x4001C000U)
00166 #define PWM0    (0x40020000U)
00167 #define USART0  (0x40024000U)
00168 #define USART1  (0x40028000U)
00169 #define USART2  (0x4002C000U)
00170 #define MCAN0   (0x40030000U)
00171 #define MCAN1   (0x40034000U)
00172 #define USBHS   (0x40038000U)
00173 #define AFEC0   (0x4003C000U)
00174 #define DACC    (0x40040000U)
00175 #define ACC     (0x40044000U)
00176 #define ICM     (0x40048000U)
00177 #define ISI     (0x4004C000U)
00178 #define GMAC    (0x40050000U)
00179 #define TC3     (0x40054000U)
00180 #define SPI1    (0x40058000U)
00181 #define PWM1    (0x4005C000U)
00182 #define TWIHS2  (0x40060000U)
00183 #define AFEC1   (0x40064000U)
00184 #define AES     (0x4006C000U)
00185 #define TRNG    (0x40070000U)
00186 #define XDMAC   (0x40078000U)
00187 #define QSPI    (0x4007C000U)
00188 #define SMC     (0x40080000U)
00189 #define SDRAMC  (0x40084000U)
00190 #define MATRIX  (0x40088000U)
00191 #define UTMI    (0x400E0400U)
00192 #define PMC     (0x400E0600U)
00193 #define UART0   (0x400E0800U)
00194 #define CHIPID  (0x400E0940U)
00195 #define UART1   (0x400E0A00U)
00196 #define EFC     (0x400E0C00U)
00197 #define PIOA    (0x400E0E00U)
00198 #define PIOB    (0x400E1000U)
00199 #define PIOC    (0x400E1200U)
00200 #define PIOD    (0x400E1400U)
00201 #define PIOE    (0x400E1600U)
00202 #define RSTC    (0x400E1800U)
00203 #define SUPC    (0x400E1810U)
00204 #define RTT     (0x400E1830U)
```

```
00205 #define WDT    (0x400E1850U)
00206 #define RTC    (0x400E1860U)
00207 #define GPBR   (0x400E1890U)
00208 #define RSWDT  (0x400E1900U)
00209 #define UART2  (0x400E1A00U)
00210 #define UART3  (0x400E1C00U)
00211 #define UART4  (0x400E1E00U)
00212 #else
00213 #define HSMCI  ((Hsmci  *)0x40000000U)
00214 #define SSC    ((Ssc    *)0x40004000U)
00215 #define SPI0   ((Spi    *)0x40008000U)
00216 #define TC0    ((Tc     *)0x4000C000U)
00217 #define TC1    ((Tc     *)0x40010000U)
00218 #define TC2    ((Tc     *)0x40014000U)
00219 #define TWIHS0 ((Twihs  *)0x40018000U)
00220 #define TWIHS1 ((Twihs  *)0x4001C000U)
00221 #define PWM0   ((Pwm    *)0x40020000U)
00222 #define USART0 ((Usart  *)0x40024000U)
00223 #define USART1 ((Usart  *)0x40028000U)
00224 #define USART2 ((Usart  *)0x4002C000U)
00225 #define MCAN0  ((Mcan   *)0x40030000U)
00226 #define MCAN1  ((Mcan   *)0x40034000U)
00227 #define USBHS  ((Usbhs  *)0x40038000U)
00228 #define AFEC0  ((Afec   *)0x4003C000U)
00229 #define DACC   ((Dacc   *)0x40040000U)
00230 #define ACC    ((Acc    *)0x40044000U)
00231 #define ICM    ((Icm    *)0x40048000U)
00232 #define ISI    ((Isi    *)0x4004C000U)
00233 #define GMAC   ((Gmac   *)0x40050000U)
00234 #define TC3    ((Tc     *)0x40054000U)
00235 #define SPI1   ((Spi    *)0x40058000U)
00236 #define PWM1   ((Pwm    *)0x4005C000U)
00237 #define TWIHS2 ((Twihs  *)0x40060000U)
00238 #define AFEC1  ((Afec   *)0x40064000U)
00239 #define AES    ((Aes    *)0x4006C000U)
00240 #define TRNG   ((Trng   *)0x40070000U)
00241 #define XDMAC  ((Xdmac  *)0x40078000U)
00242 #define QSPI   ((Qspi   *)0x4007C000U)
00243 #define SMC    ((Smc    *)0x40080000U)
00244 #define SDRAMC ((Sdramc *)0x40084000U)
00245 #define MATRIX ((Matrix *)0x40088000U)
00246 #define UTMI   ((Utmi   *)0x400E0400U)
00247 #define PMC    ((Pmc    *)0x400E0600U)
00248 #define UART0  ((Uart   *)0x400E0800U)
00249 #define CHIPID ((Chipid *)0x400E0940U)
00250 #define UART1  ((Uart   *)0x400E0A00U)
00251 #define EFC    ((Efc    *)0x400E0C00U)
00252 #define PIO    ((Pio[5]  )0x400E0E00U)
00253 #define PIOA   ((Pio    *)0x400E0E00U)
00254 #define PIOB   ((Pio    *)0x400E1000U)
00255 #define PIOC   ((Pio    *)0x400E1200U)
00256 #define PIOD   ((Pio    *)0x400E1400U)
00257 #define PIOE   ((Pio    *)0x400E1600U)
00258 #define RSTC   ((Rstc   *)0x400E1800U)
00259 #define SUPC   ((Supc   *)0x400E1810U)
00260 #define RTT    ((Rtt    *)0x400E1830U)
00261 #define WDT    ((Wdt    *)0x400E1850U)
00262 #define RTC    ((Rtc    *)0x400E1860U)
00263 #define GPBR   ((Gpbr   *)0x400E1890U)
00264 #define RSWDT  ((Rswdt  *)0x400E1900U)
00265 #define UART2  ((Uart   *)0x400E1A00U)
00266 #define UART3  ((Uart   *)0x400E1C00U)
00267 #define UART4  ((Uart   *)0x400E1E00U)
00269 #define ADDR_HSMCI  (0x40000000U)
00270 #define ADDR_SSC    (0x40004000U)
00271 #define ADDR_SPI0   (0x40008000U)
00272 #define ADDR_TC0    (0x4000C000U)
00273 #define ADDR_TC1    (0x40010000U)
00274 #define ADDR_TC2    (0x40014000U)
00275 #define ADDR_TWIHS0 (0x40018000U)
00276 #define ADDR_TWIHS1 (0x4001C000U)
00277 #define ADDR_PWM0   (0x40020000U)
00278 #define ADDR_USART0 (0x40024000U)
00279 #define ADDR_USART1 (0x40028000U)
00280 #define ADDR_USART2 (0x4002C000U)
00281 #define ADDR_MCAN0  (0x40030000U)
00282 #define ADDR_MCAN1  (0x40034000U)
00283 #define ADDR_USBHS  (0x40038000U)
00284 #define ADDR_AFEC0  (0x4003C000U)
00285 #define ADDR_DACC   (0x40040000U)
00286 #define ADDR_ACC    (0x40044000U)
00287 #define ADDR_ICM    (0x40048000U)
00288 #define ADDR_ISI    (0x4004C000U)
00289 #define ADDR_GMAC   (0x40050000U)
00290 #define ADDR_TC3    (0x40054000U)
00291 #define ADDR_SPI1   (0x40058000U)
00292 #define ADDR_PWM1   (0x4005C000U)
```

```
00293 #define ADDR_TWIHS2 (0x40060000U)
00294 #define ADDR_AFEC1  (0x40064000U)
00295 #define ADDR_AES    (0x4006C000U)
00296 #define ADDR_TRNG   (0x40070000U)
00297 #define ADDR_XDMAC  (0x40078000U)
00298 #define ADDR_QSPI   (0x4007C000U)
00299 #define ADDR_SMC    (0x40080000U)
00300 #define ADDR_SDRAMC (0x40084000U)
00301 #define ADDR_MATRIX (0x40088000U)
00302 #define ADDR_UTMI   (0x400E0400U)
00303 #define ADDR_PMC    (0x400E0600U)
00304 #define ADDR_UART0  (0x400E0800U)
00305 #define ADDR_CHIPID (0x400E0940U)
00306 #define ADDR_UART1  (0x400E0A00U)
00307 #define ADDR_EFC    (0x400E0C00U)
00308 #define ADDR_PIOA   (0x400E0E00U)
00309 #define ADDR_PIOB   (0x400E1000U)
00310 #define ADDR_PIOC   (0x400E1200U)
00311 #define ADDR_PIOD   (0x400E1400U)
00312 #define ADDR_PIOE   (0x400E1600U)
00313 #define ADDR_RSTC   (0x400E1800U)
00314 #define ADDR_SUPC   (0x400E1810U)
00315 #define ADDR_RTT    (0x400E1830U)
00316 #define ADDR_WDT    (0x400E1850U)
00317 #define ADDR_RTC    (0x400E1860U)
00318 #define ADDR_GPBR   (0x400E1890U)
00319 #define ADDR_RSWDT  (0x400E1900U)
00320 #define ADDR_UART2  (0x400E1A00U)
00321 #define ADDR_UART3  (0x400E1C00U)
00322 #define ADDR_UART4  (0x400E1E00U)
00324 #endif /* (defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_ASM__)) */
00325 /* @} */
00326
00327 /*
00328  *   PIO DEFINITIONS FOR SAME70Q21
00329  */
00330
00331 /* \addtogroup SAME70Q21_pio Peripheral Pio Definitions
00332  * @{
00333  */
00334
00335 #include "pio/same70q21.h"
00336 /* @} */
00337
00338
00339
00340 #endif   /* ----- #ifndef __SAME70Q21_SIM_H  ----- */
```

24.58 system_same70.h File Reference

`#include <stdint.h>`

Functions

- void **SystemInit** (void)

  Setup the microcontroller system. Initialize the System and update the SystemCoreClock variable.
- void **SystemCoreClockUpdate** (void)

  Updates the SystemCoreClock with current core Clock retrieved from cpu registers.
- void system_init_flash (uint32_t dw_clk)

Variables

- uint32_t SystemCoreClock

24.58.1 Detailed Description

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

24.58.2 Function Documentation

24.58.2.1 system_init_flash()

```
void system_init_flash (
            uint32_t dw_clk )
```

Initialize flash.

24.58.3 Variable Documentation

24.58.3.1 SystemCoreClock

```
uint32_t SystemCoreClock  [extern]
```

INDENT-OFF INDENT-ON

24.59 system_same70.h

Go to the documentation of this file.

```
00001
00037 /*
00038  * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel Support</a>
00039  */
00040
00041 #ifndef SYSTEM_SAME70_H_INCLUDED
00042 #define SYSTEM_SAME70_H_INCLUDED
00043
00044 /* cond 0 */
00046 #ifdef __cplusplus
00047 extern "C" {
00048 #endif
00050 /* endcond */
00051
00052 #include <stdint.h>
00053
00054 extern uint32_t SystemCoreClock; /* System Clock Frequency (Core Clock) */
00055
00060 void SystemInit(void);
00061
00066 void SystemCoreClockUpdate(void);
00067
00071 void system_init_flash(uint32_t dw_clk);
00072
00073 /* cond 0 */
00075 #ifdef __cplusplus
00076 }
00077 #endif
```

```
00079 /* endcond */
00080
00081 #endif /* SYSTEM_SAME70_H_INCLUDED */
```

24.60 timer_dispatch.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef _TIMER_DISPATCH_H
00005 #define _TIMER_DISPATCH_H
00006
00007 #include <predef.h>
00008
00009 #include <constants.h>
00010 #include <sim.h>
00011
00012
00013 /* This adds a timer dispatch capability to the system so timers can be used for multiple independant
      functions*/
00014
00015
00016 //ISR callback will have timer number and a channel object
00017 typedef void (*TimerIsrHandler)(int timer_num, TcChannel &tc,void * pExtra);
00018
00019 /* Returns timer alocated, or -1 if timer selected is in use, or -2 if none are availible*/
00020 int AllocateTimer( TimerIsrHandler tisr,void * pExtra,int timer=FIRST_UNUSED_TIMER);
00021 void FreeTimer(int timer);
00022
00023
00024 /* helper functions */
00025 IRQn GetTimerPeriphID(int timer);
00026
00027 TcChannel * GetTc(int timer);
00028
00029
00030
00031
00032 #endif
```

24.61 usart.h File Reference

NetBurner DMA USART in SPI mode API for ARM SAME70.

```
#include <nbrtos.h>
#include <basictypes.h>
#include <xdmac.h>
#include <dspi.h>
```

Classes

- class SPI_USART

  USART in SPI mode Peripheral Module Class.

Macros

- #define **DEFAULT_USART_SPI_MODULE** 0

  Default QUADSPI module.
- #define **USART_SPI_MODULE_COUNT** 2

  Number of modules: 0, 1.

24.61.1 Detailed Description

NetBurner DMA USART in SPI mode API for ARM SAME70.

24.62 usart.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00024 #ifndef  _DMA_USART_SPI_H_INC
00025 #define  _DMA_USART_SPI_H_INC
00026
00027 #include <nbrtos.h>
00028 #include <basictypes.h>
00029 #include <xdmac.h>
00030 #include <dspi.h>
00031
00032 #ifndef __cplusplus
00033 #error QuadSPI driver is a C++ only library
00034 #endif
00035
00036 // #define __DEBUG_USART_SPI     1
00037
00038
00044 #define DEFAULT_USART_SPI_MODULE 0
00045 #define USART_SPI_MODULE_COUNT   2
00054 class SPI_USART : public SPI_SPI
00055 {
00056 protected:
00057     static  SPI_USART *lastUSARTCtxs[USART_SPI_MODULE_COUNT];
00058     static  spiDriverStruct UsartSpiDriverCtx[USART_SPI_MODULE_COUNT];
00059
00060 private:
00061     uint32_t    m_regCR;
00062     uint32_t    m_regBRGR;          // Baud Rate Generator Register
00063
00064     virtual void ReadyHW() override;
00065     virtual spiDriverStruct *getCtx() override { return UsartSpiDriverCtx + m_moduleNum; }
00066     inline Usart * usartSpi() { return USART0 + m_moduleNum; }
00067
00068 public:
00069
00070 #ifdef __DEBUG_QUADSPI
00071     virtual void    dumpRegs() override;
00072 #endif
00073
00074
00083     SPI_USART( uint8_t USARTModule );
00084
00102     SPI_USART( uint8_t USARTModule, uint32_t baudRateInBps,
00103             uint8_t transferSizeInBits = 8, uint8_t peripheralChipSelects = 0x00,
00104             uint8_t chipSelectPolarity = 0x0F, uint8_t clockPolarity = 0,
00105             uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00106             uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 );
00107
00123     virtual uint8_t Init( uint32_t baudRateInBps = 2000000,
00124             uint8_t transferSizeInBits = 8, uint8_t peripheralChipSelects = 0x00,
00125             uint8_t chipSelectPolarity = 0x0F, uint8_t clockPolarity = 0,
00126             uint8_t clockPhase = 1, BOOL doutHiz = TRUE,
00127             uint8_t csToClockDelay = 0, uint8_t delayAfterTransfer = 0 ) override;
00128
00138     virtual uint32_t SetBusSpeed(uint32_t maxSpeed) override;
00139
00156     virtual uint8_t Start( uint8_t *transmitBufferPtr, volatile uint8_t *receiveBufferPtr,
00157                 uint32_t byteCount, int csReturnToInactive = DEASSERT_AFTER_LAST ) override;
00158
00168     virtual inline uint8_t Tx( uint8_t *transmitBufferPtr, uint32_t byteCount,
00169                     int csReturnToInactive = DEASSERT_AFTER_LAST ) override
00170     { return Start(transmitBufferPtr, NULL, byteCount, csReturnToInactive); }
00171
00181     inline uint8_t Rx( uint8_t *receiveBufferPtr, uint32_t byteCount,
00182                     int csReturnToInactive = DEASSERT_AFTER_LAST )
00183     { return Start(NULL, receiveBufferPtr, byteCount, csReturnToInactive); }
00184
00185     // The SAME70 always uses DMA. These functions kept here for reference to other platforms
00186     //    bool EnableDMA(bool enableDMA = true);
00187     //    inline bool DisableDMA() { return EnableDMA(false); }
00188
00189     /*
00190      * @brief Register a semaphore for the SPI module.
00191      *
00192      * The SPI module will post to this semaphore when a transaction is complete
00193      *
00194      * @param finishedSem  Pointer to the semaphore
00195      *
00196      * @return true if the registration was successful, false if a SPI transaction is in progress
00197      */
00198     // bool RegisterSem( OS_SEM *finishedSem );
```

```
00199
00200     /*
00201      * @brief Clear a semaphore registration
00202      *
00203      * @return true if the clear was successful, false if a SPI transaction is in progress
00204      */
00205     // inline bool ClrSem() { return RegisterSem(NULL); }
00206
00207     /*
00208      * @brief Obtain a pointer to the SPI finished semaphore
00209      *
00210      * @return A pointer to the semaphore
00211      */
00212     // inline OS_SEM * GetSem() { return m_finishedSem; }
00213
00214     /*
00215      * @brief Function to check SPI status
00216      *
00217      * Called as a class method on a specific SPI object. For example: MySpi.Done()
00218      *
00219      * @return true if SPI is finished, false if active
00220      */
00221     // inline bool Done() { return !m_inProgress; }
00222
00223     /*
00224      * @brief Returns the active baud rate
00225      *
00226      * The baud rate will be set to the value specified when the SPI module is initialized.
00227      * If that value is not possible, the next lowest baud rate will be chosen.
00228      *
00229      * @return The actual SPI module baud rate
00230      */
00231     // inline uint32_t GetActualBaudrate() { return m_busSpeed; }
00232
00233     /*
00234      * @brief Set the chip select configuration for the SPI object's bus transactions
00235      *
00236      * @param CS    USART modules only have one option for chip select.
00237      *
00238      * @return true if successful, false if SPI is currently active
00239      */
00240     virtual inline bool SetCS( uint8_t CS ) override
00241     {
00242         OSLockObj lock;
00243         if (m_inProgress) { return false; }
00244
00245         return true;
00246     }
00247
00248     // The ISR used by the USART SPI driver. Internal use only (usart.cpp).
00249     friend void USART_DMA_Isr(XdmaCh_t *dma, int module);
00250 };
00251
00252 #endif   /* ----- #ifndef _DMA_USART_SPI_H_INC  ----- */
00253
```

24.63 xdmac.h

```
00001 #ifndef  __XDMAC_H
00002 #define  __XDMAC_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <same70.h>
00007 #include <basictypes.h>
00008 #include <nbrtos.h>
00009
00010 struct dma_descview_0;
00011 struct dma_descview_1;
00012 struct dma_descview_2;
00013 struct dma_descview_3;
00014
00015 struct dma_transfer_desc_t
00016 {
00017     union
00018     {
00019         dma_transfer_desc_t *pNext;
00020         dma_descview_0      *pNext_0;
00021         dma_descview_1      *pNext_1;
00022         dma_descview_2      *pNext_2;
00023         dma_descview_3      *pNext_3;
00024         uint32_t            nda;
00025     };
00026     uint32_t ubCtrl;
00027 };
```

```
00028 struct dma_descview_0 : public dma_transfer_desc_t
00029 {
00030     uint32_t addr;
00031 };
00032
00033 struct dma_descview_1 : public dma_transfer_desc_t
00034 {
00035     uint32_t src;
00036     uint32_t dst;
00037 };
00038
00039 struct dma_descview_2 : public dma_transfer_desc_t
00040 {
00041     uint32_t src;
00042     uint32_t dst;
00043     uint32_t cfg;
00044 };
00045
00046 struct dma_descview_3 : public dma_transfer_desc_t
00047 {
00048     uint32_t src;
00049     uint32_t dst;
00050     uint32_t cfg;
00051     uint32_t blkCtrl;
00052     uint32_t stride;
00053     uint32_t srcuStride;
00054     uint32_t dstuStride;
00055 };
00056
00057 #include <type_traits>
00058 struct XdmaCh_t;
00059 extern "C" XdmaCh_t * xdmacGetFreeCh();
00060 extern "C" void xdmacReleaseCh(XdmaCh_t *xdmaCh);
00061
00062
00063 struct XdmaChCtx_t
00064 {
00065     uint32_t lastISR;
00066     uint32_t stickyISR;
00067     void (* isr) (XdmaCh_t *);
00068     void reset();
00069 };
00070
00071 struct XdmaCh_t : public XdmacChid
00072 {
00073     private:
00074     static XdmaChCtx_t      ctx[XDMACCHID_NUMBER];
00075     static uint32_t         bInUse;
00076     public:
00077     enum TransType_t
00078     {
00079         Mem2Mem         = XDMAC_CC_TYPE_MEM_TRAN_Val,
00080         Periph          = XDMAC_CC_TYPE_PER_TRAN_Val
00081     };
00082     enum MBSiz_t
00083     {
00084         MBSiz_1         = XDMAC_CC_MBSIZE_SINGLE_Val,
00085         MBSiz_4         = XDMAC_CC_MBSIZE_FOUR_Val,
00086         MBSiz_8         = XDMAC_CC_MBSIZE_EIGHT_Val,
00087         MBSiz_16        = XDMAC_CC_MBSIZE_SIXTEEN_Val
00088     };
00089     enum DSync_t
00090     {
00091         Periph2Mem      = XDMAC_CC_DSYNC_PER2MEM_Val,
00092         Mem2Periph      = XDMAC_CC_DSYNC_MEM2PER_Val
00093     };
00094     enum SWReq_t
00095     {
00096         Req_HW          = XDMAC_CC_SWREQ_HWR_CONNECTED_Val,
00097         Req_SW          = XDMAC_CC_SWREQ_SWR_CONNECTED_Val
00098     };
00099     enum ChunkSiz_t
00100     {
00101         ChunkSiz_1      = XDMAC_CC_CSIZE_CHK_1_Val,
00102         ChunkSiz_2      = XDMAC_CC_CSIZE_CHK_2_Val,
00103         ChunkSiz_4      = XDMAC_CC_CSIZE_CHK_4_Val,
00104         ChunkSiz_8      = XDMAC_CC_CSIZE_CHK_8_Val,
00105         ChunkSiz_16     = XDMAC_CC_CSIZE_CHK_16_Val
00106     };
00107     enum DWidth_t
00108     {
00109         DWidth_Byte     = XDMAC_CC_DWIDTH_BYTE_Val,
00110         DWidth_HalfWord = XDMAC_CC_DWIDTH_HALFWORD_Val,
00111         DWidth_Word     = XDMAC_CC_DWIDTH_WORD_Val
00112     };
00113     enum AHB_IF_t
00114     {
```

```
00115         AHB_IF_0        = XDMAC_CC_SIF_AHB_IF0_Val,
00116         AHB_IF_1        = XDMAC_CC_SIF_AHB_IF1_Val
00117     };
00118     enum AddrMode_t
00119     {
00120         AddrMode_Fixed  = XDMAC_CC_SAM_FIXED_AM_Val,
00121         AddrMode_Inc    = XDMAC_CC_SAM_INCREMENTED_AM_Val,
00122         AddrMode_uBS    = XDMAC_CC_SAM_UBS_AM_Val,
00123         AddrMode_uBS_DS = XDMAC_CC_SAM_UBS_DS_AM_Val
00124     };
00125     // Fetches the status from HW and stores to ctx
00126     inline uint32_t readyStatus()
00127     {
00128         ctx[getID()].stickyISR |= ctx[getID()].lastISR = XDMAC_CIS;
00129         return ctx[getID()].lastISR;
00130     }
00131     // Returns the status as stored in ctx
00132     inline uint32_t getStatus()     { return ctx[getID()].lastISR;}
00133     inline uint32_t getStickyStatus()     { return ctx[getID()].stickyISR;}
00134     inline void     clrStickyStatus(uint32_t mask = 0xFFFFFFFFul)
00135     {
00136         ctx[getID()].stickyISR &= ~mask;
00137     }
00138     inline uint32_t getID()         { return((XdmacChid *)this) - XDMAC->XDMAC_CHID;}
00139
00140     inline void enableGIrq() { XDMAC->XDMAC_GIE = 1 « getID();}
00141     inline void disableGIrq() { XDMAC->XDMAC_GID = 1 « getID();}
00142
00143     inline void suspRd()    { XDMAC->XDMAC_GRS |= 1 « getID();}
00144     inline void suspWr()    { XDMAC->XDMAC_GWS |= 1 « getID();}
00145     inline void suspRdWr()  { XDMAC->XDMAC_GRWS = 1 « getID();}
00146     inline void resRd()     { XDMAC->XDMAC_GRS &= ~(1 « getID());}
00147     inline void resWr()     { XDMAC->XDMAC_GWS &= ~(1 « getID());}
00148     inline void resRdWr()   { XDMAC->XDMAC_GRWR = 1 « getID();}
00149     inline void flush()     { XDMAC->XDMAC_GSWF = 1 « getID();}
00150     inline void disable()   { XDMAC->XDMAC_GD = 1 « getID();}
00151     inline void enable()    { XDMAC->XDMAC_GE = 1 « getID();}
00152     inline bool isEnabled() { return(XDMAC->XDMAC_GS) & (1 « getID());}
00153     inline void suspRdAndFlush()
00154     {
00155         /* For some reason the channel cannot flush the transfer FIFO if
00156          *  it is requesting to fetch the next descriptor and the read
00157          *  interface is suspended. So, if the channel is suspended *as*
00158          *  the previous configuration is depleted *and* it is configured to
00159          *  fetch a new descriptor, we need to renable the read interface,
00160          *  let the new descriptor be fetched, and *then* we can suspend the
00161          *  channel for the flush.
00162          *
00163          *  No, this isn't documented in the manual. */
00164         suspRd();
00165
00166         readyStatus();
00167         flush();
00168         int i = 0;
00169         do
00170         {
00171             if (i < 25)
00172             {
00173                 i++;
00174                 asm("dsb");
00175                 continue;
00176             }
00177             if ((XDMAC_CNDC & XDMAC_CNDC_NDE)
00178                 && (guBLen() == 0))
00179             {
00180                 resRd();
00181                 asm("dsb");
00182                 asm("dsb");
00183
00184                 while (!(readyStatus() & XDMAC_CIS_FIS))
00185                 {
00186                 }
00187                 suspRd();
00188                 flush();
00189             }
00190         } while (!(readyStatus() & XDMAC_CIS_FIS));
00191     }
00192
00193     inline void sSrcAddr(uint32_t src)  { XDMAC_CSA = src;}
00194     inline void sDstAddr(uint32_t dst)  { XDMAC_CDA = dst;}
00195     inline uint32_t gSrcAddr()          { return XDMAC_CSA;}
00196     inline uint32_t gDstAddr()          { return XDMAC_CDA;}
00197     inline void suBLen(uint32_t uBLen)  { XDMAC_CUBC = uBLen;}
00198     inline void s_BLen(uint32_t BLen)   { XDMAC_CBC = BLen;}
00199     inline uint32_t guBLen()            { return XDMAC_CUBC;}
00200     dma_transfer_desc_t * getNextDesc(uint32_t *remDataRet);
00201     inline void *gNextDest()
```

```
00202     {
00203         dma_transfer_desc_t *desc = getNextDesc(NULL);
00204         if ((XDMAC_CNDA & XDMAC_CNDA_NDA_Msk) == 0)
00205         {
00206             return NULL;
00207         }
00208         return(XDMAC_CNDC » 3)
00209         ? (void *)(((dma_descview_1 *)XDMAC_CNDA)->dst)
00210         : (void *)(((dma_descview_0 *)XDMAC_CNDA)->addr);
00211     }
00212     inline uint32_t gNextuBLen()
00213     {
00214         return((XDMAC_CNDA & XDMAC_CNDA_NDA_Msk) == 0)
00215         ? 0
00216         : (((dma_descview_0 *)XDMAC_CNDA)->ubCtrl) & 0xFFFFFF;
00217     }
00218     void sNextDescAddr(dma_transfer_desc_t *pDesc);
00219     inline uint32_t gXfrWidth()
00220     { return(XDMAC_CC & XDMAC_CC_DWIDTH_Msk) » (XDMAC_CC_DWIDTH_Pos - 1);}
00221
00222     inline void sConfig(TransType_t type, MBSiz_t mbsiz, DSync_t dir,
00223                         SWReq_t reqSrc, ChunkSiz_t chunksiz, DWidth_t dataWidth,
00224                         AHB_IF_t srcIF, AHB_IF_t dstIF, AddrMode_t srcAddrMode,
00225                         AddrMode_t dstAddrMode, uint8_t periphID)
00226     {
00227
00228         // The following is a fix for a SAME70 XDMAC errata
00229         /***** BEGIN ERRATA FIX *****/
00230         if (dataWidth != DWidth_Word)
00231         {
00232             if (srcAddrMode == AddrMode_Fixed)
00233             {
00234                 srcAddrMode = AddrMode_uBS_DS;
00235                 // set source data stride stride to -1
00236                 XDMAC_CDS_MSP |= (0xFFFF);
00237                 // ERRATA IS WRONG!
00238                 //  We need to set the microstride to -1 as well.
00239                 XDMAC_CSUS = 0xFFFFFF;
00240             }
00241             if (dstAddrMode == AddrMode_Fixed)
00242             {
00243                 dstAddrMode = AddrMode_uBS_DS;
00244                 // set dest data stride stride to -1
00245                 XDMAC_CDS_MSP |= (0xFFFF « 16);
00246                 // ERRATA IS WRONG!
00247                 //  We need to set the microstride to -1 as well.
00248                 XDMAC_CDUS = 0xFFFFFF;
00249             }
00250         }
00251         /***** END ERRATA FIX *****/
00252         XDMAC_CC = (type « XDMAC_CC_TYPE_Pos) | (mbsiz « XDMAC_CC_MBSIZE_Pos)
00253                    | (dir « XDMAC_CC_DSYNC_Pos) | (chunksiz « XDMAC_CC_CSIZE_Pos)
00254                    | (dataWidth « XDMAC_CC_DWIDTH_Pos)
00255                    | (srcIF « XDMAC_CC_SIF_Pos) | (dstIF « XDMAC_CC_DIF_Pos)
00256                    | (srcAddrMode « XDMAC_CC_SAM_Pos)
00257                    | (dstAddrMode « XDMAC_CC_DAM_Pos)
00258                    | (periphID « XDMAC_CC_PERID_Pos);
00259     }
00260
00261
00262
00263     inline void RegisterIsr(void (*isr)(XdmaCh_t *)) { ctx[getID()].isr = isr;}
00264     // First Descriptor is only used to declare initial loading conditions
00265
00266     friend void XDMAC_Handler();
00267     friend XdmaCh_t * xdmacGetFreeCh();
00268     friend void xdmacReleaseCh(XdmaCh_t *xdmaCh);
00269 };
00270
00271 XdmaCh_t::AHB_IF_t xdmacGetIntfForAddr(uint32_t addr);
00272
00273 inline void XdmaCh_t::sNextDescAddr(dma_transfer_desc_t *pDesc)
00274 {
00275     XDMAC_CNDA = (((uint32_t)pDesc) & ~0x3)
00276                  | xdmacGetIntfForAddr((uint32_t)pDesc);
00277 }
00278
00279 #define XDMA_BD_INUSE           (0x1)
00280 #define XDMA_BD_SWEEP_MARK      (0x2)
00281 #define XDMA_BD_BUFFER_DONE_Msk (XDMA_BD_SWEEP_MARK | XDMA_BD_INUSE)
00282 #define XDMA_BD_BUFFER_DONE     (XDMA_BD_INUSE)
00283
00284 struct Desc0Ring
00285 {
00286     XdmaCh_t       *xdmaCh;
00287     dma_descview_0 *descs;
00288     dma_descview_0 *queueHead;
```

```
00289     dma_descview_0 *modRelease;
00290     uint32_t       *BDFlag;
00291     uint32_t        ringLen;
00292     uint32_t        spinLength;
00293     uint32_t        spinBuf;
00294     int             xfrIncSiz;
00295     uint8_t         ubCtrlBits;
00296     bool            spinOnLastNotNext;
00297
00298     uint32_t        nConfigFirst;
00299     uint32_t        nAddXfr;
00300     uint32_t        nModNext;
00301     uint32_t        nBDUsed;
00302     uint32_t        nBDFreed;
00303
00304     int GetFreeDesc(dma_descview_0 *next);
00305     dma_descview_0 * ModifyNextActive(uint32_t addr, uint32_t unumXfrs, uint32_t suspThresh);
00306     void ConfigFirstAndEnableCh(uint32_t addr, uint32_t numXfrs);
00307     dma_descview_0 * AddXfr(uint32_t addr, uint32_t numXfrs, uint32_t suspThresh);
00308
00309     void PrintCounts();
00310     void PrintDescChain();
00311 };
00312
00313
00314
00315
00316 #endif   /* ----- #ifndef __XDMAC_H  ----- */
```

24.64 basictypes.h File Reference

Platform specific basic type definitions.

`#include <stddef.h>`

24.64.1 Detailed Description

Platform specific basic type definitions.

24.65 coldfire/include/basictypes.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00010 /*
00011  * Defintions of the basic data types.
00012  */
00013
00014 #ifndef _BASICTYPES_H_
00015 #define _BASICTYPES_H_
00016 #include <stddef.h>
00017
00018 typedef unsigned char BOOL;
00019 typedef unsigned char BOOLEAN;
00020 typedef volatile unsigned char VBOOLEAN;
00021 typedef VBOOLEAN *PVBOOLEAN;
00022 typedef BOOLEAN *PBOOLEAN;
00023 typedef const BOOLEAN *PCBOOLEAN;
00024
00025 typedef const char *PCSTR;
00026 typedef char *PSTR;
00027
00028 #ifdef TRUE
00029 #undef TRUE
00030 #endif
00031 #define TRUE (1)
00032 #ifdef FALSE
00033 #undef FALSE
00034 #endif
00035 #define FALSE (0)
00036 #ifdef NULL
00037 #undef NULL
00038 #endif
00039 #define NULL (0)
00040
00041 typedef signed char int8_t;
```

```
00042 typedef unsigned char uint8_t;
00043 typedef volatile signed char vint8_t;
00044 typedef volatile unsigned char vuint8_t;
00045
00046 typedef signed short int16_t;
00047 typedef unsigned short uint16_t;
00048 typedef volatile signed short vint16_t;
00049 typedef volatile unsigned short vuint16_t;
00050
00051 typedef signed long int32_t;
00052 typedef unsigned long uint32_t;
00053 typedef volatile signed long vint32_t;
00054 typedef volatile unsigned long vuint32_t;
00055
00056 typedef signed long long int64_t;
00057 typedef unsigned long long uint64_t;
00058 typedef volatile signed long long vint64_t;
00059 typedef volatile unsigned long long vuint64_t;
00060
00061 typedef signed char INT8;
00062 typedef unsigned char UINT8;
00063 typedef volatile signed char VINT8;
00064 typedef volatile unsigned char VUINT8;
00065
00066 typedef signed short INT16;
00067 typedef unsigned short UINT16;
00068 typedef volatile signed short VINT16;
00069 typedef volatile unsigned short VUINT16;
00070
00071 typedef signed long INT32;
00072 typedef unsigned long UINT32;
00073 typedef volatile signed long VINT32;
00074 typedef volatile unsigned long VUINT32;
00075
00076 typedef vuint8_t *pvuint8_t;
00077 typedef vint8_t *pvint8_t;
00078 typedef vint16_t *pvint16_t;
00079 typedef vuint16_t *pvuint16_t;
00080 typedef vuint32_t *pvuint32_t;
00081 typedef vint32_t *pvint32_t;
00082
00083 typedef unsigned char *puint8_t;
00084 typedef signed char *pint8_t;
00085 typedef signed short *pint16_t;
00086 typedef unsigned short *puint16_t;
00087 typedef unsigned long *puint32_t;
00088 typedef signed long *pint32_t;
00089
00090 typedef const unsigned char *pcuint8_t;
00091 typedef const signed char *pcint8_t;
00092 typedef const signed short *pcint16_t;
00093 typedef const unsigned short *pcuint16_t;
00094 typedef const unsigned long *pcuint32_t;
00095 typedef const signed long *pcint32_t;
00096
00097 #define beint16_t int16_t
00098 #define beuint16_t uint16_t
00099 #define beuint32_t uint32_t
00100 #define beint32_t int32_t
00101
00102 #define pbeint16_t pint16_t
00103 #define pbeuint16_t puint16_t
00104 #define pbeuint32_t puint32_t
00105 #define pbeint32_t pint32_t
00106
00107 #define vbeint16_t vint16_t
00108 #define vbeuint16_t vuint16_t
00109 #define vbeuint32_t vuint32_t
00110 #define vbeint32_t vint32_t
00111
00112 #define pvbeint16_t pvint16_t
00113 #define pvbeuint16_t pvuint16_t
00114 #define pvbeuint32_t pvuint32_t
00115 #define pvbeint32_t pvint32_t
00116
00117 typedef float befloat;
00118 typedef double bedouble;
00119 typedef volatile float vbefloat;
00120 typedef volatile double vbedouble;
00121 typedef float *pbefloat;
00122 typedef double *pbedouble;
00123 typedef volatile float *pvbefloat;
00124 typedef volatile double *pvbedouble;
00125
00126 #ifdef __cplusplus
00127 typedef volatile bool vbool_t;
00128 typedef bool *pbool_t;
```

```
00129 typedef vbool_t *pvbool_t;
00130
00131 inline uint16_t __REV16(uint16_t v)
00132 {
00133     return ((v » 8) & 0xFF) + ((v « 8) & 0xFF00);
00134 }
00135 inline int16_t __REVSH(int16_t v)
00136 {
00137     return ((v » 8) & 0xFF) + ((v « 8) & 0xFF00);
00138 };
00139 inline uint32_t __REV(uint32_t v)
00140 {
00141     return ((v » 24) & 0XFF) + ((v » 8) & 0XFF00) + ((v « 24) & 0XFF000000) + ((v « 8) & 0x00FF0000);
00142 };
00143 inline int32_t __REV(int32_t v)
00144 {
00145     return ((v » 24) & 0XFF) + ((v » 8) & 0XFF00) + ((v « 24) & 0XFF000000) + ((v « 8) & 0x00FF0000);
00146 };
00147
00148 extern "C"
00149 void * memcpy(void *,const void*,size_t);
00150 inline float __REV(float f)
00151 {
00152     uint32_t v;
00153     memcpy(&v, &f, sizeof(float));
00154     v = ((v » 24) & 0XFF) + ((v » 8) & 0XFF00) + ((v « 24) & 0XFF000000) + ((v « 8) & 0x00FF0000);
00155     memcpy(&f, &v, sizeof(float));
00156 //    f = *(float *)&v;
00157     return f;
00158 }
00159
00160 template<typename T>
00161 class LEu16
00162 {
00163     T val;
00164
00165   public:
00166     LEu16() = default;
00167     //    inline LEu16() : val(0) {}
00168     inline LEu16(T rhs) : val(__REV16(rhs)) {}
00169     inline LEu16(uint32_t &rhs) : val(__REV16((uint16_t)rhs)) {}
00170     inline T operator=(T rhs)
00171     {
00172         val = __REV16(rhs);
00173         return rhs;
00174     }
00175     inline operator T() const { return __REV16(val); }
00176     inline explicit operator uint32_t() const { return __REV16(val); }
00177     inline T getMem() { return val; }
00178
00179     inline T operator^=(T rhs)
00180     {
00181         val ^= __REV16(rhs);
00182         return __REV16(val);
00183     }
00184     inline T operator|=(T rhs)
00185     {
00186         val |= __REV16(rhs);
00187         return __REV16(val);
00188     }
00189     inline T operator&=(T rhs)
00190     {
00191         val &= __REV16(rhs);
00192         return __REV16(val);
00193     }
00194     inline T operator%=(T rhs)
00195     {
00196         T ret = __REV16(val) % rhs;
00197         val = __REV16(ret);
00198         return ret;
00199     }
00200     inline T operator+=(T rhs)
00201     {
00202         T ret = __REV16(val) + rhs;
00203         val = __REV16(ret);
00204         return ret;
00205     }
00206     inline T operator-=(T rhs)
00207     {
00208         T ret = __REV16(val) - rhs;
00209         val = __REV16(ret);
00210         return ret;
00211     }
00212
00213     inline T operator++()
00214     {
00215         T ret = __REV16(val) + 1;
```

```
00216         val = __REV16(ret);
00217         return ret;
00218     }
00219     inline T operator--()
00220     {
00221         T ret = __REV16(val) - 1;
00222         val = __REV16(ret);
00223         return ret;
00224     }
00225     inline T operator++(int)
00226     {
00227         T ret = __REV16(val);
00228         val = __REV16(ret + 1);
00229         return ret;
00230     }
00231     inline T operator--(int)
00232     {
00233         T ret = __REV16(val);
00234         val = __REV16(ret - 1);
00235         return ret;
00236     }
00237 } __attribute__((packed));
00238
00239 template<typename T>
00240 class LEs16
00241 {
00242     T val;
00243
00244   public:
00245     LEs16() = default;
00246     //    inline LEs16() : val(0) {}
00247     inline LEs16(T rhs) : val(__REV16(rhs)) {}
00248     inline LEs16(int32_t &rhs) : val(__REVSH((int16_t)rhs)) {}
00249     inline T operator=(T rhs)
00250     {
00251         val = __REVSH(rhs);
00252         return rhs;
00253     }
00254     inline operator T() const { return __REVSH(val); }
00255     inline explicit operator int32_t() const { return (int32_t)(__REV16(val)); }
00256     inline T getMem() { return val; }
00257
00258     inline T operator^=(T rhs)
00259     {
00260         val ^= __REVSH(rhs);
00261         return __REVSH(val);
00262     }
00263     inline T operator|=(T rhs)
00264     {
00265         val |= __REVSH(rhs);
00266         return __REVSH(val);
00267     }
00268     inline T operator&=(T rhs)
00269     {
00270         val &= __REVSH(rhs);
00271         return __REVSH(val);
00272     }
00273     inline T operator%=(T rhs)
00274     {
00275         T ret = __REVSH(val) % rhs;
00276         val = __REVSH(ret);
00277         return ret;
00278     }
00279     inline T operator+=(T rhs)
00280     {
00281         T ret = __REVSH(val) + rhs;
00282         val = __REVSH(ret);
00283         return ret;
00284     }
00285     inline T operator-=(T rhs)
00286     {
00287         T ret = __REVSH(val) - rhs;
00288         val = __REVSH(ret);
00289         return ret;
00290     }
00291
00292     inline T operator++()
00293     {
00294         T ret = __REVSSH(val) + 1;
00295         val = __REV16(ret);
00296         return ret;
00297     }
00298     inline T operator--()
00299     {
00300         T ret = __REVSH(val) - 1;
00301         val = __REVSH(ret);
00302         return ret;
```

```
00303     }
00304     inline T operator++(int)
00305     {
00306         T ret = __REVSH(val);
00307         val = __REVSH(ret + 1);
00308         return ret;
00309     }
00310     inline T operator--(int)
00311     {
00312         T ret = __REVSH(val);
00313         val = __REVSH(ret - 1);
00314         return ret;
00315     }
00316 } __attribute__((packed));
00317
00318 template<typename T>
00319 class LE32
00320 {
00321     T val;
00322
00323   public:
00324     LE32() = default;
00325     //    inline LE32() : val(0) {}
00326     inline LE32(T rhs) : val(__REV(rhs)) {}
00327     //    LE32(IPADDR4 rhs);
00328     inline T operator=(T rhs)
00329     {
00330         val = __REV(rhs);
00331         return rhs;
00332     }
00333     inline operator T() const { return __REV(val); }
00334     inline T getMem() { return val; }
00335
00336     inline T operator^=(T rhs)
00337     {
00338         val ^= __REV(rhs);
00339         return __REV(val);
00340     }
00341     inline T operator|=(T rhs)
00342     {
00343         val |= __REV(rhs);
00344         return __REV(val);
00345     }
00346     inline T operator&=(T rhs)
00347     {
00348         val &= __REV(rhs);
00349         return __REV(val);
00350     }
00351     inline T operator%=(T rhs)
00352     {
00353         T ret = __REV(val) % rhs;
00354         val = __REV(ret);
00355         return ret;
00356     }
00357     inline T operator+=(T rhs)
00358     {
00359         T ret = __REV(val) + rhs;
00360         val = __REV(ret);
00361         return ret;
00362     }
00363     inline T operator-=(T rhs)
00364     {
00365         T ret = __REV(val) - rhs;
00366         val = __REV(ret);
00367         return ret;
00368     }
00369
00370     inline T operator++()
00371     {
00372         T ret = __REVSSH(val) + 1;
00373         val = __REV16(ret);
00374         return ret;
00375     }
00376     inline T operator--()
00377     {
00378         T ret = __REV(val) - 1;
00379         val = __REV(ret);
00380         return ret;
00381     }
00382     inline T operator++(int)
00383     {
00384         T ret = __REV(val);
00385         val = __REV(ret + 1);
00386         return ret;
00387     }
00388     inline T operator--(int)
00389     {
```

```
00390         T ret = __REV(val);
00391         val = __REV(ret - 1);
00392         return ret;
00393     }
00394
00395 } __attribute__((packed));
00396
00397 class LE_Float
00398 {
00399     union {
00400         float val;
00401         uint8_t valBuf[4];
00402     };
00403
00404   public:
00405     LE_Float() = default;
00406     //    inline BE32() : val(0) {}
00407     inline LE_Float(float rhs) : val(__REV(rhs)) {}
00408     //    BE32(IPADDR4 rhs);
00409     inline float operator=(float rhs)
00410     {
00411         val = __REV(rhs);
00412         return rhs;
00413     }
00414     inline operator float() const { return __REV(val); }
00415     inline float getMem() { return val; }
00416
00417     inline float operator+=(float rhs)
00418     {
00419         float ret = __REV(val) + rhs;
00420         val = __REV(ret);
00421         return ret;
00422     }
00423     inline float operator-=(float rhs)
00424     {
00425         float ret = __REV(val) - rhs;
00426         val = __REV(ret);
00427         return ret;
00428     }
00429
00430     inline float operator++()
00431     {
00432         float ret = __REV(val) + 1;
00433         val = __REV16(ret);
00434         return ret;
00435     }
00436     inline float operator--()
00437     {
00438         float ret = __REV(val) - 1;
00439         val = __REV(ret);
00440         return ret;
00441     }
00442     inline float operator++(int)
00443     {
00444         float ret = __REV(val);
00445         val = __REV(ret + 1);
00446         return ret;
00447     }
00448     inline float operator--(int)
00449     {
00450         float ret = __REV(val);
00451         val = __REV(ret - 1);
00452         return ret;
00453     }
00454
00455     inline float operator=(float rhs) volatile
00456     {
00457         val = __REV(rhs);
00458         return rhs;
00459     }
00460     inline float getMem() volatile { return val; }
00461     inline operator float() const volatile { return __REV(val); }
00462
00463     inline float operator+=(float rhs) volatile
00464     {
00465         float ret = __REV(val) + rhs;
00466         val = __REV(ret);
00467         return ret;
00468     }
00469     inline float operator-=(float rhs) volatile
00470     {
00471         float ret = __REV(val) - rhs;
00472         val = __REV(ret);
00473         return ret;
00474     }
00475
00476     inline float operator++() volatile
```

```
00477     {
00478         float ret = __REV(val) + 1;
00479         val = __REV16(ret);
00480         return ret;
00481     }
00482     inline float operator--() volatile
00483     {
00484         float ret = __REV(val) - 1;
00485         val = __REV(ret);
00486         return ret;
00487     }
00488     inline float operator++(int) volatile
00489     {
00490         float ret = __REV(val);
00491         val = __REV(ret + 1);
00492         return ret;
00493     }
00494     inline float operator--(int) volatile
00495     {
00496         float ret = __REV(val);
00497         val = __REV(ret - 1);
00498         return ret;
00499     }
00500
00501     //    friend inline float operator«(BE32<T> lhs, unsigned int shift)
00502     //        { return (__REV(lhs.val)) « shift; }
00503     //    friend inline float operator»(BE32<T> lhs, unsigned int shift)
00504     //        { return (__REV(lhs.val)) » shift; }
00505 } __attribute__((packed));
00506
00507 class LE_Double
00508 {
00509     union {
00510         double val;
00511         uint8_t valBuf[8];
00512     };
00513
00514    public:
00515     LE_Double() = default;
00516     //    inline BE32() : val(0) {}
00517     inline LE_Double(double rhs)
00518         :valBuf{
00519             (uint8_t)(((uint64_t)rhs)»56),(uint8_t)(((uint64_t)rhs)»48),
00520             (uint8_t)(((uint64_t)rhs)»40),(uint8_t)(((uint64_t)rhs)»32),
00521             (uint8_t)(((uint64_t)rhs)»24),(uint8_t)(((uint64_t)rhs)»16),
00522             (uint8_t)(((uint64_t)rhs)» 8),(uint8_t)(((uint64_t)rhs)» 0)
00523         }
00524     { }
00525     //    BE32(IPADDR4 rhs);
00526     inline double operator=(double rhs)
00527     {
00528         valBuf[0] = (uint8_t)(((uint64_t)rhs)»56);
00529         valBuf[1] = (uint8_t)(((uint64_t)rhs)»48),
00530         valBuf[2] = (uint8_t)(((uint64_t)rhs)»40);
00531         valBuf[3] = (uint8_t)(((uint64_t)rhs)»32);
00532         valBuf[4] = (uint8_t)(((uint64_t)rhs)»24);
00533         valBuf[5] = (uint8_t)(((uint64_t)rhs)»16);
00534         valBuf[6] = (uint8_t)(((uint64_t)rhs)» 8);
00535         valBuf[7] = (uint8_t)(((uint64_t)rhs)» 0);
00536         return rhs;
00537     }
00538     inline operator double() const
00539     {
00540         return (double)
00541             (
00542              (((uint64_t)valBuf[0])«56) | (((uint64_t)valBuf[0])«48)
00543                 | (((uint64_t)valBuf[0])«40) | (((uint64_t)valBuf[0])«32)
00544                 | (((uint64_t)valBuf[0])«24) | (((uint64_t)valBuf[0])«16)
00545                 | (((uint64_t)valBuf[0])« 8) | (((uint64_t)valBuf[0])« 0)
00546                 );
00547     }
00548     inline double getMem() { return val; }
00549
00550     inline double operator+=(double rhs)
00551     {
00552         double ret = ((double)(*this)) + rhs;
00553         *this = ret;
00554         return ret;
00555     }
00556     inline double operator-=(double rhs)
00557     {
00558         double ret = ((double)(*this)) - rhs;
00559         *this = ret;
00560         return ret;
00561     }
00562
00563     inline double operator++()
```

```
00564     {
00565         double ret = ((double)(*this)) + 1;
00566         *this = ret;
00567         return ret;
00568     }
00569     inline double operator--()
00570     {
00571         double ret = ((double)(*this)) - 1;
00572         *this = ret;
00573         return ret;
00574     }
00575     inline double operator++(int)
00576     {
00577         double ret = ((double)(*this));
00578         *this = ret + 1;
00579         return ret;
00580     }
00581     inline double operator--(int)
00582     {
00583         double ret = ((double)(*this));
00584         *this = ret - 1;
00585         return ret;
00586     }
00587
00588     inline double operator=(double rhs) volatile
00589     {
00590         valBuf[0] = (uint8_t)(((uint64_t)rhs)»56);
00591         valBuf[1] = (uint8_t)(((uint64_t)rhs)»48),
00592         valBuf[2] = (uint8_t)(((uint64_t)rhs)»40);
00593         valBuf[3] = (uint8_t)(((uint64_t)rhs)»32);
00594         valBuf[4] = (uint8_t)(((uint64_t)rhs)»24);
00595         valBuf[5] = (uint8_t)(((uint64_t)rhs)»16);
00596         valBuf[6] = (uint8_t)(((uint64_t)rhs)» 8);
00597         valBuf[7] = (uint8_t)(((uint64_t)rhs)» 0);
00598         return rhs;
00599     }
00600     inline double getMem() volatile { return val; }
00601     inline operator double() const volatile
00602     {
00603         return (double)
00604             (
00605              (((uint64_t)valBuf[0])«56) | (((uint64_t)valBuf[0])«48)
00606                 | (((uint64_t)valBuf[0])«40) | (((uint64_t)valBuf[0])«32)
00607                 | (((uint64_t)valBuf[0])«24) | (((uint64_t)valBuf[0])«16)
00608                 | (((uint64_t)valBuf[0])« 8) | (((uint64_t)valBuf[0])« 0)
00609                 );
00610     }
00611
00612     inline double operator+=(double rhs) volatile
00613     {
00614         double ret = ((double)(*this)) + rhs;
00615         *this = ret;
00616         return ret;
00617     }
00618     inline double operator-=(double rhs) volatile
00619     {
00620         double ret = ((double)(*this)) - rhs;
00621         *this = ret;
00622         return ret;
00623     }
00624
00625     inline double operator++() volatile
00626     {
00627         double ret = ((double)(*this)) + 1;
00628         *this = ret;
00629         return ret;
00630     }
00631     inline double operator--() volatile
00632     {
00633         double ret = ((double)(*this)) - 1;
00634         *this = ret;
00635         return ret;
00636     }
00637     inline double operator++(int) volatile
00638     {
00639         double ret = ((double)(*this));
00640         *this = ret + 1;
00641         return ret;
00642     }
00643     inline double operator--(int) volatile
00644     {
00645         double ret = ((double)(*this));
00646         *this = ret - 1;
00647         return ret;
00648     }
00649
00650     //    friend inline double operator«(BE32<T> lhs, unsigned int shift)
```

```
00651     //        { return (__REV(lhs.val)) « shift; }
00652     //    friend inline double operator»(BE32<T> lhs, unsigned int shift)
00653     //        { return (__REV(lhs.val)) » shift; }
00654 } __attribute__((packed));
00655
00656 typedef LEs16<int16_t> leint16_t;
00657 typedef LEu16<uint16_t> leuint16_t;
00658 typedef LE32<uint32_t> leuint32_t;
00659 typedef LE32<uint32_t> leint32_t;
00660
00661 typedef LEs16<int16_t> *pleint16_t;
00662 typedef LEu16<uint16_t> *pleuint16_t;
00663 typedef LE32<uint32_t> *pleuint32_t;
00664 typedef LE32<uint32_t> *pleint32_t;
00665
00666 typedef LEs16<volatile int16_t> vleint16_t;
00667 typedef LEu16<volatile uint16_t> vleuint16_t;
00668 typedef LE32<volatile uint32_t> vleuint32_t;
00669 typedef LE32<volatile uint32_t> vleint32_t;
00670
00671 typedef LEs16<volatile int16_t> *pvleint16_t;
00672 typedef LEu16<volatile uint16_t> *pvleuint16_t;
00673 typedef LE32<volatile uint32_t> *pvleuint32_t;
00674 typedef LE32<volatile uint32_t> *pvleint32_t;
00675
00676 typedef LE_Float lefloat;
00677 typedef LE_Double ledouble;
00678 typedef volatile LE_Float vlefloat;
00679 typedef volatile LE_Double vledouble;
00680 typedef LE_Float *plefloat;
00681 typedef LE_Double *pledouble;
00682 typedef volatile LE_Float *pvlefloat;
00683 typedef volatile LE_Double *pvledouble;
00684
00685 #ifdef MAKE_LEINT_TEST
00686 #include <stdio.h>
00687 void LEINT_TEST()
00688 {
00689     uint8_t buf[4];
00690     buf[0] = 0x01;
00691     buf[1] = 0x23;
00692     buf[2] = 0x45;
00693     buf[3] = 0x67;
00694
00695     iprintf("int16_t=%08X\r\n", (int)(*((int16_t *)buf)));
00696     iprintf("uint16_t=%08X\r\n", (unsigned int)(*((uint16_t *)buf)));
00697     iprintf("int32_t=%08X\r\n", (int)(*((int32_t *)buf)));
00698     iprintf("uint32_t=%08X\r\n", (unsigned int)(*((uint32_t *)buf)));
00699
00700     iprintf("leint16_t=%08X\r\n", (int)(*((leint16_t *)buf)));
00701     iprintf("leuint16_t=%08X\r\n", (unsigned int)(*((leuint16_t *)buf)));
00702     iprintf("leint32_t=%08X\r\n", (int)(*((leint32_t *)buf)));
00703     iprintf("leuint32_t=%08X\r\n", (unsigned int)(*((leuint32_t *)buf)));
00704 }
00705 #endif
00706 #endif
00707
00708 #define uint32_t_max (0xFFFFFFFF)
00709
00710 #define HTONS(x) (x)   // HTOBES(x)
00711 #define HTONL(x) (x)   // HTOBEL(x)
00712 #define NTOHL(x) (x)   // HTOBEL(x)
00713 #define NTOHS(x) (x)   // HTOBES(x)
00714
00715 #endif /* _BASICTYPES_H_ */
```

24.66 basictypes.h File Reference

Platform specific basic type definitions.

24.66.1 Detailed Description

Platform specific basic type definitions.

24.67 cortex-m7/include/basictypes.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
```

```
00003 /*NB_COPYRIGHT*/
00004
00010 /*
00011  * Defintions of the basic data types.
00012  */
00013
00014 #ifndef _BASICTYPES_H_
00015 #define _BASICTYPES_H_
00016
00017 typedef unsigned char BOOL;
00018 typedef unsigned char BOOLEAN;
00019 typedef volatile unsigned char VBOOLEAN;
00020 typedef VBOOLEAN *PVBOOLEAN;
00021 typedef BOOLEAN *PBOOLEAN;
00022 typedef const BOOLEAN *PCBOOLEAN;
00023
00024 typedef const char *PCSTR;
00025 typedef char *PSTR;
00026
00027 #ifdef TRUE
00028 #undef TRUE
00029 #endif
00030 #define TRUE (1)
00031 #ifdef FALSE
00032 #undef FALSE
00033 #endif
00034 #define FALSE (0)
00035 #ifdef NULL
00036 #undef NULL
00037 #endif
00038 #define NULL (0)
00039
00040 // TYPEDEFS needed for Freescale ETPU API
00041 typedef signed char int8_t;
00042 typedef unsigned char uint8_t;
00043 typedef volatile signed char vint8_t;
00044 typedef volatile unsigned char vuint8_t;
00045
00046 typedef signed short int16_t;
00047 typedef unsigned short uint16_t;
00048 typedef volatile signed short vint16_t;
00049 typedef volatile unsigned short vuint16_t;
00050
00051 typedef signed long int32_t;
00052 typedef unsigned long uint32_t;
00053 typedef volatile signed long vint32_t;
00054 typedef volatile unsigned long vuint32_t;
00055
00056 typedef signed char INT8;
00057 typedef unsigned char UINT8;
00058 typedef volatile signed char VINT8;
00059 typedef volatile unsigned char VUINT8;
00060
00061 typedef signed short INT16;
00062 typedef unsigned short UINT16;
00063 typedef volatile signed short VINT16;
00064 typedef volatile unsigned short VUINT16;
00065
00066 typedef signed long INT32;
00067 typedef unsigned long UINT32;
00068 typedef volatile signed long VINT32;
00069 typedef volatile unsigned long VUINT32;
00070
00071 typedef vuint8_t *pvuint8_t;
00072 typedef vint8_t *pvint8_t;
00073 typedef vint16_t *pvint16_t;
00074 typedef vuint16_t *pvuint16_t;
00075 typedef vuint32_t *pvuint32_t;
00076 typedef vint32_t *pvint32_t;
00077
00078 typedef unsigned char *puint8_t;
00079 typedef signed char *pint8_t;
00080 typedef signed short *pint16_t;
00081 typedef unsigned short *puint16_t;
00082 typedef unsigned long *puint32_t;
00083 typedef signed long *pint32_t;
00084
00085 #ifdef __cplusplus
00086
00087 #include <cm_core_config.h>
00088 #include <cmsis_compiler.h>
00089 //#include <core_cmInstr.h>
00090 template<typename T>
00091 class BEu16
00092 {
00093     T val;
00094
```

```
00095     public:
00096      BEu16() = default;
00097      //    inline BEu16() : val(0) {}
00098      inline BEu16(T rhs) : val(__REV16(rhs)) {}
00099      inline BEu16(uint32_t &rhs) : val(__REV16((uint16_t)rhs)) {}
00100      inline T operator=(T rhs)
00101      {
00102          val = __REV16(rhs);
00103          return rhs;
00104      }
00105      inline operator T() const { return __REV16(val); }
00106      inline explicit operator uint32_t() const { return __REV16(val); }
00107      inline T getMem() { return val; }
00108
00109      inline T operator^=(T rhs)
00110      {
00111          val ^= __REV16(rhs);
00112          return __REV16(val);
00113      }
00114      inline T operator|=(T rhs)
00115      {
00116          val |= __REV16(rhs);
00117          return __REV16(val);
00118      }
00119      inline T operator&=(T rhs)
00120      {
00121          val &= __REV16(rhs);
00122          return __REV16(val);
00123      }
00124      inline T operator%=(T rhs)
00125      {
00126          T ret = __REV16(val) % rhs;
00127          val = __REV16(ret);
00128          return ret;
00129      }
00130      inline T operator+=(T rhs)
00131      {
00132          T ret = __REV16(val) + rhs;
00133          val = __REV16(ret);
00134          return ret;
00135      }
00136      inline T operator-=(T rhs)
00137      {
00138          T ret = __REV16(val) - rhs;
00139          val = __REV16(ret);
00140          return ret;
00141      }
00142
00143      inline T operator++()
00144      {
00145          T ret = __REV16(val) + 1;
00146          val = __REV16(ret);
00147          return ret;
00148      }
00149      inline T operator--()
00150      {
00151          T ret = __REV16(val) - 1;
00152          val = __REV16(ret);
00153          return ret;
00154      }
00155      inline T operator++(int)
00156      {
00157          T ret = __REV16(val);
00158          val = __REV16(ret + 1);
00159          return ret;
00160      }
00161      inline T operator--(int)
00162      {
00163          T ret = __REV16(val);
00164          val = __REV16(ret - 1);
00165          return ret;
00166      }
00167
00168      inline T operator=(T rhs) volatile
00169      {
00170          val = __REV16(rhs);
00171          return rhs;
00172      }
00173      inline operator T() const volatile { return __REV16(val); }
00174      inline explicit operator uint32_t() const volatile { return __REV16(val); }
00175      inline T getMem() volatile { return val; }
00176
00177      inline T operator^=(T rhs) volatile
00178      {
00179          val ^= __REV16(rhs);
00180          return __REV16(val);
00181      }
```

```
00182     inline T operator|=(T rhs) volatile
00183     {
00184         val |= __REV16(rhs);
00185         return __REV16(val);
00186     }
00187     inline T operator&=(T rhs) volatile
00188     {
00189         val &= __REV16(rhs);
00190         return __REV16(val);
00191     }
00192     inline T operator%=(T rhs) volatile
00193     {
00194         T ret = __REV16(val) % rhs;
00195         val = __REV16(ret);
00196         return ret;
00197     }
00198     inline T operator+=(T rhs) volatile
00199     {
00200         T ret = __REV16(val) + rhs;
00201         val = __REV16(ret);
00202         return ret;
00203     }
00204     inline T operator-=(T rhs) volatile
00205     {
00206         T ret = __REV16(val) - rhs;
00207         val = __REV16(ret);
00208         return ret;
00209     }
00210
00211     inline T operator++() volatile
00212     {
00213         T ret = __REV16(val) + 1;
00214         val = __REV16(ret);
00215         return ret;
00216     }
00217     inline T operator--() volatile
00218     {
00219         T ret = __REV16(val) - 1;
00220         val = __REV16(ret);
00221         return ret;
00222     }
00223     inline T operator++(int) volatile
00224     {
00225         T ret = __REV16(val);
00226         val = __REV16(ret + 1);
00227         return ret;
00228     }
00229     inline T operator--(int) volatile
00230     {
00231         T ret = __REV16(val);
00232         val = __REV16(ret - 1);
00233         return ret;
00234     }
00235
00236 } __attribute__((packed));
00237
00238 template<typename T>
00239 class BEs16
00240 {
00241     T val;
00242
00243   public:
00244     BEs16() = default;
00245     //    inline BEs16() : val(0) {}
00246     inline BEs16(T rhs) : val(__REV16(rhs)) {}
00247     inline BEs16(int32_t &rhs) : val(__REVSH((int16_t)rhs)) {}
00248     inline T operator=(T rhs)
00249     {
00250         val = __REVSH(rhs);
00251         return rhs;
00252     }
00253     inline operator T() const { return __REVSH(val); }
00254     inline explicit operator int32_t() const { return (int32_t)(__REV16(val)); }
00255     inline T getMem() { return val; }
00256
00257     inline T operator^=(T rhs)
00258     {
00259         val ^= __REVSH(rhs);
00260         return __REVSH(val);
00261     }
00262     inline T operator|=(T rhs)
00263     {
00264         val |= __REVSH(rhs);
00265         return __REVSH(val);
00266     }
00267     inline T operator&=(T rhs)
00268     {
```

```
00269         val &= __REVSH(rhs);
00270         return __REVSH(val);
00271     }
00272     inline T operator%=(T rhs)
00273     {
00274         T ret = __REVSH(val) % rhs;
00275         val = __REVSH(ret);
00276         return ret;
00277     }
00278     inline T operator+=(T rhs)
00279     {
00280         T ret = __REVSH(val) + rhs;
00281         val = __REVSH(ret);
00282         return ret;
00283     }
00284     inline T operator-=(T rhs)
00285     {
00286         T ret = __REVSH(val) - rhs;
00287         val = __REVSH(ret);
00288         return ret;
00289     }
00290
00291     inline T operator++()
00292     {
00293         T ret = __REVSH(val) + 1;
00294         val = __REV16(ret);
00295         return ret;
00296     }
00297     inline T operator--()
00298     {
00299         T ret = __REVSH(val) - 1;
00300         val = __REVSH(ret);
00301         return ret;
00302     }
00303     inline T operator++(int)
00304     {
00305         T ret = __REVSH(val);
00306         val = __REVSH(ret + 1);
00307         return ret;
00308     }
00309     inline T operator--(int)
00310     {
00311         T ret = __REVSH(val);
00312         val = __REVSH(ret - 1);
00313         return ret;
00314     }
00315 } __attribute__((packed));
00316
00317 struct IPADDR4;
00318
00319 template<typename T>
00320 class BE32
00321 {
00322     T val;
00323
00324   public:
00325     BE32() = default;
00326     //    inline BE32() : val(0) {}
00327     inline BE32(T rhs) : val(__REV(rhs)) {}
00328     //    BE32(IPADDR4 rhs);
00329     inline T operator=(T rhs)
00330     {
00331         val = __REV(rhs);
00332         return rhs;
00333     }
00334     inline operator T() const { return __REV(val); }
00335     inline T getMem() { return val; }
00336
00337     inline T operator^=(T rhs)
00338     {
00339         val ^= __REV(rhs);
00340         return __REV(val);
00341     }
00342     inline T operator|=(T rhs)
00343     {
00344         val |= __REV(rhs);
00345         return __REV(val);
00346     }
00347     inline T operator&=(T rhs)
00348     {
00349         val &= __REV(rhs);
00350         return __REV(val);
00351     }
00352     inline T operator%=(T rhs)
00353     {
00354         T ret = __REV(val) % rhs;
00355         val = __REV(ret);
```

```
00356         return ret;
00357     }
00358     inline T operator+=(T rhs)
00359     {
00360         T ret = __REV(val) + rhs;
00361         val = __REV(ret);
00362         return ret;
00363     }
00364     inline T operator-=(T rhs)
00365     {
00366         T ret = __REV(val) - rhs;
00367         val = __REV(ret);
00368         return ret;
00369     }
00370
00371     inline T operator++()
00372     {
00373         T ret = __REVSSH(val) + 1;
00374         val = __REV16(ret);
00375         return ret;
00376     }
00377     inline T operator--()
00378     {
00379         T ret = __REV(val) - 1;
00380         val = __REV(ret);
00381         return ret;
00382     }
00383     inline T operator++(int)
00384     {
00385         T ret = __REV(val);
00386         val = __REV(ret + 1);
00387         return ret;
00388     }
00389     inline T operator--(int)
00390     {
00391         T ret = __REV(val);
00392         val = __REV(ret - 1);
00393         return ret;
00394     }
00395
00396     inline T operator=(T rhs) volatile
00397     {
00398         val = __REV(rhs);
00399         return rhs;
00400     }
00401     inline T getMem() volatile { return val; }
00402     inline operator T() const volatile { return __REV(val); }
00403
00404     inline T operator^=(T rhs) volatile
00405     {
00406         val ^= __REV(rhs);
00407         return __REV(val);
00408     }
00409     inline T operator|=(T rhs) volatile
00410     {
00411         val |= __REV(rhs);
00412         return __REV(val);
00413     }
00414     inline T operator&=(T rhs) volatile
00415     {
00416         val &= __REV(rhs);
00417         return __REV(val);
00418     }
00419     inline T operator%=(T rhs) volatile
00420     {
00421         T ret = __REV(val) % rhs;
00422         val = __REV(ret);
00423         return ret;
00424     }
00425     inline T operator+=(T rhs) volatile
00426     {
00427         T ret = __REV(val) + rhs;
00428         val = __REV(ret);
00429         return ret;
00430     }
00431     inline T operator-=(T rhs) volatile
00432     {
00433         T ret = __REV(val) - rhs;
00434         val = __REV(ret);
00435         return ret;
00436     }
00437
00438     inline T operator++() volatile
00439     {
00440         T ret = __REVSSH(val) + 1;
00441         val = __REV16(ret);
00442         return ret;
```

```
00443     }
00444     inline T operator--() volatile
00445     {
00446         T ret = __REV(val) - 1;
00447         val = __REV(ret);
00448         return ret;
00449     }
00450     inline T operator++(int) volatile
00451     {
00452         T ret = __REV(val);
00453         val = __REV(ret + 1);
00454         return ret;
00455     }
00456     inline T operator--(int) volatile
00457     {
00458         T ret = __REV(val);
00459         val = __REV(ret - 1);
00460         return ret;
00461     }
00462
00463     //    friend inline T operator«(BE32<T> lhs, unsigned int shift)
00464     //        { return (__REV(lhs.val)) « shift; }
00465     //    friend inline T operator»(BE32<T> lhs, unsigned int shift)
00466     //        { return (__REV(lhs.val)) » shift; }
00467 } __attribute__((packed));
00468
00469 class BE_Float
00470 {
00471     union {
00472         float val;
00473         uint8_t valBuf[4];
00474     };
00475
00476    public:
00477     BE_Float() = default;
00478     //    inline BE32() : val(0) {}
00479     inline BE_Float(float rhs) : val(__REV(rhs)) {}
00480     //    BE32(IPADDR4 rhs);
00481     inline float operator=(float rhs)
00482     {
00483         val = __REV(rhs);
00484         return rhs;
00485     }
00486     inline operator float() const { return __REV(val); }
00487     inline float getMem() { return val; }
00488
00489     inline float operator+=(float rhs)
00490     {
00491         float ret = __REV(val) + rhs;
00492         val = __REV(ret);
00493         return ret;
00494     }
00495     inline float operator-=(float rhs)
00496     {
00497         float ret = __REV(val) - rhs;
00498         val = __REV(ret);
00499         return ret;
00500     }
00501
00502     inline float operator++()
00503     {
00504         float ret = __REV(val) + 1;
00505         val = __REV16(ret);
00506         return ret;
00507     }
00508     inline float operator--()
00509     {
00510         float ret = __REV(val) - 1;
00511         val = __REV(ret);
00512         return ret;
00513     }
00514     inline float operator++(int)
00515     {
00516         float ret = __REV(val);
00517         val = __REV(ret + 1);
00518         return ret;
00519     }
00520     inline float operator--(int)
00521     {
00522         float ret = __REV(val);
00523         val = __REV(ret - 1);
00524         return ret;
00525     }
00526
00527     inline float operator=(float rhs) volatile
00528     {
00529         val = __REV(rhs);
```

```
00530         return rhs;
00531     }
00532     inline float getMem() volatile { return val; }
00533     inline operator float() const volatile { return __REV(val); }
00534
00535     inline float operator+=(float rhs) volatile
00536     {
00537         float ret = __REV(val) + rhs;
00538         val = __REV(ret);
00539         return ret;
00540     }
00541     inline float operator-=(float rhs) volatile
00542     {
00543         float ret = __REV(val) - rhs;
00544         val = __REV(ret);
00545         return ret;
00546     }
00547
00548     inline float operator++() volatile
00549     {
00550         float ret = __REV(val) + 1;
00551         val = __REV16(ret);
00552         return ret;
00553     }
00554     inline float operator--() volatile
00555     {
00556         float ret = __REV(val) - 1;
00557         val = __REV(ret);
00558         return ret;
00559     }
00560     inline float operator++(int) volatile
00561     {
00562         float ret = __REV(val);
00563         val = __REV(ret + 1);
00564         return ret;
00565     }
00566     inline float operator--(int) volatile
00567     {
00568         float ret = __REV(val);
00569         val = __REV(ret - 1);
00570         return ret;
00571     }
00572
00573     //    friend inline float operator«(BE32<T> lhs, unsigned int shift)
00574     //        { return (__REV(lhs.val)) « shift; }
00575     //    friend inline float operator»(BE32<T> lhs, unsigned int shift)
00576     //        { return (__REV(lhs.val)) » shift; }
00577 } __attribute__((packed));
00578
00579 class BE_Double
00580 {
00581     union {
00582         double val;
00583         uint8_t valBuf[8];
00584     };
00585
00586    public:
00587     BE_Double() = default;
00588     //    inline BE32() : val(0) {}
00589     inline BE_Double(double rhs)
00590         :valBuf{
00591             (uint8_t)(((uint64_t)rhs)»56),(uint8_t)(((uint64_t)rhs)»48),
00592             (uint8_t)(((uint64_t)rhs)»40),(uint8_t)(((uint64_t)rhs)»32),
00593             (uint8_t)(((uint64_t)rhs)»24),(uint8_t)(((uint64_t)rhs)»16),
00594             (uint8_t)(((uint64_t)rhs)» 8),(uint8_t)(((uint64_t)rhs)» 0)
00595         }
00596     { }
00597     //    BE32(IPADDR4 rhs);
00598     inline double operator=(double rhs)
00599     {
00600         valBuf[0] = (uint8_t)(((uint64_t)rhs)»56);
00601         valBuf[1] = (uint8_t)(((uint64_t)rhs)»48),
00602         valBuf[2] = (uint8_t)(((uint64_t)rhs)»40);
00603         valBuf[3] = (uint8_t)(((uint64_t)rhs)»32);
00604         valBuf[4] = (uint8_t)(((uint64_t)rhs)»24);
00605         valBuf[5] = (uint8_t)(((uint64_t)rhs)»16);
00606         valBuf[6] = (uint8_t)(((uint64_t)rhs)» 8);
00607         valBuf[7] = (uint8_t)(((uint64_t)rhs)» 0);
00608         return rhs;
00609     }
00610     inline operator double() const
00611     {
00612         return (double)
00613             (
00614              (((uint64_t)valBuf[0])«56) | (((uint64_t)valBuf[0])«48)
00615                 | (((uint64_t)valBuf[0])«40) | (((uint64_t)valBuf[0])«32)
00616                 | (((uint64_t)valBuf[0])«24) | (((uint64_t)valBuf[0])«16)
```

```
00617                 | (((uint64_t)valBuf[0])« 8) | (((uint64_t)valBuf[0])« 0)
00618                 );
00619     }
00620     inline double getMem() { return val; }
00621
00622     inline double operator+=(double rhs)
00623     {
00624         double ret = ((double)(*this)) + rhs;
00625         *this = ret;
00626         return ret;
00627     }
00628     inline double operator-=(double rhs)
00629     {
00630         double ret = ((double)(*this)) - rhs;
00631         *this = ret;
00632         return ret;
00633     }
00634
00635     inline double operator++()
00636     {
00637         double ret = ((double)(*this)) + 1;
00638         *this = ret;
00639         return ret;
00640     }
00641     inline double operator--()
00642     {
00643         double ret = ((double)(*this)) - 1;
00644         *this = ret;
00645         return ret;
00646     }
00647     inline double operator++(int)
00648     {
00649         double ret = ((double)(*this));
00650         *this = ret + 1;
00651         return ret;
00652     }
00653     inline double operator--(int)
00654     {
00655         double ret = ((double)(*this));
00656         *this = ret - 1;
00657         return ret;
00658     }
00659
00660     inline double operator=(double rhs) volatile
00661     {
00662         valBuf[0] = (uint8_t)(((uint64_t)rhs)»56);
00663         valBuf[1] = (uint8_t)(((uint64_t)rhs)»48),
00664         valBuf[2] = (uint8_t)(((uint64_t)rhs)»40);
00665         valBuf[3] = (uint8_t)(((uint64_t)rhs)»32);
00666         valBuf[4] = (uint8_t)(((uint64_t)rhs)»24);
00667         valBuf[5] = (uint8_t)(((uint64_t)rhs)»16);
00668         valBuf[6] = (uint8_t)(((uint64_t)rhs)» 8);
00669         valBuf[7] = (uint8_t)(((uint64_t)rhs)» 0);
00670         return rhs;
00671     }
00672     inline double getMem() volatile { return val; }
00673     inline operator double() const volatile
00674     {
00675         return (double)
00676             (
00677              (((uint64_t)valBuf[0])«56) | (((uint64_t)valBuf[0])«48)
00678                 | (((uint64_t)valBuf[0])«40) | (((uint64_t)valBuf[0])«32)
00679                 | (((uint64_t)valBuf[0])«24) | (((uint64_t)valBuf[0])«16)
00680                 | (((uint64_t)valBuf[0])« 8) | (((uint64_t)valBuf[0])« 0)
00681                 );
00682     }
00683
00684     inline double operator+=(double rhs) volatile
00685     {
00686         double ret = ((double)(*this)) + rhs;
00687         *this = ret;
00688         return ret;
00689     }
00690     inline double operator-=(double rhs) volatile
00691     {
00692         double ret = ((double)(*this)) - rhs;
00693         *this = ret;
00694         return ret;
00695     }
00696
00697     inline double operator++() volatile
00698     {
00699         double ret = ((double)(*this)) + 1;
00700         *this = ret;
00701         return ret;
00702     }
00703     inline double operator--() volatile
```

```
00704     {
00705         double ret = ((double)(*this)) - 1;
00706         *this = ret;
00707         return ret;
00708     }
00709     inline double operator++(int) volatile
00710     {
00711         double ret = ((double)(*this));
00712         *this = ret + 1;
00713         return ret;
00714     }
00715     inline double operator--(int) volatile
00716     {
00717         double ret = ((double)(*this));
00718         *this = ret - 1;
00719         return ret;
00720     }
00721
00722     //    friend inline double operator«(BE32<T> lhs, unsigned int shift)
00723     //        { return (__REV(lhs.val)) « shift; }
00724     //    friend inline double operator»(BE32<T> lhs, unsigned int shift)
00725     //        { return (__REV(lhs.val)) » shift; }
00726 } __attribute__((packed));
00727
00728 typedef BEs16<int16_t> beint16_t;
00729 typedef BEu16<uint16_t> beuint16_t;
00730 typedef BE32<uint32_t> beuint32_t;
00731 typedef BE32<uint32_t> beint32_t;
00732
00733 typedef BEs16<int16_t> *pbeint16_t;
00734 typedef BEu16<uint16_t> *pbeuint16_t;
00735 typedef BE32<uint32_t> *pbeuint32_t;
00736 typedef BE32<uint32_t> *pbeint32_t;
00737
00738 typedef BEs16<volatile int16_t> vbeint16_t;
00739 typedef BEu16<volatile uint16_t> vbeuint16_t;
00740 typedef BE32<volatile uint32_t> vbeuint32_t;
00741 typedef BE32<volatile uint32_t> vbeint32_t;
00742
00743 typedef BEs16<volatile int16_t> *pvbeint16_t;
00744 typedef BEu16<volatile uint16_t> *pvbeuint16_t;
00745 typedef BE32<volatile uint32_t> *pvbeuint32_t;
00746 typedef BE32<volatile uint32_t> *pvbeint32_t;
00747
00748 typedef BE_Float befloat;
00749 typedef BE_Double bedouble;
00750 typedef volatile BE_Float vbefloat;
00751 typedef volatile BE_Double vbedouble;
00752 typedef BE_Float *pbefloat;
00753 typedef BE_Double *pbedouble;
00754 typedef volatile BE_Float *pvbefloat;
00755 typedef volatile BE_Double *pvbedouble;
00756
00757 typedef int16_t leint16_t;
00758 typedef uint16_t leuint16_t;
00759 typedef uint32_t leuint32_t;
00760 typedef int32_t leint32_t;
00761
00762 typedef int16_t leint16_t;
00763 typedef uint16_t leuint16_t;
00764 typedef uint32_t leuint32_t;
00765 typedef int32_t leint32_t;
00766
00767 typedef pint16_t pleint16_t;
00768 typedef puint16_t pleuint16_t;
00769 typedef puint32_t pleuint32_t;
00770 typedef pint32_t pleint32_t;
00771
00772 typedef vint16_t vleint16_t;
00773 typedef vuint16_t vleuint16_t;
00774 typedef vuint32_t vleuint32_t;
00775 typedef vint32_t vleint32_t;
00776
00777 typedef pvint16_t pvleint16_t;
00778 typedef pvuint16_t pvleuint16_t;
00779 typedef pvuint32_t pvleuint32_t;
00780 typedef pvint32_t pvleint32_t;
00781
00782 typedef float lefloat;
00783 typedef double ledouble;
00784 typedef volatile float vlefloat;
00785 typedef volatile double vledouble;
00786 typedef float *plefloat;
00787 typedef double *pledouble;
00788 typedef volatile float *pvlefloat;
00789 typedef volatile double *pvledouble;
00790
```

```
00791 // inline bool operator==(beuint16_t const& lhs, beuint16_t const& rhs)
00792 //     { return lhs.val == rhs.val; }
00793 // inline bool operator!=(beuint16_t const& lhs, beuint16_t const& rhs)
00794 //     { return lhs.val != rhs.val; }
00795 // inline bool operator>(beuint16_t const& lhs, beuint16_t const& rhs)
00796 //     { return __REV16(lhs.val) > __REV16(rhs.val); }
00797
00798 // inline bool operator==(beuint16_t const& lhs, beuint16_t const& rhs)
00799 //     { return lhs.val == rhs.val; }
00800 // inline bool operator!=(beuint16_t const& lhs, beuint16_t const& rhs)
00801 //     { return lhs.val != rhs.val; }
00802 // inline bool operator>(beuint16_t const& lhs, beuint16_t const& rhs)
00803 //     { return __REV16(lhs.val) > __REV16(rhs.val); }
00804 // inline bool operator<(beuint16_t const& lhs, beuint16_t const& rhs)
00805 //     { return __REV16(lhs.val) < __REV16(rhs.val); }
00806
00807 typedef volatile bool vbool_t;
00808 typedef bool *pbool_t;
00809 typedef vbool_t *pvbool_t;
00810 #endif
00811
00812 /*
00813  * Convert values between host and network byte order
00814  */
00815 //#include <core_cmInstr.h>
00816 #define HTONS(x) __REV16(x)    // HTOBES(x)
00817 #define HTONL(x) __REV(x)      // HTOBEL(x)
00818 #define NTOHL(x) __REV(x)      // HTOBEL(x)
00819 #define NTOHS(x) __REV16(x)    // HTOBES(x)
00820
00821 #endif /* _BASICTYPES_H_ */
```

24.68 bit_overlay.h

```
00001 #ifndef __BITS_H
00002 #define __BITS_H
00003 /*****************************************************************************/
00035 #include <basictypes.h>
00036
00037 #define _STR(x) #x
00038
00039 void ser_putstring(const char *);
00040 void ser_putword(uint32_t);
00041 void ser_putbyte(uint8_t);
00042 template<uint8_t e, uint8_t s, uint8_t w>
00043 class bit_overlay
00044 {
00045     static_assert(e >= s, "Ending bit must be after starting bit");
00046     static_assert(e < w, "Ending bit must be less than mapped width");
00047
00048   public:
00049     inline uint32_t asVal()
00050     {
00051 //        ser_putstring("\r\n*this: 0x");
00052 //        ser_putword(*(beuint32_t*)this);
00053 //        ser_putstring("\r\n*this » ");
00054 //        ser_putbyte(s);
00055 //        ser_putstring(": 0x");
00056 //        ser_putword((*((beuint32_t *)this) » s));
00057 //        ser_putstring("\r\nmask: 0x");
00058 //        ser_putword(((1UL « (e-s+1))-1));
00059 //        ser_putstring("\r\n(*this » ");
00060 //        ser_putbyte(s);
00061 //        ser_putstring(") & mask: 0x");
00062 //        ser_putword((*((beuint32_t *)this) » s) & ((1UL « (e-s+1))-1));
00063 #pragma GCC diagnostic push
00064 #pragma GCC diagnostic ignored "-Wshift-count-overflow"
00065         return (*((uint32_t *)this) » s) & ((1UL « (e - s + 1)) - 1);
00066 #pragma GCC diagnostic pop
00067     }
00068     inline operator uint32_t() { return asVal(); }
00069     inline uint32_t operator=(uint32_t rhs)
00070     {
00071 #pragma GCC diagnostic push
00072 #pragma GCC diagnostic ignored "-Wshift-count-overflow"
00073         register uint32_t mask = ((1UL « (e - s + 1)) - 1);
00074 #pragma GCC diagnostic pop
00075         *((uint32_t *)this) &= ~(mask « s);
00076         *((uint32_t *)this) |= (rhs & mask) « s;
00077         return rhs;
00078     }
00079     inline bool operator==(uint32_t rhs) { return asVal() == rhs; }
00080     inline bool operator>=(uint32_t rhs) { return asVal() >= rhs; }
00081     inline bool operator<(uint32_t rhs) { return asVal() < rhs; }
00082     inline uint32_t operator«(uint32_t rhs) { return asVal() « rhs; }
```

```
00083     inline uint32_t operator*(uint32_t rhs) { return asVal() * rhs; }
00084
00085 } __attribute__((packed));
00086
00087 template<uint8_t e, uint8_t s, uint8_t w>
00088 inline uint32_t operator*(bit_overlay<e, s, w> lhs, uint32_t rhs)
00089 {
00090     return lhs.asVal() * rhs;
00091 }
00092
00093 // Specialized template for 8 bit fields
00094 template<uint8_t e, uint8_t s>
00095 class bit_overlay<e, s, 8>
00096 {
00097     static_assert(e >= s, "Ending bit must be after starting bit");
00098     static_assert(e < 8, "Ending bit must be less than mapped width");
00099
00100    public:
00101     inline uint32_t asVal() { return (*((uint8_t *)this) » s) & ((1UL « (e - s + 1)) - 1); }
00102     inline operator uint8_t() { return asVal(); }
00103     inline uint8_t operator=(uint8_t rhs)
00104     {
00105         register uint8_t mask = ((1UL « (e - s + 1)) - 1);
00106         *((uint8_t *)this) &= ~(mask « s);
00107         *((uint8_t *)this) |= (rhs & mask) « s;
00108         return rhs;
00109     }
00110     inline bool operator==(uint32_t rhs) { return asVal() == rhs; }
00111     inline bool operator>=(uint32_t rhs) { return asVal() >= rhs; }
00112     inline bool operator<(uint32_t rhs) { return asVal() < rhs; }
00113     inline uint8_t operator«(uint32_t rhs) { return asVal() « rhs; }
00114     inline uint8_t operator*(uint32_t rhs) { return asVal() * rhs; }
00115 } __attribute__((packed));
00116
00117 // Specialized template for 16 bit fields
00118 template<uint8_t e, uint8_t s>
00119 class bit_overlay<e, s, 16>
00120 {
00121     static_assert(e >= s, "Ending bit must be after starting bit");
00122     static_assert(e < 16, "Ending bit must be less than mapped width");
00123
00124    public:
00125     inline uint16_t asVal() { return (*((uint16_t *)this) » s) & ((1UL « (e - s + 1)) - 1); }
00126     inline operator uint16_t() { return asVal(); }
00127     inline uint16_t operator=(uint16_t rhs)
00128     {
00129         register uint16_t mask = ((1UL « (e - s + 1)) - 1);
00130         *((uint16_t *)this) &= ~(mask « s);
00131         *((uint16_t *)this) |= (rhs & mask) « s;
00132         return rhs;
00133     }
00134     inline bool operator==(uint32_t rhs) { return asVal() == rhs; }
00135     inline bool operator>=(uint32_t rhs) { return asVal() >= rhs; }
00136     inline bool operator<(uint32_t rhs) { return asVal() < rhs; }
00137     inline uint16_t operator«(uint32_t rhs) { return asVal() « rhs; }
00138     inline uint16_t operator*(uint32_t rhs) { return asVal() * rhs; }
00139 } __attribute__((packed));
00140
00141 #include <type_traits>
00142 static_assert(std::is_pod<bit_overlay<31, 0, 32»::value, "Data Structure requires constructible
      elements");
00143
00144 #endif /* ----- #ifndef __BITS_H  ----- */
```

24.69 coldfire/include/debugtraps.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef DBUGTRAP_H
00005 #define DBUGTRAP_H
00006
00007 void SetAddressWrittenTrap(uint32_t addr);
00008 void SetAddressWriteRangeTrap(uint32_t startaddr, uint32_t endaddr);
00009 void SetAddressReadTrap(uint32_t addr);
00010 void SetAddressReadRangeTrap(uint32_t startaddr, uint32_t endaddr);
00011
00012 /* Call this macro to monitor writes to a single variable */
00013 /* You may only monitor ONE VARIABLE                      */
00014 /* You may not write to this variable after making this call unless */
00015 /* You first use the BEFORE_WRITING_MONITORED_VAR macro */
00016 #define MONITOR_VAR_WRITES(x) SetAddressWrittenTrap((uint32_t)&x);
00017
00018 /* use this before writing to a monitored var */
00019 #define BEFORE_WRITING_MONITORED_VAR(x) \
```

```
00020     asm(" move.w #0x2700,%sr ");        \
00021     SetAddressWrittenTrap(0);
00022
00023 /* Use this after writing to a monitored var */
00024 #define AFTER_WRITING_MONITORED_VAR(x)   \
00025     asm(" nop");                         \
00026     SetAddressWrittenTrap((uint32_t)&x); \
00027     asm(" move.w (OSISRLevel),%d0 ");    \
00028     asm(" move.w %d0,%sr ");
00029
00030 #endif
```

24.70 cortex-m7/include/debugtraps.h

```
00001 #ifndef DBUGTRAP_H
00002 #define DBUGTRAP_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <stdint.h>
00007
00008 enum db_trap_acc_t
00009 {
00010     DB_TRP_ACC_RD = 0x5,
00011     DB_TRP_ACC_WR = 0x6,
00012     DB_TRP_ACC_RW = 0x7,
00013 };
00014
00015 enum db_trap_mem_width_t
00016 {
00017     DB_TRP_WIDTH_8 = 0,
00018     DB_TRP_WIDTH_16 = 1,
00019     DB_TRP_WIDTH_32 = 2,
00020 };
00021
00022 void SetValueWrittenTrap(volatile uint32_t value);
00023 void SetAddressWrittenTrap(uint32_t addr);
00024 void SetAddressWrittenTrap(uint32_t addr, uint32_t value, db_trap_mem_width_t width);
00025 void SetAddressWriteRangeTrap(uint32_t startaddr, uint32_t endaddr, int debugModule = 1);
00026 void SetAddressReadTrap(uint32_t addr);
00027 void SetAddressReadRangeTrap(uint32_t startaddr, uint32_t endaddr);
00028
00029 /* Call this macro to monitor writes to a single variable */
00030 /* You may only monitor ONE VARIABLE                      */
00031 /* You may not write to this variable after making this call unless */
00032 /* You first use the BEFORE_WRITING_MONITORED_VAR macro */
00033 #define MONITOR_VAR_WRITES(x) SetAddressWrittenTrap((uint32_t)&x);
00034
00035 /* use this before writing to a monitored var */
00036 #define BEFORE_WRITING_MONITORED_VAR(x) \
00037     asm("cpsid i");                     \
00038     SetAddressWrittenTrap(0);
00039
00040 /* Use this after writing to a monitored var */
00041 #define AFTER_WRITING_MONITORED_VAR(x)   \
00042     asm(" dsb");                         \
00043     SetAddressWrittenTrap((uint32_t)&x); \
00044     asm(" cpsie i");
00045
00046 #endif
```

24.71 exidx_unwind.h

```
00001 #ifndef __EXIDX_UNWIND_H
00002 #define __EXIDX_UNWIND_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 #include <predef.h>
00008 #include <stdint.h>
00009 #include <stddef.h>
00010
00011 #define COMP_ENTRY_IDX_MASK (0xF « 24)
00012 #define ARM_EXIDX_CMD_FINISH 0xB0
00013 #define ARM_EXIDX_CMD_REFUSED 0x80
00014 #define ARM_EXIDX_CMD_REG_POP 0xB1
00015 #define ARM_EXIDX_CMD_DATA_POP 0xB2
00016 #define ARM_EXIDX_CMD_VFP_POP 0xB3
00017 #define ARM_EXIDX_CMD_WCGR_POP 0xC7
00018 #define ARM_EXIDX_PRS_CANT_UNWIND 0x00000001
00019
```

```
00020 typedef enum
00021 {
00022     ARM_RET_HND_FP = 0xFFFFFFE1,
00023     ARM_RET_MSP_FP = 0xFFFFFFE9,
00024     ARM_RET_PSP_FP = 0xFFFFFFED,
00025     ARM_RET_HND_NOFP = 0xFFFFFFF1,
00026     ARM_RET_MSP_NOFP = 0xFFFFFFF9,
00027     ARM_RET_PSP_NOFP = 0xFFFFFFFD,
00028 } ExceptRet_LR_t;
00029
00030 struct insn_t
00031 {
00032     const uint32_t *prs32;   // pointer to table entry holding the instructions
00033     int rem;                 // Remaining instruction count to unwind frame
00034     int idx;                 // index of next instruction in entry word
00035
00036     int getNext();
00037     int peek();
00038     uint32_t get_uleb128();
00039     void init(const uint32_t *tab_entry);
00040 };
00041
00042 /*
00043  * To convert the index table entry into an exception address, we must first
00044  *  sign extend the offset value (since only 31 bits are stored).
00045  *  We then add this offset to the address of the entry to get the handling
00046  *  table entry. I think...
00047  */
00048 inline uint32_t prel31Addr(const uint32_t *prel31)
00049 {
00050     int32_t s_ext_offset = (*((int32_t *)prel31) « 1) » 1;
00051     return (((uint32_t)prel31) + s_ext_offset) & 0x7FffFFff;
00052 }
00053
00054 struct exidx_t
00055 {
00056     uint32_t offset;
00057     uint32_t xtab_insn;
00058
00059     inline uint32_t asAddr() const { return prel31Addr(&offset); }
00060     const int getInsn(insn_t *insn) const;
00061 };
00062
00063 extern const exidx_t __exidx_start[];
00064 extern const exidx_t __exidx_end[];
00065
00066 // Virtual Register set for stack unwinding
00067 struct vrs_t
00068 {
00069     union
00070     {
00071         uint32_t r[16];
00072         struct
00073         {
00074             uint32_t _r[12];
00075             uint32_t ip;
00076             uint32_t *sp;
00077             uint32_t lr;
00078             uint32_t pc;
00079         };
00080     };
00081     uint32_t xpsr;
00082     uint64_t fpr[16];
00083     uint32_t fpscr;
00084     bool bVPSP;   // is our virtual stack pointer currently the Process Stack Pointer
00085 };
00086
00087 typedef void *unwind_trace_dat_t;
00088 typedef int (*unwind_trace_fn)(const vrs_t &vrs, unwind_trace_dat_t trace_dat);
00089
00090 struct unwind_ctx
00091 {
00092     vrs_t vrs;
00093     insn_t insn;
00094     uint32_t maxRecurse;
00095     uint32_t flags;
00096
00097     int step();
00098     int unwind_frame();
00099     inline void init() volatile
00100     {
00101         // Do not modify without verifying dissassembly...
00102         uint32_t offset;
00103         offset = offsetof(unwind_ctx, vrs) + offsetof(vrs_t, r) + 4 * sizeof(vrs.r[0]);
00104         asm volatile(
00105             "add %%r1,  %0, %1\n\t"
00106             "str %%r4,  [%%r1], #4\n\t"
```

```
00107             "str %%r5,  [%%r1], #4\n\t"
00108             "str %%r6,  [%%r1], #4\n\t"
00109             "str %%r7,  [%%r1], #4\n\t"
00110             "str %%r8,  [%%r1], #4\n\t"
00111             "str %%r9,  [%%r1], #4\n\t"
00112             "str %%r10, [%%r1], #4\n\t"
00113             "str %%r11, [%%r1], #4\n\t"
00114             :
00115             : "r"(this), "r"(offset)
00116             : "r1", "r4", "r5", "r6", "r7", "r8", "r9", "r10", "r11", "memory");
00117     }
00118     inline void initFpr() volatile
00119     {
00120 #if ((__SOFTFP__ != 1) || (defined __ARM_FP))
00121         // Do not modify without verifying dissassembly...
00122         uint32_t offset;
00123         offset = offsetof(unwind_ctx, vrs) + offsetof(vrs_t, fpr);
00124         asm volatile(
00125             "add %%r1,  %0, %1\n\t"
00126             "vstm.64 %%r1!, {%%d0-%%d15}\n\t"
00127             "vmrs    %%r2, FPSCR\n\t"
00128             "str     %%r2, [%%r1]\n\t"
00129             :
00130             : "r"(this), "r"(offset)
00131             : "r1", "r2", "memory");
00132 #endif
00133     }
00134     inline void initForRunning() volatile
00135     {
00136         // Do not modify without verifying dissassembly...
00137         uint32_t offset;
00138         offset = offsetof(unwind_ctx, vrs) + offsetof(vrs_t, r) + 2 * sizeof(vrs.r[0]);
00139         asm volatile(
00140             "add %%r1,  %0, %1\n\t"
00141             "stm r1!, {r2-r12}\n\t"
00142             "str sp, [r1], #4\n\t"
00143             "str lr, [r1], #4\n\t"
00144             :
00145             : "r"(this), "r"(offset)
00146             : "r1", "memory");
00147     }
00148     void init(volatile cpu_tcb *ptcb);
00149     void init(uint32_t sp, uint32_t lr, uint32_t pc);
00150     void unwind_eframe();
00151     int unwind_stack(unwind_trace_fn trace_cb, unwind_trace_dat_t trace_dat);
00152 };
00153
00154 // Exception Frame for no FPU or FPU disabled.
00155 typedef struct
00156 {
00157     union
00158     {
00159         struct
00160         {
00161             uint32_t r0;
00162             uint32_t r1;
00163             uint32_t r2;
00164             uint32_t r3;
00165             uint32_t r12;
00166             uint32_t lr;
00167             uint32_t pc;
00168             uint32_t xprs;
00169         };
00170         uint32_t r[16];
00171     };
00172 } eframe;
00173
00174 // Exception Frame with FPU
00175 typedef struct
00176 {
00177     union
00178     {
00179         struct
00180         {
00181             uint32_t r0;
00182             uint32_t r1;
00183             uint32_t r2;
00184             uint32_t r3;
00185             uint32_t r12;
00186             uint32_t lr;
00187             uint32_t pc;
00188             uint32_t xprs;
00189         };
00190         uint32_t r[16];
00191     };
00192     union
00193     {
```

```
00194         uint32_t fp[32];
00195         uint64_t dfp[16];
00196     };
00197     uint32_t fpscr;
00198 } eframe_fp;
00199
00200 // Minimal state context for stack unwind
00201 typedef struct
00202 {
00203     uint32_t pc;
00204     uint32_t lr;
00205     uint32_t sp;
00206 } uwctx;
00207
00208 // extab exception table entry
00209 typedef struct
00210 {
00211     uint32_t prs_fn_offset;
00212     uint32_t prs_data;
00213 } extab_entry_gen;
00214
00215 struct extab_insn_tbl
00216 {
00217     inline uint8_t operator[](int i);
00218 };
00219
00220 struct extab_insn_ctx
00221 {
00222 };
00223
00224 struct extab_entry_comp
00225 {
00226     uint32_t idx_prs;
00227     uint32_t prs_data[];
00228     uint8_t operator[](int i);
00229 };
00230
00231 exidx_t *find_insn(uint32_t *start, uint32_t *end, uint32_t pc);
00232
00233 #endif /* ----- #ifndef __EXIDX_UNWIND_H  ----- */
```

24.72 mpu_armv7.h

```
00001 /**************************************************************************
00002  * @file     mpu_armv7.h
00003  * @brief    CMSIS MPU API for Armv7-M MPU
00004  * @version  V5.1.0
00005  * @date     08. March 2019
00006  **************************************************************************/
00007 /*
00008  * Copyright (c) 2017-2019 Arm Limited. All rights reserved.
00009  *
00010  * SPDX-License-Identifier: Apache-2.0
00011  *
00012  * Licensed under the Apache License, Version 2.0 (the License); you may
00013  * not use this file except in compliance with the License.
00014  * You may obtain a copy of the License at
00015  *
00016  * www.apache.org/licenses/LICENSE-2.0
00017  *
00018  * Unless required by applicable law or agreed to in writing, software
00019  * distributed under the License is distributed on an AS IS BASIS, WITHOUT
00020  * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00021  * See the License for the specific language governing permissions and
00022  * limitations under the License.
00023  */
00024
00025 #if defined(__ICCARM__)
00026 #pragma system_include /* treat file as system include file for MISRA check */
00027 #elif defined(__clang__)
00028 #pragma clang system_header /* treat file as system include file */
00029 #endif
00030
00031 #ifndef ARM_MPU_ARMV7_H
00032 #define ARM_MPU_ARMV7_H
00033
00034 #define ARM_MPU_REGION_SIZE_32B ((uint8_t)0x04U)
00035 #define ARM_MPU_REGION_SIZE_64B ((uint8_t)0x05U)
00036 #define ARM_MPU_REGION_SIZE_128B ((uint8_t)0x06U)
00037 #define ARM_MPU_REGION_SIZE_256B ((uint8_t)0x07U)
00038 #define ARM_MPU_REGION_SIZE_512B ((uint8_t)0x08U)
00039 #define ARM_MPU_REGION_SIZE_1KB ((uint8_t)0x09U)
00040 #define ARM_MPU_REGION_SIZE_2KB ((uint8_t)0x0AU)
00041 #define ARM_MPU_REGION_SIZE_4KB ((uint8_t)0x0BU)
00042 #define ARM_MPU_REGION_SIZE_8KB ((uint8_t)0x0CU)
```

```
00043 #define ARM_MPU_REGION_SIZE_16KB ((uint8_t)0x0DU)
00044 #define ARM_MPU_REGION_SIZE_32KB ((uint8_t)0x0EU)
00045 #define ARM_MPU_REGION_SIZE_64KB ((uint8_t)0x0FU)
00046 #define ARM_MPU_REGION_SIZE_128KB ((uint8_t)0x10U)
00047 #define ARM_MPU_REGION_SIZE_256KB ((uint8_t)0x11U)
00048 #define ARM_MPU_REGION_SIZE_512KB ((uint8_t)0x12U)
00049 #define ARM_MPU_REGION_SIZE_1MB ((uint8_t)0x13U)
00050 #define ARM_MPU_REGION_SIZE_2MB ((uint8_t)0x14U)
00051 #define ARM_MPU_REGION_SIZE_4MB ((uint8_t)0x15U)
00052 #define ARM_MPU_REGION_SIZE_8MB ((uint8_t)0x16U)
00053 #define ARM_MPU_REGION_SIZE_16MB ((uint8_t)0x17U)
00054 #define ARM_MPU_REGION_SIZE_32MB ((uint8_t)0x18U)
00055 #define ARM_MPU_REGION_SIZE_64MB ((uint8_t)0x19U)
00056 #define ARM_MPU_REGION_SIZE_128MB ((uint8_t)0x1AU)
00057 #define ARM_MPU_REGION_SIZE_256MB ((uint8_t)0x1BU)
00058 #define ARM_MPU_REGION_SIZE_512MB ((uint8_t)0x1CU)
00059 #define ARM_MPU_REGION_SIZE_1GB ((uint8_t)0x1DU)
00060 #define ARM_MPU_REGION_SIZE_2GB ((uint8_t)0x1EU)
00061 #define ARM_MPU_REGION_SIZE_4GB ((uint8_t)0x1FU)
00062
00063 #define ARM_MPU_AP_NONE 0U
00064 #define ARM_MPU_AP_PRIV 1U
00065 #define ARM_MPU_AP_URO 2U
00066 #define ARM_MPU_AP_FULL 3U
00067 #define ARM_MPU_AP_PRO 5U
00068 #define ARM_MPU_AP_RO 6U
00069
00075 #define ARM_MPU_RBAR(Region, BaseAddress) (((BaseAddress)&MPU_RBAR_ADDR_Msk) |
      ((Region)&MPU_RBAR_REGION_Msk) | (MPU_RBAR_VALID_Msk))
00076
00086 #define ARM_MPU_ACCESS_(TypeExtField, IsShareable, IsCacheable, IsBufferable)
      \
00087     ((((TypeExtField) << MPU_RASR_TEX_Pos) & MPU_RASR_TEX_Msk) | (((IsShareable) << MPU_RASR_S_Pos) &
      MPU_RASR_S_Msk) | \
00088      (((IsCacheable) << MPU_RASR_C_Pos) & MPU_RASR_C_Msk) | (((IsBufferable) << MPU_RASR_B_Pos) &
      MPU_RASR_B_Msk))
00089
00099 #define ARM_MPU_RASR_EX(DisableExec, AccessPermission, AccessAttributes, SubRegionDisable, Size)
      \
00100     ((((DisableExec) << MPU_RASR_XN_Pos) & MPU_RASR_XN_Msk) | (((AccessPermission) << MPU_RASR_AP_Pos) &
      MPU_RASR_AP_Msk) | \
00101      (((AccessAttributes) & (MPU_RASR_TEX_Msk | MPU_RASR_S_Msk | MPU_RASR_C_Msk | MPU_RASR_B_Msk))) |
      \
00102      (((SubRegionDisable) << MPU_RASR_SRD_Pos) & MPU_RASR_SRD_Msk) | (((Size) << MPU_RASR_SIZE_Pos) &
      MPU_RASR_SIZE_Msk) |  \
00103      (((MPU_RASR_ENABLE_Msk))))
00104
00117 #define ARM_MPU_RASR(DisableExec, AccessPermission, TypeExtField, IsShareable, IsCacheable,
      IsBufferable, SubRegionDisable, Size) \
00118     ARM_MPU_RASR_EX(DisableExec, AccessPermission, ARM_MPU_ACCESS_(TypeExtField, IsShareable,
      IsCacheable, IsBufferable),         \
00119                     SubRegionDisable, Size)
00120
00128 #define ARM_MPU_ACCESS_ORDERED ARM_MPU_ACCESS_(0U, 1U, 0U, 0U)
00129
00139 #define ARM_MPU_ACCESS_DEVICE(IsShareable) ((IsShareable) ? ARM_MPU_ACCESS_(0U, 1U, 0U, 1U) :
      ARM_MPU_ACCESS_(2U, 0U, 0U, 0U))
00140
00152 #define ARM_MPU_ACCESS_NORMAL(OuterCp, InnerCp, IsShareable) ARM_MPU_ACCESS_((4U | (OuterCp)),
      IsShareable, ((InnerCp)&2U), ((InnerCp)&1U))
00153
00157 #define ARM_MPU_CACHEP_NOCACHE 0U
00158
00162 #define ARM_MPU_CACHEP_WB_WRA 1U
00163
00167 #define ARM_MPU_CACHEP_WT_NWA 2U
00168
00172 #define ARM_MPU_CACHEP_WB_NWA 3U
00173
00177 typedef struct
00178 {
00179     uint32_t RBAR;
00180     uint32_t RASR;
00181 } ARM_MPU_Region_t;
00182
00186 __STATIC_INLINE void ARM_MPU_Enable(uint32_t MPU_Control)
00187 {
00188     MPU->CTRL = MPU_Control | MPU_CTRL_ENABLE_Msk;
00189 #ifdef SCB_SHCSR_MEMFAULTENA_Msk
00190     SCB->SHCSR |= SCB_SHCSR_MEMFAULTENA_Msk;
00191 #endif
00192     __DSB();
00193     __ISB();
00194 }
00195
00198 __STATIC_INLINE void ARM_MPU_Disable(void)
00199 {
```

```
00200     __DMB();
00201 #ifdef SCB_SHCSR_MEMFAULTENA_Msk
00202     SCB->SHCSR &= ~SCB_SHCSR_MEMFAULTENA_Msk;
00203 #endif
00204     MPU->CTRL &= ~MPU_CTRL_ENABLE_Msk;
00205 }
00206
00210 __STATIC_INLINE void ARM_MPU_ClrRegion(uint32_t rnr)
00211 {
00212     MPU->RNR = rnr;
00213     MPU->RASR = 0U;
00214 }
00215
00220 __STATIC_INLINE void ARM_MPU_SetRegion(uint32_t rbar, uint32_t rasr)
00221 {
00222     MPU->RBAR = rbar;
00223     MPU->RASR = rasr;
00224 }
00225
00231 __STATIC_INLINE void ARM_MPU_SetRegionEx(uint32_t rnr, uint32_t rbar, uint32_t rasr)
00232 {
00233     MPU->RNR = rnr;
00234     MPU->RBAR = rbar;
00235     MPU->RASR = rasr;
00236 }
00237
00243 __STATIC_INLINE void ARM_MPU_OrderedMemcpy(volatile uint32_t *dst, const uint32_t *__RESTRICT src,
    uint32_t len)
00244 {
00245     uint32_t i;
00246     for (i = 0U; i < len; ++i)
00247     {
00248         dst[i] = src[i];
00249     }
00250 }
00251
00256 __STATIC_INLINE void ARM_MPU_Load(ARM_MPU_Region_t const *table, uint32_t cnt)
00257 {
00258     const uint32_t rowWordSize = sizeof(ARM_MPU_Region_t) / 4U;
00259     while (cnt > MPU_TYPE_RALIASES)
00260     {
00261         ARM_MPU_OrderedMemcpy(&(MPU->RBAR), &(table->RBAR), MPU_TYPE_RALIASES * rowWordSize);
00262         table += MPU_TYPE_RALIASES;
00263         cnt -= MPU_TYPE_RALIASES;
00264     }
00265     ARM_MPU_OrderedMemcpy(&(MPU->RBAR), &(table->RBAR), cnt * rowWordSize);
00266 }
00267
00268 #endif
```

24.73 nbrtos_cm7.h

```
00001 #ifndef __NBRTOS_CM7_H
00002 #define __NBRTOS_CM7_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 // Macros expectdy by core CM functions
00008 #ifndef __STATIC_INLINE
00009 #define __STATIC_INLINE static inline
00010 #endif
00011 #ifndef __INLINE
00012 #define __INLINE inline
00013 #endif
00014 #ifndef __ASM
00015 #define __ASM __asm
00016 #endif
00017 #include <core_cm7.h>
00018
00019 #define SCB_ICSR_REG (*((uint32_t *)0xE000ED04UL))
00020 #define SCB_ICSR_PENDSVSET_BIT (0x10000000);
00021
00022 extern volatile uint32_t critical_count;
00023
00024 #define USER_ENTER_CRITICAL() \
00025     {                         \
00026         asm("cpsid i");       \
00027         critical_count++;     \
00028     }
00029
00030 #define USER_EXIT_CRITICAL()                              \
00031     {                                                     \
00032         if (--critical_count <= 0) { asm("cpsie i"); } \
00033     }
```

```
00034
00035 //#define OS_TASK_SW() { SCB_ICSR_REG = SCB_ICSR_PENDSVSET_BIT; }
00036 //#define OSIntCtxSw() { SCB_ICSR_REG = SCB_ICSR_PENDSVSET_BIT; }
00037
00038 extern "C"
00039 {
00040     void UCosSetup();
00041 }
00042 void UCosBegin();
00043
00044 #endif /* ----- #ifndef __NBRTOS_CM7_H  ----- */
```

24.74 coldfire/include/nbrtoscpu.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NBRTOS_CPU_COLDFIRE
00006 #define NB_NBRTOS_CPU_H
00007 #define NBRTOS_CPU_COLDFIRE 1
00008 /**************************************************************************/
00018 /**************************************************************************/
00019
00020 /**************************************************************************/
00023 extern volatile unsigned long critical_count;
00024 extern "C"
00025 {
00026     void NBRtosSetup();
00027     void OSIntCtxSw(void);
00028 }
00029 void NBRtosBegin();
00030
00031 #include <constants.h>
00032
00033 /**************************************************************************/
00038 #define NBRTOS_ENTER_CRITICAL()                \
00039     {                                          \
00040         asm volatile(" nop");                  \
00041         asm volatile(" move.w #0x2700,%sr "); \
00042     }
00043
00044 #define NBRTOS_EXIT_CRITICAL()
    \
00045     {
    \
00046         asm volatile(".extern critical_count");
    \
00047         asm volatile(".extern OSISRLevel32");
    \
00048         asm volatile(" nop");
    \
00049         if (critical_count == 0)
    \
00050         {
    \
00051             asm volatile(" move.l OSISRLevel32,%%d1 \n\t" : /* out */ : /* input */ : "%%d1"
    /*modified */); \
00052             asm volatile(" move.w %d1, %sr");
    \
00053         }
    \
00054     }
00055
00068 #define USER_ENTER_CRITICAL()                        \
00069     {                                                \
00070         asm volatile(".extern critical_count");     \
00071         asm volatile(" nop");                        \
00072         asm volatile(" move.w #0x2700,%sr ");        \
00073         asm volatile(" addq.l #1,critical_count "); \
00074     }
00075
00084 #define USER_EXIT_CRITICAL()
    \
00085     {
    \
00086         asm volatile(".extern critical_count");
    \
00087         asm volatile(".extern OSISRLevel32");
    \
00088         asm volatile(" nop");
    \
00089         asm volatile(" subq.l #1,critical_count ");
    \
```

```
00090         if (critical_count == 0)
      \
00091         {
      \
00092             asm volatile(" move.l OSISRLevel32,%%d1 \n\t" : /* out */ : /* input */ : "%%d1"
      /*Modified */); \
00093             asm volatile(" move.w %d1, %sr");
      \
00094         }
      \
00095     }
00096
00097 #define RAW_OS_TASK_SW() asm volatile(" trap #14 ");
00098
00099 #define OS_TASK_SW()                \
00100     asm volatile(" trap #14 ");     \
00101     asm(".global  RTOSWAITS_HERE"); \
00102     asm("RTOSWAITS_HERE:");
00103
00104 #define OS_IDLE_LOOP() asm volatile(" stop #0x2000 ");
00105
00106 #define MASK_ALL_ISR() asm volatile("move.w #0x2700,%sr ")
00107
00108 #define NOP() asm volatile(" nop")
00109 #define CACHE_FLUSH() asm volatile(" nop")
00110
00111 #define FORCE_TRAP() asm volatile("illegal")
00112
00113 /*
00114 *********************************************************
00115 *         NBRTOS CPU Specific TASK CONTROL BLOCK CLASS
00116 *********************************************************
00117 */
00118
00119 class cpu_tcb
00120 {
00121    public:
00122     void *OSTCBStkPtr;
00123     long *OSTCBStkBot;
00124     long *OSTCBStkTop;
00125 };
00126
00127 #endif
```

24.75 cortex-m7/include/nbrtoscpu.h

```
00001 #ifndef __NBRTOS_CM7_H
00002 #define NB_NBRTOS_CPU_H
00003 #define __NBRTOS_CM7_H
00004
00005 /*NB_REVISION*/
00006
00007 /*NB_COPYRIGHT*/
00008 #include <basictypes.h>
00009
00010 // Macros expectdy by core CM functions
00011 #ifndef __STATIC_INLINE
00012 #define __STATIC_INLINE static inline
00013 #endif
00014 #ifndef __INLINE
00015 #define __INLINE inline
00016 #endif
00017 #ifndef __ASM
00018 #define __ASM __asm
00019 #endif
00020 //#include <core_cmFunc.h>
00021
00022 #define SCB_ICSR_REG (*((uint32_t *)0xE000ED04UL))
00023 #define SCB_ICSR_PENDSVSET_BIT (0x10000000);
00024 static __inline__ void *get_pc(void)
00025 {
00026     void *pc;
00027     asm("mov %0, pc" : "=r"(pc));
00028     return pc;
00029 }
00030 /*#define NBRTOS_ENTER_CRITICAL() { register uint32_t pc, tmp_12=0; \
00031         lockPC = (uint32_t)get_pc(); \
00032     asm volatile ("add %2, pc, #4\n\t" \
00033         "str %2, [%1]\n\t" \
00034         "cpsid i" : "=r"(pc) : "r"(&lockPC), "r"(tmp_12)); }\
00035     asm volatile ("cpsid i"); }*/
00036
00037 extern volatile uint32_t critical_count;
00038 extern volatile uint32_t lockPC;
00039 extern volatile uint32_t OSISRLevel32;
```

```
00040
00041 #ifndef _DEBUG
00042 // If NOT Debug, but we do reserve some IRQ levels to not be masked by the RTOS
00043 #if defined OS_MAX_IRQ_MASK && (OS_MAX_IRQ_MASK < CPU_MAX_IRQ)
00044 #define NBRTOS_ENTER_CRITICAL()                                                    \
00045     {                                                                              \
00046         __set_BASEPRI((CPU_MAX_IRQ - OS_MAX_IRQ_MASK) << (8 - __NVIC_PRIO_BITS)); \
00047         asm volatile("dsb" ::: "memory");                                          \
00048     }
00049 #define NBRTOS_EXIT_CRITICAL()                                  \
00050     {                                                           \
00051         if (critical_count <= 0)                                \
00052         {                                                       \
00053             __set_BASEPRI(OSISRLevel32 << (8 - __NVIC_PRIO_BITS)); \
00054             asm volatile("dsb" ::: "memory");                   \
00055         }                                                       \
00056     }
00057
00058 #define USER_ENTER_CRITICAL()                                                      \
00059     {                                                                              \
00060         __set_BASEPRI((CPU_MAX_IRQ - OS_MAX_IRQ_MASK) << (8 - __NVIC_PRIO_BITS)); \
00061         asm volatile("dsb" ::: "memory");                                          \
00062         critical_count++;                                                          \
00063     }
00064
00065 #define USER_EXIT_CRITICAL()                                                       \
00066     {                                                                              \
00067         if (--critical_count <= 0)                                                 \
00068         {                                                                          \
00069             __set_BASEPRI(OSISRLevel32 << (8 - __NVIC_PRIO_BITS)); \
00070             asm volatile("dsb" ::: "memory");                                      \
00071         }                                                                          \
00072     }
00073
00074 #else /* -> #if OS_MAX_IRQ_MASK < CPU_MAX_IRQ */
00075 // If NOT Debug and no RTOS ISR blocking limitations, disable Interrupts for
00076 // RTOS Critical Sections
00077 #define NBRTOS_ENTER_CRITICAL() asm volatile("cpsid i\nisb");
00078 #define NBRTOS_EXIT_CRITICAL()                                  \
00079     {                                                           \
00080         if (critical_count <= 0) { asm volatile("cpsie i"); } \
00081     }
00082
00083 #define USER_ENTER_CRITICAL()         \
00084     {                                 \
00085         asm volatile("cpsid i\nisb"); \
00086         critical_count++;             \
00087     }
00088
00089 #define USER_EXIT_CRITICAL()                                    \
00090     {                                                           \
00091         if (--critical_count <= 0) { asm volatile("cpsie i"); } \
00092     }
00093 #endif /* #if OS_MAX_IRQ_MASK < CPU_MAX_IRQ */
00094 #else  /* #ifndef _DEBUG */
00095
00096 extern volatile uint32_t dbgState;
00097 #define DBG_STATE_INT_MASK (0x2)
00098 #define DBG_CRITMASK_RTOS (0x4)
00099 #define DBG_CRITMASK_USER (0x8)
00100 #define DBG_CRITMASK_GDB (0x10)
00101
00102 #define NBRTOS_ENTER_CRITICAL()                     \
00103     {                                               \
00104         asm volatile("cpsid i\nisb");               \
00105         __set_BASEPRI(1 << (8 - __NVIC_PRIO_BITS)); \
00106         asm volatile("cpsie i");                    \
00107         dbgState |= DBG_CRITMASK_RTOS;              \
00108     }
00109 #define NBRTOS_EXIT_CRITICAL()
    \
00110     {
    \
00111         if (critical_count <= 0)
    \
00112         {
    \
00113             dbgState &= ~DBG_CRITMASK_RTOS;
    \
00114             if (!(dbgState & (DBG_STATE_INT_MASK | DBG_CRITMASK_GDB))) { __set_BASEPRI(OSISRLevel32 <<
    (8 - __NVIC_PRIO_BITS)); } \
00115         }
    \
00116     }
00117
00118 #define USER_ENTER_CRITICAL()                       \
00119     {                                               \
```

```
00120         asm volatile("cpsid i\nisb");                \
00121         __set_BASEPRI(1 « (8 - __NVIC_PRIO_BITS)); \
00122         asm volatile("cpsie i");                     \
00123         critical_count++;                            \
00124         dbgState |= DBG_CRITMASK_USER;               \
00125     }
00126
00127 #define USER_EXIT_CRITICAL()
    \
00128     {
    \
00129         if (--critical_count <= 0)
    \
00130         {
    \
00131             dbgState &= ~DBG_CRITMASK_USER;
    \
00132             if (!(dbgState & (DBG_STATE_INT_MASK | DBG_CRITMASK_GDB))) { __set_BASEPRI(OSISRLevel32 «
    (8 - __NVIC_PRIO_BITS)); } \
00133         }
    \
00134     }
00135
00136 #endif /* #ifndef _DEBUG */
00137
00138 #define OS_TASK_SW()                                \
00139     {                                               \
00140         SCB_ICSR_REG = SCB_ICSR_PENDSVSET_BIT; \
00141         NBRTOS_EXIT_CRITICAL();                     \
00142         asm(".global  RTOSWAITS_HERE");             \
00143         asm("RTOSWAITS_HERE:");                     \
00144         NBRTOS_ENTER_CRITICAL();                    \
00145     }
00146
00147 #define OSIntCtxSw()                                \
00148     {                                               \
00149         SCB_ICSR_REG = SCB_ICSR_PENDSVSET_BIT; \
00150     }
00151
00152 #define FORCE_TRAP() asm volatile("udf")
00153
00154 extern "C"
00155 {
00156     void NBRtosSetup();
00157 }
00158 void NBRtosBegin();
00159
00160 void FlushCache_ByAddr(uint32_t *addr, uint32_t len);
00161 void InvalidateCache_ByAddr(uint32_t *addr, uint32_t len);
00162
00163 enum TaskDlyState
00164 {
00165     DLY_NO_PEND,    // Pend callback is being called in a path that
00166                     // does not pend the task
00167     DLY_PEND,       // Callback is being called just prior to
00168                     // task suspension and switch
00169     DLY_RESUME      // Callback is being called just *after* task resumption
00170 };
00171
00172 /*
00173 *********************************************************
00174 *         uCOS TASK CONTROL BLOCK DATA STRUCTURE
00175 *********************************************************
00176 */
00177 struct cpu_tcb
00178 {
00179     uint32_t r4_r11[8];   // Storage for task registers when swapping
00180     double d[16];         // Storage for FPU context, FPU is lazily restored
00181     uint32_t fpscr;
00182     void *OSTCBStkPtr;
00183
00184     long *OSTCBStkBot;
00185     long *OSTCBStkTop;
00186 };
00187
00188 //   struct OS_TASK_DLY_OBJ
00189 //   {
00190 //       volatile uint32_t OSTbl[TASK_TABLE_SIZE];
00191 //         volatile uint32_t backstop{0x80000000}; // this exists to run OSGetHighBit_Set
00192 //      public:
00193 //         OS_TASK_DLY_OBJ();
00194 //         void Init();
00195 //         void Wait(volatile OS_TCB &tcb, uint8_t StatReason, unsigned long timeout);
00196 //         void ClearWaiting(uint8_t prio);
00197 //         void Ready(uint8_t StatReason);
00198 //   };
00199
```

```
00200 #endif /* ----- #ifndef __NBRTOS_CM7_H  ----- */
```

24.76 coldfire/include/NetworkDebug.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NETWORKDEBUG_H
00006 #define NETWORKDEBUG_H
00007
00008 void InitializeNetworkGDB();
00009 void InitializeNetworkGDB_and_Wait();
00010 extern volatile bool bDebuggerInitialized;
00011
00012
00013 #define BREAKPOINT() asm("   trap #2");
00014
00015 class dbip_obj
00016 {
00017     const char *cp;
00018
00019   public:
00020     dbip_obj(const char *x);
00021 };
00022
00023 #define DebugIP(x) static dbip_obj dbip_obj1(x);
00024 #define DebugRebootOnTrap() static dbip_obj dbip_obj2("RT");
00025 #define DebugRebootOnDisconnect() static dbip_obj dbip_obj3("RD");
00026 #define DebugNormalArp() static dbip_obj dbip_obj4("A");
00027
00028 #endif /* ----- #ifndef NETWORKDEBUG_H ----- */
```

24.77 cortex-m7/include/NetworkDebug.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NETWORKDEBUG_H
00006 #define NETWORKDEBUG_H
00007
00008 void InitializeNetworkGDB();
00009 void InitializeNetworkGDB_and_Wait();
00010 extern volatile bool bDebuggerInitialized;
00011 extern volatile bool bHaltWithoutConnection;
00012
00013 #define BREAKPOINT() asm("bkpt");
00014
00015 class dbip_obj
00016 {
00017     const char *cp;
00018
00019   public:
00020     dbip_obj(const char *x);
00021 };
00022
00023 #define DebugIP(x) static dbip_obj dbip_obj1(x);
00024 #define DebugRebootOnTrap() static dbip_obj dbip_obj2("RT");
00025 #define DebugRebootOnDisconnect() static dbip_obj dbip_obj3("RD");
00026 #define DebugNormalArp() static dbip_obj dbip_obj4("A");
00027
00028 #endif /* ----- #ifndef NETWORKDEBUG_H ----- */
```

24.78 startup.h

```
00001 #ifndef __STARTUP_H
00002 #define __STARTUP_H
00003 #include <stdint.h>
00004
00005 // Flags define bits for the dwFlags in the MonExtraStruct
00006 // They are used for various image configurations on boot
00007 #define FLAG_DTCM_EN (0x1)   // Data TCM should be enabled
00008 #define FLAG_ITCM_EN (0x2)   // Instruction TCM should be enabled
00009 #define FLAG_DBG_WFE_EN (0x4)
00010 #define FLAG_RELOC (0x8)   // image is relocatable in flash (assuming header updates)
00011
00012 #define NBURN_CSUM_MAGIC (0x4255524E)
00013
```

```
00014 typedef struct
00015 {
00016     uint32_t u32Start;
00017     uint32_t u32End;
00018     uint8_t u8Region;
00019     uint8_t u8pad;
00020     uint16_t u16Attrs;
00021 } __attribute__((packed)) MPUConfig_t;
00022
00023 typedef struct
00024 {
00025     uint32_t dwFlags;
00026     uint32_t count;
00027     MPUConfig_t config[];
00028     // uint32_t checksum will follow immediately
00029 } __attribute__((packed)) MonExtraStruct;
00030
00031 typedef struct
00032 {
00033     uint32_t dwImgAddr;
00034     uint32_t dwExecutionAddr;
00035     MonExtraStruct *pExtra;
00036     uint32_t dwImgLen;
00037     uint32_t dwImgSum;
00038     uint32_t dwStructSum;
00039 } AppHeader_t;
00040
00041 #define IMG_ID_XSUM_MAGIC 0x4E42   // 'N"B'
00042
00043 // This is just an App Header with the image ID field, as it would be found at
00044 // the sector base
00045 typedef struct
00046 {
00047     uint16_t wImgID;
00048     uint16_t wImgIDXSum;
00049     uint32_t dwImgAddr;
00050     uint32_t dwExecutionAddr;
00051     MonExtraStruct *pExtra;
00052     uint32_t dwImgLen;
00053     uint32_t dwImgSum;
00054     uint32_t dwStructSum;
00055 } StartUpStruct;
00056
00057 // allows for easy mapping of StartupStructs onto block boundaries
00058 typedef struct
00059 {
00060     union
00061     {
00062         StartUpStruct startHdr;
00063         struct
00064         {
00065             uint16_t wImgID;
00066             uint16_t wImgIDXSum;
00067             AppHeader_t appHdr;
00068         };
00069     };
00070     uint8_t padd[0x2000 - sizeof(StartUpStruct)];
00071 } __attribute__((packed)) StartUpBlockStruct;
00072
00073 extern StartUpBlockStruct *RunningAppImage;
00074
00075 #endif /* ----- #ifndef __STARTUP_H  ----- */
```

24.79 htmlvars.h

```
00001 extern config_uint gMyOwnuVal;
00002 extern config_string gMyOwnStrVal;
```

24.80 _common/lpUtil/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _IP_UTIL_H_
00006 #define _IP_UTIL_H_
00007
00008 void showIpAddressesSerial();
00009 void showLinkStatus();
00010
00011 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00012 extern "C"
```

```
00013 {
00014     void showIpAddressesWeb(int socket, char *url);
00015 }
00016
00017 #endif
```

24.81 DHCP/ChangeIP/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00009 extern "C"
00010 {
00011     void showIpAddressesWeb(int socket, char *url);
00012 }
```

24.82 DHCP/ChangeIPViaWebpage/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00009 extern "C"
00010 {
00011     void showIpAddressesWeb(int socket, char *url);
00012 }
```

24.83 Ethernet/ManualConfig/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00009 extern "C"
00010 {
00011     void showIpAddressesWeb(int socket, char *url);
00012 }
```

24.84 IPv6/IPv6-DHCPv6/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00009 extern "C"
00010 {
00011     void showIpAddressesWeb(int socket, char *url);
00012 }
```

24.85 PlatformSpecific/MCF5441X/MOD5441X/Mod5441xFactory↩ App/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
```

```
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00009 extern "C"
00010 {
00011     void showIpAddressesWeb(int socket, char *url);
00012 }
```

24.86 PlatformSpecific/MCF5441X/NANO54415/NANO54415FactoryApp/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _IP_UTIL_H_
00006 #define _IP_UTIL_H_
00007
00008 void showIpAddressesSerial();
00009 void showLinkStatus();
00010
00011 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00012 extern "C"
00013 {
00014     void showIpAddressesWeb(int socket, char *url);
00015 }
00016
00017 #endif
```

24.87 PlatformSpecific/MCF5441X/RTC-OnChip/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00009 extern "C"
00010 {
00011     void showIpAddressesWeb(int socket, char *url);
00012 }
```

24.88 PlatformSpecific/SAME70/MODM7AE70/MODM7AE70FactoryApp/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _IP_UTIL_H_
00006 #define _IP_UTIL_H_
00007
00008 void showIpAddressesSerial();
00009 void showLinkStatus();
00010
00011 // Functions called by web server with FUNCTIONCALL tag need to be extern"C" to link
00012 extern "C"
00013 {
00014     void showIpAddressesWeb(int socket, char *url);
00015 }
00016
00017 #endif
```

24.89 ShowInterfaces/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
```

```
00007
00008 void showIpAddressesWeb(int socket, char *url);
```

24.90 VLan/src/ip_util.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void showIpAddressesSerial();
00006 void showLinkStatus();
00007
00008 void showIpAddressesWeb(int socket, char *url);
```

24.91 MyAlloc.h

```
00001
00002 class MyModAlloc : public DHCP::BlockAllocator
00003 {
00004    public:
00005     MyModAlloc();
00006     ~MyModAlloc();
00007     virtual bool OfferLease(DHCP::DhcpLeaseRequest *pLease, int intfNum);
00008     virtual bool RequestLease(DHCP::DhcpLeaseRequest *pLease, int intfNum);
00009     virtual bool SetStaticLease(DHCP::DhcpLeaseRequest *pLease);
00010 };
```

24.92 fileup.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #define FAT_UPDATE_OK (0)
00006 #define FAT_UPDATE_SAMEVERSION (-1)
00007 #define FAT_UPDATE_WRONG_PLATFORM (-2)
00008 #define FAT_UPDATE_BAD_FORMAT (-3)
00009 #define FAT_UPDATE_NO_MEM (-4)
00010
00011 /* This is the structure of an APP File */
00012 struct FlashStartUpStruct
00013 {
00014     unsigned long dwBlockRamStart;
00015     unsigned long dwExecutionAddr;
00016     unsigned long dwBlockSize;
00017     unsigned long dwSrcBlockSize;
00018     unsigned long dwBlockSum;
00019     unsigned long dwStructSum;
00020 };
00021
00022 int UpdateFromFat(F_FILE *f, BOOL bUpdateEvenIfCurrent);
```

24.93 _common/EFFS/FAT/src/cardtype.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _CARDTYPE_H
00007 #define _CARDTYPE_H
00008
00009 /*
00010   Change between devices by uncommenting one or the other. You cannot
00011   use both types at the same time. Whenever you change card types, you
00012   should do a make clean on your project.
00013
00014   Warning! You must have CFC bus interface hardware on your platform or
00015   the code will repeatedly trap. If you are getting traps you will need to
00016   perform an application download using the monitor program to recover.
00017 */
00018 // #define USE_SDHC     // SD/SDHC cards
00019 #define USE_MMC   // SD/MMC cards
00020 //#define USE_CFC       // Compact Flash cards
00021 //#define USE_RAM       // RAM FileSystem, See EFFS-RAM-minimal for details
00022
00023 #if (defined USE_CFC)
```

```
00024 #define EXT_FLASH_DRV_NUM (CFC_DRV_NUM)
00025 #elif (defined USE_RAM)
00026 #define EXT_FLASH_DRV_NUM (F_RAM_DRIVE0)
00027 #elif (defined USE_MMC)
00028 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00029 #elif (defined USE_SDHC)
00030 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00031 #else
00032 #define EXT_FLASH_DRV_NUM (-1)
00033 #endif
00034
00035 #endif /* _CARDTYPE_H */
```

24.94 EFFS/Fat/Performance/src/cardtype.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _CARDTYPE_H
00007 #define _CARDTYPE_H
00008
00009 /*
00010   Change between devices by uncommenting one or the other. You cannot
00011   use both types at the same time. Whenever you change card types, you
00012   should do a make clean on your project.
00013
00014   Warning! You must have CFC bus interface hardware on your platform or
00015   the code will repeatedly trap. If you are getting traps you will need to
00016   perform an application download using the monitor program to recover.
00017 */
00018 // #define USE_SDHC     // SD/SDHC cards
00019 #define USE_MMC   // SD/MMC cards
00020 //#define USE_CFC       // Compact Flash cards
00021 //#define USE_RAM       // RAM FileSystem, See EFFS-RAM-minimal for details
00022
00023 #if (defined USE_CFC)
00024 #define EXT_FLASH_DRV_NUM (CFC_DRV_NUM)
00025 #elif (defined USE_RAM)
00026 #define EXT_FLASH_DRV_NUM (F_RAM_DRIVE0)
00027 #elif (defined USE_MMC)
00028 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00029 #elif (defined USE_SDHC)
00030 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00031 #else
00032 #define EXT_FLASH_DRV_NUM (-1)
00033 #endif
00034
00035 #endif /* _CARDTYPE_H */
```

24.95 PlatformSpecific/MCF5441X/EffsLoadAppFromFlashCard/src/cardtype.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _CARDTYPE_H
00007 #define _CARDTYPE_H
00008
00009 /*
00010   Change between devices by uncommenting one or the other. You cannot
00011   use both types at the same time. Whenever you change card types, you
00012   should do a make clean on your project.
00013
00014   Warning! You must have CFC bus interface hardware on your platform or
00015   the code will repeatedly trap. If you are getting traps you will need to
00016   perform an application download using the monitor program to recover.
00017 */
00018 // #define USE_SDHC     // SD/SDHC cards
00019 #define USE_MMC   // SD/MMC cards
00020 //#define USE_CFC       // Compact Flash cards
00021 //#define USE_RAM       // RAM FileSystem, See EFFS-RAM-minimal for details
00022
00023 #if (defined USE_CFC)
00024 #define EXT_FLASH_DRV_NUM (CFC_DRV_NUM)
00025 #elif (defined USE_RAM)
00026 #define EXT_FLASH_DRV_NUM (F_RAM_DRIVE0)
00027 #elif (defined USE_MMC)
```

```
00028 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00029 #elif (defined USE_SDHC)
00030 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00031 #else
00032 #define EXT_FLASH_DRV_NUM (-1)
00033 #endif
00034
00035 #endif /* _CARDTYPE_H */
```

24.96 PlatformSpecific/MCF5441X/EffsSDHC/src/cardtype.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _CARDTYPE_H
00007 #define _CARDTYPE_H
00008
00009 /*
00010   Change between devices by uncommenting one or the other. You cannot
00011   use both types at the same time. Whenever you change card types, you
00012   should do a make clean on your project.
00013
00014   Warning! You must have CFC bus interface hardware on your platform or
00015   the code will repeatedly trap. If you are getting traps you will need to
00016   perform an application download using the monitor program to recover.
00017 */
00018  #define USE_SDHC       // SD/SDHC cards
00019 //#define USE_MMC   // SD/MMC cards
00020 //#define USE_CFC       // Compact Flash cards
00021 //#define USE_RAM       // RAM FileSystem, See EFFS-RAM-minimal for details
00022
00023 #if (defined USE_CFC)
00024 #define EXT_FLASH_DRV_NUM (CFC_DRV_NUM)
00025 #elif (defined USE_RAM)
00026 #define EXT_FLASH_DRV_NUM (F_RAM_DRIVE0)
00027 #elif (defined USE_MMC)
00028 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00029 #elif (defined USE_SDHC)
00030 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00031 #else
00032 #define EXT_FLASH_DRV_NUM (-1)
00033 #endif
00034
00035 #endif /* _CARDTYPE_H */
```

24.97 PlatformSpecific/MCF5441X/MOD5441X/Mod5441xFactory↩ App/src/cardtype.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _CARDTYPE_H
00007 #define _CARDTYPE_H
00008
00009 /*
00010   Change between devices by uncommenting one or the other. You cannot
00011   use both types at the same time. Whenever you change card types, you
00012   should do a make clean on your project.
00013
00014   Warning! You must have CFC bus interface hardware on your platform or
00015   the code will repeatedly trap. If you are getting traps you will need to
00016   perform an application download using the monitor program to recover.
00017 */
00018 #define USE_MMC   // SD/MMC cards
00019 //#define USE_CFC       // Compact Flash cards
00020 //#define USE_RAM       // RAM FileSystem, See EFFS-RAM-minimal for details
00021 //#define EXT_FLASH_DRV_NUM (F_RAM_DRIVE0)
00022
00023 #if (defined USE_CFC)
00024 #define EXT_FLASH_DRV_NUM (CFC_DRV_NUM)
00025 #else
00026 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00027 #endif
00028
00029 #endif /* _CARDTYPE_H */
```

24.98 SSH/SecureSerToEthFactoryApp/src/cardtype.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Select the card type */
00006 #ifndef _CARDTYPE_H
00007 #define _CARDTYPE_H
00008
00009 /*
00010   Change between devices by uncommenting one or the other. You cannot
00011   use both types at the same time. Whenever you change card types, you
00012   should do a make clean on your project.
00013
00014   Warning! You must have CFC bus interface hardware on your platform or
00015   the code will repeatedly trap. If you are getting traps you will need to
00016   perform an application download using the monitor program to recover.
00017 */
00018 // #define USE_SDHC     // SD/SDHC cards
00019 #define USE_MMC   // SD/MMC cards
00020 //#define USE_CFC       // Compact Flash cards
00021 //#define USE_RAM       // RAM FileSystem, See EFFS-RAM-minimal for details
00022
00023 #if (defined USE_CFC)
00024 #define EXT_FLASH_DRV_NUM (CFC_DRV_NUM)
00025 #elif (defined USE_RAM)
00026 #define EXT_FLASH_DRV_NUM (F_RAM_DRIVE0)
00027 #elif (defined USE_MMC)
00028 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00029 #elif (defined USE_SDHC)
00030 #define EXT_FLASH_DRV_NUM (MMC_DRV_NUM)
00031 #else
00032 #define EXT_FLASH_DRV_NUM (-1)
00033 #endif
00034
00035 #endif /* _CARDTYPE_H */
```

24.99 _common/EFFS/FAT/src/FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
00009
00010 #include "cardtype.h"
00011
00012 #define MAX_EFFS_ERRORCODE (38)
00013 extern char EffsErrorCode[][80];
00014
00015 #ifdef __cplusplus
00016 extern "C"
00017 {
00018 #endif
00019
00020 // FAT Media Types for Format
00021 #define F_FAT12_FORMAT (1)
00022 #define F_FAT16_FORMAT (2)
00023 #define F_FAT32_FORMAT (3)
00024
00025     void DisplayEffsErrorCode(int code);
00026     uint8_t InitExtFlash();
00027     uint8_t UnmountExtFlash();
00028     uint8_t FormatExtFlash(long FATtype = F_FAT32_FORMAT);
00029     uint8_t DisplayEffsSpaceStats();
00030     uint8_t DumpDir();
00031     uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00032     uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00033     uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t NumBytes);
00034     uint8_t DeleteFile(char *pFileName);
00035     int DeleteAllFiles();
00036     void ReadWriteTest(char *FileName = "TestFile.txt");
00037     void DisplayTextFile(char *FileName);
00038     void fgets_test(char *FileName);
00039     void fprintf_test();
00040     void fputs_test(char *FileName);
00041
00042 #ifdef __cplusplus
00043 }
00044 #endif
```

```
00045
00046 #endif
```

24.100 _common/EFFS/STD/src/FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
00009
00010 #define MAX_EFFS_ERRORCODE (38)
00011 extern char EffsErrorCode[][80];
00012
00013 #ifdef __cplusplus
00014 extern "C"
00015 {
00016 #endif
00017
00018     void DisplayEffsErrorCode(int code);
00019     uint8_t FormatEffsStdFlash();
00020     uint8_t DisplayEffsSpaceStats();
00021     uint8_t DumpDir();
00022     uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00023     uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00024     uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t NumBytes);
00025     uint8_t DeleteFile(char *pFileName);
00026     void ReadWriteTest();
00027     void DisplayTextFile(char *FileName);
00028     void fgets_test(char *FileName);
00029     void fprintf_test();
00030     void fputs_test(char *FileName);
00031
00032 #ifdef __cplusplus
00033 }
00034 #endif
00035
00036 #endif
```

24.101 EFFS/Fat/Performance/src/FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
00009
00010 #include "cardtype.h"
00011
00012 #define MAX_EFFS_ERRORCODE (38)
00013 extern char EffsErrorCode[][80];
00014
00015 #ifdef __cplusplus
00016 extern "C"
00017 {
00018 #endif
00019
00020 // FAT Media Types for Format
00021 #define F_FAT12_FORMAT (1)
00022 #define F_FAT16_FORMAT (2)
00023 #define F_FAT32_FORMAT (3)
00024
00025     void DisplayEffsErrorCode(int code);
00026     uint8_t InitExtFlash();
00027     uint8_t UnmountExtFlash();
00028     uint8_t FormatExtFlash(long FATtype = F_FAT32_FORMAT);
00029     uint8_t DisplayEffsSpaceStats();
00030     uint8_t DumpDir();
00031     uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00032     uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00033     uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t NumBytes);
00034     uint8_t DeleteFile(char *pFileName);
00035     int DeleteAllFiles();
00036     void ReadWriteTest(char *FileName = "TestFile.txt");
00037     void DisplayTextFile(char *FileName);
00038     void fgets_test(char *FileName);
```

```
00039     void fprintf_test();
00040     void fputs_test(char *FileName);
00041
00042 #ifdef __cplusplus
00043 }
00044 #endif
00045
00046 #endif
```

24.102 Parallax/src/FileSystemUtils.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 #ifndef _FILESYSUTIL_H
00028 #define _FILESYSUTIL_H
00029
00030 #include <effs_fat/fat.h>
00031
00032 #define MAX_EFFS_ERRORCODE (38)
00033 extern char EffsErrorCode[][80];
00034
00035 #ifdef __cplusplus
00036 extern "C"
00037 {
00038 #endif
00039
00040     void DisplayEffsErrorCode(int code);
00041     uint8_t FormatEffsStdFlash();
00042     uint8_t DisplayEffsSpaceStats();
00043     uint8_t DumpDir();
00044     uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00045     uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00046     uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t NumBytes);
00047     uint8_t DeleteFile(char *pFileName);
00048     void ReadWriteTest();
00049     void DisplayTextFile(char *FileName);
00050     void fgets_test(char *FileName);
00051     void fprintf_test();
00052     void fputs_test(char *FileName);
00053
00054 #ifdef __cplusplus
00055 }
00056 #endif
00057
00058 #endif
```

24.103 PlatformSpecific/MCF5441X/EffsLoadAppFromFlashCard/src/↩ FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
```

```
00009
00010 extern char EffsErrorCode[][80];
00011
00012 // FAT Media Types for Format
00013 #define F_FAT12_FORMAT (1)
00014 #define F_FAT16_FORMAT (2)
00015 #define F_FAT32_FORMAT (3)
00016
00017 int OpenOnBoardFlash();
00018 int OpenOffBoardFlash();
00019 int UnmountFlash(int drv);
00020
00021 void DisplayEffsErrorCode(int code);
00022 uint8_t FormatExtFlash(int drv, long FATtype = F_FAT32_FORMAT);
00023 uint8_t DisplayEffsSpaceStats(int drv);
00024 uint8_t DumpDir();
00025
00026 uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t Numuint8_ts);
00027 uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t Numuint8_ts);
00028 uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t Numuint8_ts);
00029 uint8_t DeleteFile(char *pFileName);
00030
00031 void ReadWriteTest(const char *FileName = "TestFile.txt");
00032 void DisplayTextFile(char *FileName);
00033 void fgets_test(char *FileName);
00034 void fprintf_test();
00035 void fputs_test(char *FileName);
00036
00037 #endif
```

24.104 PlatformSpecific/MCF5441X/EffsSDHC/src/FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
00009
00010 #include "cardtype.h"
00011
00012 #define MAX_EFFS_ERRORCODE (38)
00013 extern char EffsErrorCode[][80];
00014
00015 #ifdef __cplusplus
00016 extern "C"
00017 {
00018 #endif
00019
00020 // FAT Media Types for Format
00021 #define F_FAT12_FORMAT (1)
00022 #define F_FAT16_FORMAT (2)
00023 #define F_FAT32_FORMAT (3)
00024
00025     void DisplayEffsErrorCode(int code);
00026     uint8_t InitExtFlash();
00027     uint8_t UnmountExtFlash();
00028     uint8_t FormatExtFlash(long FATtype = F_FAT32_FORMAT);
00029     uint8_t DisplayEffsSpaceStats();
00030     uint8_t DumpDir();
00031     uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00032     uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t NumBytes);
00033     uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t NumBytes);
00034     uint8_t DeleteFile(char *pFileName);
00035     int DeleteAllFiles();
00036     void ReadWriteTest(char *FileName = "TestFile.txt");
00037     void DisplayTextFile(char *FileName);
00038     void fgets_test(char *FileName);
00039     void fprintf_test();
00040     void fputs_test(char *FileName);
00041
00042 #ifdef __cplusplus
00043 }
00044 #endif
00045
00046 #endif
```

24.105 PlatformSpecific/MCF5441X/MOD5441X/EffsMultipleMmc/src/FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
00009
00010 extern char EffsErrorCode[][80];
00011
00012 // FAT Media Types for Format
00013 #define F_FAT12_FORMAT (1)
00014 #define F_FAT16_FORMAT (2)
00015 #define F_FAT32_FORMAT (3)
00016
00017 int OpenOnBoardFlash();
00018 int OpenOffBoardFlash();
00019 int UnmountFlash(int drv);
00020
00021 void DisplayEffsErrorCode(int code);
00022 uint8_t FormatExtFlash(int drv, long FATtype = F_FAT32_FORMAT);
00023 uint8_t DisplayEffsSpaceStats(int drv);
00024 uint8_t DumpDir();
00025
00026 uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t Numuint8_ts);
00027 uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t Numuint8_ts);
00028 uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t Numuint8_ts);
00029 uint8_t DeleteFile(char *pFileName);
00030
00031 void ReadWriteTest(const char *FileName = "TestFile.txt");
00032 void DisplayTextFile(char *FileName);
00033 void fgets_test(char *FileName);
00034 void fprintf_test();
00035 void fputs_test(char *FileName);
00036
00037 #endif
```

24.106 PlatformSpecific/MCF5441X/MOD5441X/Mod5441xFactoryApp/src/FileSystemUtils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FILESYSUTIL_H
00006 #define _FILESYSUTIL_H
00007
00008 #include <effs_fat/fat.h>
00009
00010 #define MAX_EFFS_ERRORCODE (38)
00011 extern char EffsErrorCode[][80];
00012
00013 #ifdef __cplusplus
00014 extern "C"
00015 {
00016 #endif
00017
00018 // FAT Media Types for Format
00019 #define F_FAT12_FORMAT (1)
00020 #define F_FAT16_FORMAT (2)
00021 #define F_FAT32_FORMAT (3)
00022
00023     int OpenOnBoardFlash();
00024     int OpenOffBoardFlash();
00025     int UnmountFlash(int drv);
00026
00027     void DisplayEffsErrorCode(int code);
00028     uint8_t FormatExtFlash(int drv, long FATtype = F_FAT32_FORMAT);
00029     uint8_t DisplayEffsSpaceStats(int drv);
00030     uint8_t DumpDir();
00031     uint32_t WriteFile(uint8_t *pDataToWrite, char *pFileName, uint32_t Numuint8_ts);
00032     uint32_t AppendFile(uint8_t *pDataToWrite, char *pFileName, uint32_t Numuint8_ts);
00033     uint32_t ReadFile(uint8_t *pReadBuffer, char *pFileName, uint32_t Numuint8_ts);
00034     uint8_t DeleteFile(char *pFileName);
00035     void ReadWriteTest(const char *FileName = "TestFile.txt");
00036     void DisplayTextFile(char *FileName);
00037     void fgets_test(char *FileName);
00038     void fprintf_test();
```

```
00039     void fputs_test(char *FileName);
00040
00041 #ifdef __cplusplus
00042 }
00043 #endif
00044
00045 #endif
```

24.107 ExtraFdCircBuffer.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <predef.h>
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <iosys.h>
00009 #include <nbrtos.h>
00010 #include <iointernal.h>
00011
00012 #ifndef _EXTRAFD_CIRC_BUFFER_H
00013 #define _EXTRAFD_CIRC_BUFFER_H
00014
00015 // A circular buffer object, referred to as a "cbo"
00016 struct CircularBufferObject
00017 {
00018     OS_CRIT criticalSection;   // Critical section to make it task safe
00019
00020     uint32_t readIndex;    // Buffer index of next char to get
00021     uint32_t writeIndex;   // Buffer index of next char to put
00022
00023     uint32_t dataBufferSize;   // Size of data buffer
00024     char *dataBuffer;          // Data storage space
00025
00026     bool enableOverwrite;     // Continuously overwrite buffer instead of setting full flag
00027     char overwriteTermByte;   // When in overwrite mode and buffer is full, bytes will be
00028                               //   flushed up to the terminator instead of one at a time.
00029                               //   Useful for lines/strings terminated with \r\n
00030 };
00031
00032 // int CircularBufferInit(CircularBufferObject *pCbo, char *pBuffer, uint32_t bufferSize);
00033 int CircularBufferInit(CircularBufferObject *pCbo, char *pBuffer, uint32_t bufferSize, bool
      enableOverwrite, uint8_t overwriteTermByte);
00034 int CircularBufferRead(int fd, char *buf, int nbytes);
00035 int CircularBufferWrite(int fd, char *buf, int nbytes);
00036 void CircularBufferRawPrint(int fdCircBuffer, int fdDestination);
00037
00038 #endif
```

24.108 data.h

```
00001 struct datapoint
00002 {time_t when;
00003  float tide;
00004  int  tidepos;
00005  bool dark;
00006 };
00007
00008 const int DisplayWidth=800;
00009 const int DisplayHt=460;
00010 const int BottomBand=480-DisplayHt;
00011 const double DisplayTop=7.0;
00012 const double DisplayBot=-3.0;
00013 const double delta=((double)DisplayHt)/(DisplayTop-DisplayBot);
00014
00015 inline int DataToPos(double v)
00016 {
00017     double offset=v-DisplayBot;
00018     int rv=DisplayHt-(int)(delta*offset);
00019     if (rv<0) rv=0;
00020     if(rv>=DisplayHt) rv=DisplayHt-1;
00021     return rv;
00022 }
00023 extern volatile datapoint dataplot[DisplayWidth];
00024 extern volatile int LeftEdge;
00025 extern OS_CRIT CalcCrit;
```

24.109 drawimage.cpp File Reference

Function definitions for the DrawImageObject class.

```
#include <iosys.h>
#include <malloc.h>
#include <string.h>
#include "drawimage.h"
#include "gifCompress.h"
```

Functions

- void FlushData (int fd)
- void WriteData (int fd, const char *c, int siz)
- void OutputGifChar (const char c, int fd)
- void WriteOneChar (const FontData *pf, char c, int xp, int yp, uint8_t color, DrawImageObject &doi)

24.109.1 Detailed Description

Function definitions for the DrawImageObject class.
This class hold the data associated with and draws GIF images with text to a given file descriptor

24.109.2 Function Documentation

24.109.2.1 FlushData()

```
void FlushData (
            int fd )
```

FlushData
Flush any unwritten data to the output

24.109.2.2 OutputGifChar()

```
void OutputGifChar (
            const char c,
            int fd )
```

OutputGifChar
Writes a GIF char to the given fd

24.109.2.3 WriteData()

```
void WriteData (
            int fd,
            const char * c,
            int siz )
```

WriteData
This is the function that writes data to the output socket

24.109.2.4 WriteOneChar()

```
void WriteOneChar (
            const FontData * pf,
            char c,
            int xp,
            int yp,
            uint8_t color,
            DrawImageObject & doi )
```

WriteOneChar
Writes a single character

24.110 drawimage.cpp File Reference

Function definitions for the DrawImageObject class.

```
#include <iosys.h>
#include <malloc.h>
#include <string.h>
#include "drawimage.h"
#include "gifCompress.h"
```

Functions

- void FlushData (int fd)
- void WriteData (int fd, const char *c, int siz)
- void OutputGifChar (const char c, int fd)
- void WriteOneChar (const FontData *pf, char c, int xp, int yp, uint8_t color, DrawImageObject &doi)

24.110.1 Detailed Description

Function definitions for the DrawImageObject class.
This class hold the data associated with and draws GIF images with text to a given file descriptor

24.110.2 Function Documentation

24.110.2.1 FlushData()

```
void FlushData (
            int fd )
```

FlushData
Flush any unwritten data to the output

24.110.2.2 OutputGifChar()

```
void OutputGifChar (
            const char c,
            int fd )
```

OutputGifChar
Writes a GIF char to the given fd

24.110.2.3 WriteData()

```
void WriteData (
            int fd,
            const char * c,
            int siz )
```

WriteData
This is the function that writes data to the output socket

24.110.2.4 WriteOneChar()

```
void WriteOneChar (
            const FontData * pf,
            char c,
```

```
            int xp,
            int yp,
            uint8_t color,
            DrawImageObject & doi )
```

WriteOneChar
Writes a single character

24.111 JSON/DemoNetBurner/src/drawimage.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* This file contains a simple class for drawing Graphics into a GIF image */
00006
00007 #ifndef _DRAWIMAGE_H_
00008 #define _DRAWIMAGE_H_
00009
00010 #include <basictypes.h>
00011
00012 #include "gifCompress.h"
00013
00014 class GitCompress;
00015
00016 class DrawImageObject
00017 {
00018     uint8_t *m_pImageBuffer = nullptr;
00019     uint8_t *m_pColorArray = nullptr;
00020
00021     int m_xSize = 0;
00022     int m_ySize = 0;
00023     int m_nColors = 0;
00024
00025     int m_curx = 0;
00026     int m_cury = 0;
00027
00028     bool m_trans = false;
00029     uint8_t m_transIndex = 0;
00030
00031     GitCompress m_gitCompress;
00032
00033     friend class GitCompress;
00034
00035    private:
00036     int GIFNextPixel();
00037     void compress(int init_bits, int fd);
00038
00039    public:
00040     /* You must specify the size and color depth of the GIF object in the constructor */
00041     DrawImageObject(int x, int y, int ncolors, bool transparent, uint8_t transIndex);
00042     ~DrawImageObject();
00043
00044     /* Set a specific pixel to a specific color */
00045     void PutPixel(int x, int y, uint8_t color);
00046
00047     /* Get the color of a specific pixel */
00048     uint8_t GetPixel(int x, int y);
00049
00050     /* All colors are index based. You must define the color for each index */
00051     void SetColor(uint8_t index, uint8_t red, uint8_t green, uint8_t blue);
00052
00053     /* Draw a line */
00054     void Line(int x1, int y1, int x2, int y2, uint8_t colorindex);
00055     void DashLine(int x1, int y1, int x2, int y2, uint8_t colorindex, int space);
00056
00057     /* Draw a box */
00058     void Box(int x1, int y1, int x2, int y2, uint8_t colorindex);
00059
00060     /* Draw a filled box */
00061     void FilledBox(int x1, int y1, int x2, int y2, uint8_t fillc, uint8_t outlinec);
00062
00063     /* Draw text */
00064     void Text(const char *pText, int x1, int x2, const char *fontrecord, uint8_t color);
00065     int TextXsize(const char *pText, const char *fontrecord);
00066     int TextYsize(const char *pText, const char *fontrecord);
00067
00068     /* After you have  done all of your drawing you must call this function to send the GIF */
00069     void WriteGIF(int fd);
00070 };
00071
00072 extern const char GiantFont[];
00073 extern const char LargeFont[];
```

```
00074 extern const char MediumFont[];
00075 extern const char SmallFont[];
00076 extern const char TinyFont[];
00077
00078 #endif /* _DRAWIMAGE_H_ */
```

24.112 Web/GifCanvas/src/drawimage.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* This file contains a simple class for drawing Graphics into a GIF image */
00006
00007 #ifndef _DRAWIMAGE_H_
00008 #define _DRAWIMAGE_H_
00009
00010 #include <basictypes.h>
00011
00012 #include "gifCompress.h"
00013
00014 class GitCompress;
00015
00016 class DrawImageObject
00017 {
00018     uint8_t *m_pImageBuffer = nullptr;
00019     uint8_t *m_pColorArray = nullptr;
00020
00021     int m_xSize = 0;
00022     int m_ySize = 0;
00023     int m_nColors = 0;
00024
00025     int m_curx = 0;
00026     int m_cury = 0;
00027
00028     bool m_trans = false;
00029     uint8_t m_transIndex = 0;
00030
00031     GitCompress m_gitCompress;
00032
00033     friend class GitCompress;
00034
00035   private:
00036     int GIFNextPixel();
00037     void compress(int init_bits, int fd);
00038
00039   public:
00040     /* You must specify the size and color depth of the GIF object in the constructor */
00041     DrawImageObject(int x, int y, int ncolors, bool transparent, uint8_t transIndex);
00042     ~DrawImageObject();
00043
00044     /* Set a specific pixel to a specific color */
00045     void PutPixel(int x, int y, uint8_t color);
00046
00047     /* Get the color of a specific pixel */
00048     uint8_t GetPixel(int x, int y);
00049
00050     /* All colors are index based. You must define the color for each index */
00051     void SetColor(uint8_t index, uint8_t red, uint8_t green, uint8_t blue);
00052
00053     /* Draw a line */
00054     void Line(int x1, int y1, int x2, int y2, uint8_t colorindex);
00055
00056     /* Draw a box */
00057     void Box(int x1, int y1, int x2, int y2, uint8_t colorindex);
00058
00059     /* Draw a filled box */
00060     void FilledBox(int x1, int y1, int x2, int y2, uint8_t fillc, uint8_t outlinec);
00061
00062     /* Draw text */
00063     void Text(const char *pText, int x1, int x2, const char *fontrecord, uint8_t color);
00064     int TextXsize(const char *pText, const char *fontrecord);
00065     int TextYsize(const char *pText, const char *fontrecord);
00066
00067     /* After you have  done all of your drawing you must call this function to send the GIF */
00068     void WriteGIF(int fd);
00069 };
00070
00071 extern const char GiantFont[];
00072 extern const char LargeFont[];
00073 extern const char MediumFont[];
00074 extern const char SmallFont[];
00075 extern const char TinyFont[];
00076
00077 #endif /* _DRAWIMAGE_H_ */
```

24.113 JSON/DemoNetBurner/src/gifCompress.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-----------------------------------------------------------------------
00006  *
00007  * miGIF Compression - mouse and ivo's GIF-compatible compression
00008  *
00009  *          -run length encoding compression routines-
00010  *
00011  * Copyright (C) 1998 Hutchison Avenue Software Corporation
00012  *               http://www.hasc.com
00013  *               info@hasc.com
00014  *
00015  * Permission to use, copy, modify, and distribute this software and its
00016  * documentation for any purpose and without fee is hereby granted, provided
00017  * that the above copyright notice appear in all copies and that both that
00018  * copyright notice and this permission notice appear in supporting
00019  * documentation.  This software is provided "AS IS." The Hutchison Avenue
00020  * Software Corporation disclaims all warranties, either express or implied,
00021  * including but not limited to implied warranties of merchantability and
00022  * fitness for a particular purpose, with respect to this code and accompanying
00023  * documentation.
00024  *
00025  * The miGIF compression routines do not, strictly speaking, generate files
00026  * conforming to the GIF spec, since the image data is not LZW-compressed
00027  * (this is the point: in order to avoid transgression of the Unisys patent
00028  * on the LZW algorithm.)  However, miGIF generates data streams that any
00029  * reasonably sane LZW decompresser will decompress to what we want.
00030  *
00031  * miGIF compression uses run length encoding. It compresses horizontal runs
00032  * of pixels of the same color. This type of compression gives good results
00033  * on images with many runs, for example images with lines, text and solid
00034  * shapes on a solid-colored background. It gives little or no compression
00035  * on images with few runs, for example digital or scanned photos.
00036  *
00037  *                               der Mouse
00038  *                     mouse@rodents.montreal.qc.ca
00039  *             7D C8 61 52 5D E7 2D 39  4E F1 31 3E E8 B3 27 4B
00040  *
00041  *                                ivo@hasc.com
00042  *
00043  * The Graphics Interchange Format(c) is the Copyright property of
00044  * CompuServe Incorporated.  GIF(sm) is a Service Mark property of
00045  * CompuServe Incorporated.
00046  */
00047
00048 #ifndef _GIT_COMPRESS_H_
00049 #define _GIT_COMPRESS_H_
00050 #pragma once
00051
00052 #define GIFBITS 12
00053
00054 //#include "drawimage.h"
00055
00056 class DrawImageObject;
00057
00058 class GitCompress
00059 {
00060    public:
00061     GitCompress();
00062     void ResetStatistics();
00063     void compress(DrawImageObject &dio, int init_bits, int fd);
00064
00065    private:
00066     void write_block(void);
00067     void block_out(unsigned char c);
00068     void block_flush(void);
00069     void output(int val);
00070     void output_flush(void);
00071     void did_clear(void);
00072     void output_plain(int c);
00073     unsigned int isqrt(unsigned int x);
00074     unsigned int compute_triangle_count(unsigned int count, unsigned int nrepcodes);
00075     void max_out_clear(void);
00076     void reset_out_clear(void);
00077     void rl_flush_fromclear(int count);
00078     void rl_flush_clearorrep(int count);
00079     void rl_flush_withtable(int count);
00080     void rl_flush(void);
00081
00082     int rl_pixel = 0;
00083     int rl_basecode = 0;
00084     int rl_count = 0;
00085     int rl_table_pixel = 0;
```

```
00086     int rl_table_max = 0;
00087     int just_cleared = 0;
00088     int out_bits = 0;
00089     int out_bits_init = 0;
00090     int out_count = 0;
00091     int out_bump = 0;
00092     int out_bump_init = 0;
00093     int out_clear = 0;
00094     int out_clear_init = 0;
00095     int max_ocodes = 0;
00096     int code_clear = 0;
00097     int code_eof = 0;
00098     unsigned int obuf = 0;
00099     int obits = 0;
00100     int ofd = 0;
00101     int oblen = 0;
00102     unsigned char oblock[256];
00103     bool m_transparent = false;
00104 };
00105
00106 #endif /* _GIT_COMPRESS_H_ */
00107
00108 /*-----------------------------------------------------------------------
00109  *
00110  * End of miGIF section  - See copyright notice at start of section.
00111  *
00112  *-----------------------------------------------------------------------*/
```

24.114 Web/GifCanvas/src/gifCompress.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-----------------------------------------------------------------------
00006  *
00007  * miGIF Compression - mouse and ivo's GIF-compatible compression
00008  *
00009  *          -run length encoding compression routines-
00010  *
00011  * Copyright (C) 1998 Hutchison Avenue Software Corporation
00012  *               http://www.hasc.com
00013  *               info@hasc.com
00014  *
00015  * Permission to use, copy, modify, and distribute this software and its
00016  * documentation for any purpose and without fee is hereby granted, provided
00017  * that the above copyright notice appear in all copies and that both that
00018  * copyright notice and this permission notice appear in supporting
00019  * documentation.  This software is provided "AS IS." The Hutchison Avenue
00020  * Software Corporation disclaims all warranties, either express or implied,
00021  * including but not limited to implied warranties of merchantability and
00022  * fitness for a particular purpose, with respect to this code and accompanying
00023  * documentation.
00024  *
00025  * The miGIF compression routines do not, strictly speaking, generate files
00026  * conforming to the GIF spec, since the image data is not LZW-compressed
00027  * (this is the point: in order to avoid transgression of the Unisys patent
00028  * on the LZW algorithm.)  However, miGIF generates data streams that any
00029  * reasonably sane LZW decompresser will decompress to what we want.
00030  *
00031  * miGIF compression uses run length encoding. It compresses horizontal runs
00032  * of pixels of the same color. This type of compression gives good results
00033  * on images with many runs, for example images with lines, text and solid
00034  * shapes on a solid-colored background. It gives little or no compression
00035  * on images with few runs, for example digital or scanned photos.
00036  *
00037  *                               der Mouse
00038  *                       mouse@rodents.montreal.qc.ca
00039  *             7D C8 61 52 5D E7 2D 39  4E F1 31 3E E8 B3 27 4B
00040  *
00041  *                                ivo@hasc.com
00042  *
00043  * The Graphics Interchange Format(c) is the Copyright property of
00044  * CompuServe Incorporated.  GIF(sm) is a Service Mark property of
00045  * CompuServe Incorporated.
00046  */
00047
00048 #ifndef _GIT_COMPRESS_H_
00049 #define _GIT_COMPRESS_H_
00050 #pragma once
00051
00052 #define GIFBITS 12
00053
00054 //#include "drawimage.h"
00055
```

```
00056 class DrawImageObject;
00057
00058 class GitCompress
00059 {
00060    public:
00061     GitCompress();
00062     void ResetStatistics();
00063     void compress(DrawImageObject &dio, int init_bits, int fd);
00064
00065    private:
00066     void write_block(void);
00067     void block_out(unsigned char c);
00068     void block_flush(void);
00069     void output(int val);
00070     void output_flush(void);
00071     void did_clear(void);
00072     void output_plain(int c);
00073     unsigned int isqrt(unsigned int x);
00074     unsigned int compute_triangle_count(unsigned int count, unsigned int nrepcodes);
00075     void max_out_clear(void);
00076     void reset_out_clear(void);
00077     void rl_flush_fromclear(int count);
00078     void rl_flush_clearorrep(int count);
00079     void rl_flush_withtable(int count);
00080     void rl_flush(void);
00081
00082     int rl_pixel = 0;
00083     int rl_basecode = 0;
00084     int rl_count = 0;
00085     int rl_table_pixel = 0;
00086     int rl_table_max = 0;
00087     int just_cleared = 0;
00088     int out_bits = 0;
00089     int out_bits_init = 0;
00090     int out_count = 0;
00091     int out_bump = 0;
00092     int out_bump_init = 0;
00093     int out_clear = 0;
00094     int out_clear_init = 0;
00095     int max_ocodes = 0;
00096     int code_clear = 0;
00097     int code_eof = 0;
00098     unsigned int obuf = 0;
00099     int obits = 0;
00100     int ofd = 0;
00101     int oblen = 0;
00102     unsigned char oblock[256];
00103     bool m_transparent = false;
00104 };
00105
00106 #endif /* _GIT_COMPRESS_H_ */
00107
00108 /*-----------------------------------------------------------------------
00109  *
00110  * End of miGIF section  - See copyright notice at start of section.
00111  *
00112  *-----------------------------------------------------------------------*/
```

24.115 JSON/DemoNetBurner/src/htmlvar.h

```
00001 #ifndef HTMLVARS_H_
00002 #define HTMLVARS_H_
00003
00004 #include <nbrtos.h>     // For access to TimeTick
00005
00006 extern vuint32_t  Secs;
00007 extern const char * gBuildDate;
00008 extern const char * gBuildTime;
00009 extern char gBootTime[];
00010 extern char gBootDate[];
00011 const char * GetReleaseTag();
00012 const char * ShowHost();
00013
00014
00015 #endif
```

24.116 JSON/SimpleJSONHtml/src/htmlvar.h

```
00001 //Define the variables the web page will need to render the results
00002 extern vuint32_t  Secs;
00003 extern const char * gBuildDate;
00004 extern const char * gBuildTime;
```

```
00005 const char * GetReleaseTag();
```

24.117 JSON/SimplePostReceiver/src/htmlvar.h

```
00001 const char * GetHost();
00002
```

24.118 PlatformSpecific/MCF5441X/SB800EX/SB800AsDiag↩ Monitor/src/htmlvar.h

```
00001 #include <config_obj.h>
00002 extern config_chooser Ch1;
00003 extern config_chooser Ch2;
```

24.119 SSL/SSLConfigMirror/src/htmlvar.h

```
00001 #ifndef HTMLVARS_H_
00002 #define HTMLVARS_H_
00003
00004 #include <nbrtos.h>     // For access to TimeTick
00005
00006 extern vuint32_t  Secs;
00007 extern const char * gBuildDate;
00008 extern const char * gBuildTime;
00009 extern char gBootTime[];
00010 extern char gBootDate[];
00011 const char * GetReleaseTag();
00012
00013
00014 #endif
```

24.120 Web/HtmlPostDateTime/src/htmlvar.h

```
00001 //Nothing here
```

24.121 Web/HtmlServerGetRequest/src/htmlvar.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef HTMLVARS_H_
00006 #define HTMLVARS_H_
00007
00008 #include <nbrtos.h>   // For access to TimeTick
00009
00010 extern vuint32_t Secs;
00011
00012 #endif /*HTMLVARS_H_*/
```

24.122 Web/HtmlVariables/src/htmlvar.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef HTMLVARS_H_
00006 #define HTMLVARS_H_
00007
00008 #include <nbrtos.h>   // For access to TimeTick
00009
00010 extern int gIntVal;
00011 extern char gStrVal[];
00012
00013 const char *FooWithParameters(int fd, int v);
00014
00015 #endif /*HTMLVARS_H_*/
```

24.123 Web/SignedApp/src/htmlvar.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef HTMLVARS_H_
00006 #define HTMLVARS_H_
00007
00008 #include <tcp.h>
00009
00010 #endif /*HTMLVARS_H_*/
```

24.124 WebSockets/Console/src/htmlvar.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef HTMLVARS_H_
00005 #define HTMLVARS_H_
00006
00007 #include <tcp.h>
00008
00009 #endif /*HTMLVARS_H_*/
```

24.125 WebSockets/Echo/src/htmlvar.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef HTMLVARS_H_
00005 #define HTMLVARS_H_
00006
00007 #include <tcp.h>
00008
00009 #endif /*HTMLVARS_H_*/
```

24.126 NANOL7.h

```
00001 void L7OutByte(char c);
00002 void L7OutHex(uint32_t val, int bytes);
00003 void L7OutString(const char *cp);
00004 void OpenIrq7(int port);
00005 void RunL7StdoutPump();
00006
```

24.127 tide.h

```
00001 class TideCalc {
00002  public:
00003      TideCalc();
00004     float currentTide(time_t now); // returns predicted tide for
00005     // the supplied date and time. The time should always be given in
00006     // the local standard time for the site, not daylight savings time
00007     // output units = feet
00008     char* returnStationID(void); // NOAA station name
00009     long returnStationIDnumber(void); // NOAA station ID number
00010 };
00011
00012
```

24.128 _common/EFFS/STD/src/effs_std.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00022 #ifndef _EFFSSTD_H_
00023 #define _EFFSSTD_H_
00024
00025 /*****************************************************************
00026  * Definitions
00027  *****************************************************************/
00028 /* On-chip Flash NOR */
```

```
00029 #define USE_NOR
00030
00031 /* Drive numbers */
00032 #define NOR_DRV_NUM 0
00033 #define STDRAM_DRV_NUM 1
00034 #define MMC_DRV_NUM 2
00035 #define CFC_DRV_NUM 3
00036 #define HDD_DRV_NUM 3
00037 #define FATRAM_DRV_NUM 4
00038
00039 #define FS_NO_ERROR FS_NOERR
00040
00041 /* Routine definitions */
00042 #include <file/fsf.h>
00043 #include <file/fwerr.h>
00044
00045 /*
00046 ****************************************************************************
00047 *
00048 * "C" Routines
00049 *
00050 ****************************************************************************
00051 */
00052 #ifdef __cplusplus
00053 extern "C"
00054 {
00055 #endif
00056
00057     /*
00058     ****************************************************************************
00059
00060     Start EFFS
00061
00062     Parameters:
00063     deviceNamePtr        - Device name
00064
00065     Return:
00066     None
00067
00068     Notes:
00069     Starts EFFS-STD file system, formats if necessary.
00070
00071     ****************************************************************************
00072     */
00073     void EffsStart(char *deviceNamePtr);
00074
00075     /*
00076     ****************************************************************************
00077
00078     Lists files and directories starting with the current directory
00079
00080     Parameters:
00081     deviceNamePtr        - Device name
00082
00083     Return:
00084     None
00085
00086     Notes:
00087     None
00088
00089     ****************************************************************************
00090     */
00091     void EffsListCurrentDirectory(char *deviceNamePtr);
00092
00093     /*
00094     ****************************************************************************
00095
00096     Display space used, total and bad
00097
00098     Parameters:
00099     deviceNamePtr        - Device name
00100
00101     Return:
00102     None
00103
00104     Notes:
00105     None
00106
00107     ****************************************************************************
00108     */
00109     void EffsDisplayStatistics(char *deviceNamePtr);
00110
00111     /* Format the EFFS Flash file system */
00112     uint8_t EffsFormat();
00113
00114 #ifdef __cplusplus
00115 };
```

```
00116 #endif
00117
00118 #endif /* #ifndef _EFFSSTD_H_ */
```

24.129 Parallax/src/effs_std.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00044 #ifndef _EFFSSTD_H_
00045 #define _EFFSSTD_H_
00046
00047 /*********************************************************************
00048  * Definitions
00049  *********************************************************************/
00050 /* On-chip Flash NOR */
00051 #define USE_NOR
00052
00053 /* Drive numbers */
00054 #define NOR_DRV_NUM 0
00055 #define STDRAM_DRV_NUM 1
00056 #define MMC_DRV_NUM 2
00057 #define CFC_DRV_NUM 3
00058 #define HDD_DRV_NUM 3
00059 #define FATRAM_DRV_NUM 4
00060
00061 #define FS_NO_ERROR FS_NOERR
00062
00063 /* Routine definitions */
00064 #include <file/fsf.h>
00065 #include <file/fwerr.h>
00066
00067 /*
00068 ******************************************************************************
00069 *
00070 * "C" Routines
00071 *
00072 ******************************************************************************
00073 */
00074 #ifdef __cplusplus
00075 extern "C"
00076 {
00077 #endif
00078
00079     /*
00080     ***************************************************************************
00081
00082     Start EFFS
00083
00084     Parameters:
00085     deviceNamePtr        - Device name
00086
00087     Return:
00088     None
00089
00090     Notes:
00091     Starts EFFS-STD file system, formats if necessary.
00092
00093     ***************************************************************************
00094     */
00095     void EffsStart(char *deviceNamePtr);
00096
```

```
00097     /*
00098     ***************************************************************************
00099
00100     Lists files and directories starting with the current directory
00101
00102     Parameters:
00103     deviceNamePtr        - Device name
00104
00105     Return:
00106     None
00107
00108     Notes:
00109     None
00110
00111     ***************************************************************************
00112     */
00113     void EffsListCurrentDirectory(char *deviceNamePtr);
00114
00115     /*
00116     ***************************************************************************
00117
00118     Display space used, total and bad
00119
00120     Parameters:
00121     deviceNamePtr        - Device name
00122
00123     Return:
00124     None
00125
00126     Notes:
00127     None
00128
00129     ***************************************************************************
00130     */
00131     void EffsDisplayStatistics(char *deviceNamePtr);
00132
00133     /* Format the EFFS Flash file system */
00134     uint8_t EffsFormat();
00135
00136 #ifdef __cplusplus
00137 };
00138 #endif
00139
00140 #endif /* #ifndef _EFFSSTD_H_ */
```

24.130 _common/EFFS/FAT/src/effs_time.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 // Set the system time using a Network Time Server
00009 uint32_t SetTimeNTP();
00010
00011 // Set the system time manually
00012 void SetTimeManual(int month, int day, int weekday, int year, int hour, int min, int sec);
00013
00014 // Set the system time using a Real-Time clock
00015 void SetTimeRTC();
00016
00017 void DisplaySystemTime();
00018
00019 // This function is deprecated. Use tzsetchar for proper timezone management
00020 void SetTimeZone(int hour_offset, int isdst) __attribute__((deprecated));
00021 void WasSetTimeZone(int hour_offset, int isdst);
00022
00023 #endif /* _EFFS_TIME_H */
```

24.131 _common/EFFS/STD/src/effs_time.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 // Set the system time using a Network Time Server
00009 uint32_t SetTimeNTP();
```

```
00010
00011 // Set the system time manually
00012 void SetTimeManual(int month, int day, int weekday, int year, int hour, int min, int sec);
00013
00014 // Set the system time using a Real-Time clock
00015 void SetTimeRTC();
00016
00017 void DisplaySystemTime();
00018
00019 // This function is deprecated. Use tzsetchar for proper timezone management
00020 void SetTimeZone(int hour_offset, int isdst) __attribute__((deprecated));
00021 void WasSetTimeZone(int hour_offset, int isdst);
00022
00023 #endif /* _EFFS_TIME_H */
```

24.132 Parallax/src/effs_time.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 #ifndef _EFFS_TIME_H
00028 #define _EFFS_TIME_H
00029
00030 // Set the system time using a Network Time Server
00031 uint32_t SetTimeNTP();
00032
00033 // Set the system time manually
00034 void SetTimeManual(int month, int day, int weekday, int year, int hour, int min, int sec);
00035
00036 // Set the system time using a Real-Time clock
00037 void SetTimeRTC();
00038
00039 void DisplaySystemTime();
00040
00041 // This function is deprecated. Use tzsetchar for proper timezone management
00042 void SetTimeZone(int hour_offset, int isdst) __attribute__((deprecated));
00043 void WasSetTimeZone(int hour_offset, int isdst);
00044
00045 #endif /* _EFFS_TIME_H */
```

24.133 PlatformSpecific/MCF5441X/EffsLoadAppFromFlashCard/src/effs_time.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 // Set the system time using a Network Time Server
00009 uint32_t SetTimeNTP();
00010
00011 // Set the system time manually
00012 void SetTimeManual(int month, int day, int weekday, int year, int hour, int min, int sec);
00013
00014 // Set the system time using a Real-Time clock
00015 void SetTimeRTC();
```

```
00016
00017 void DisplaySystemTime();
00018
00019 // This function is deprecated. Use tzsetchar for proper timezone management
00020 void SetTimeZone(int hour_offset, int isdst) __attribute__((deprecated));
00021 void WasSetTimeZone(int hour_offset, int isdst);
00022
00023 #endif /* _EFFS_TIME_H */
```

24.134 PlatformSpecific/MCF5441X/EffsSDHC/src/effs_time.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 // Set the system time using a Network Time Server
00009 uint32_t SetTimeNTP();
00010
00011 // Set the system time manually
00012 void SetTimeManual(int month, int day, int weekday, int year, int hour, int min, int sec);
00013
00014 // Set the system time using a Real-Time clock
00015 void SetTimeRTC();
00016
00017 void DisplaySystemTime();
00018
00019 // This function is deprecated. Use tzsetchar for proper timezone management
00020 void SetTimeZone(int hour_offset, int isdst) __attribute__((deprecated));
00021 void WasSetTimeZone(int hour_offset, int isdst);
00022
00023 #endif /* _EFFS_TIME_H */
```

24.135 PlatformSpecific/MCF5441X/MOD5441X/EffsMultipleMmc/src/effs_time.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 bool SetTimeNTP();   // Set the system time using a Network Time Server
00009 // Set the system time manually
00010 void SetTimeManual(int month, int day, int weekday, int year, int hour, int min, int sec);
00011 void SetTimeRTC();   // Set the system time using a Real-Time clock
00012
00013 void DisplaySystemTime();
00014
00015 // extern "C"
00016 //{
00017 //    void tzsetchar(char * tzenv);
00018 // }
00019
00020 #endif /* _EFFS_TIME_H */
```

24.136 AM29LV160B.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
```

```
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 ***************************************************************************/
00026
00027 /*-------------------------------------------------------------------------
00028  * EFFS-STD configuration file for Spansion AM29LV160B flash chip.  This file is
00029  * part of an example that allocates 512kB of flash space to the file system,
00030  * and the rest to the application.
00031  *
00032  * Note:  The AM29LV160B is identical to the Spansion S29AL016D.
00033  *
00034  * To modify the amount of space allocated to the file system:
00035  *
00036  * 1. Change the definition in this file:  #define FS_SIZE ( 1024 * 1024 )
00037  * 2. Change the compcode memory address range for the application in your
00038  *    NBEclipse project settings so that the end of the application space does
00039  *    not exceed the start of the file system space.  See the EFFS Programmer's
00040  *    Guide for details, and the header comments in main.cpp of this example
00041  *    on how to make the changes in the NBEclipse project.
00042  * 3. Be sure to add the /nburn/platform/<platform>/original/lib/libStdFFile.a library
00043  *    to your NBEclipse project C/C++ build linker library options.  See the header
00044  *    comments in main.cpp for this example on how to add the library in the
00045  *    NBEclipse project.
00046  *-------------------------------------------------------------------------*/
00047
00048 #ifndef _ONCHIPFLASH_H_
00049 #define _ONCHIPFLASH_H_
00050
00051 #include "file/fsf.h"
00052 #include "basictypes.h"
00053 #include "hal.h"
00054
00058 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00059
00060 #define FLASH_NAME "S29AL016/AMDLV160"
00061
00065 #define FS_FLASHBASE (0xFFC00000)
00066
00125 #define BLOCKSIZE (64 * 1024)    // Use only the 64k sectors
00126 #define SECTORSIZE (16 * 1024)   // 4 sectors per block
00127 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00128
00129 /*
00130  * Specify the total amount of flash memory in the system, and the amount
00131  * allocated to be used by the file system (the rest is used by the
00132  * application).
00133  */
00134 #define FLASH_SIZE \
00135     (2 * 1024 * 1024)   // Size of total flash in the
00136                         //    system, 2MB
00137 #define FS_SIZE \
00138     (512 * 1024)   // Amount allocated to file
00139                    //    system, 512kB
00140 //#define FS_SIZE      (          1024 * 1024 )   // Amount allocated to file
00141 //    system, 1MB
00142 #define FIRST_ADDR \
00143     (FLASH_SIZE - FS_SIZE)   // First file system address to
00144                              //    use in the flash
00145 #define BLOCKSTART \
00146     (2)   // First block where file system
00147           //    data starts (first 2
00148           //    blocks are DESCRIPTORS)
00149
00150 /*
00151  * Descriptor Blocks:
00152  * These blocks contain critical information about the file system, block
00153  * allocation, wear information, and file/directory information.  At least two
00154  * descriptor blocks must be included in the system, which can be erased
00155  * independently.  An optional descriptor write cache may be configured which
00156  * improves the performance of the file system.  Please refer to the EFFS-STD
00157  * implementation guide for additional information.
00158  */
00159 #define DESCSIZE (8 * 1024)   // Size of one descriptor
00160 #define DESCBLOCKSTART (0)    // Position of first descriptor
00161 #define DESCBLOCKEND (1)      // Position of last descriptor
00162 #define DESCCACHE (2048)
00163
00164 #endif /* _ONCHIPFLASH_H_ */
```

24.137 AT49BV163D.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 /*-----------------------------------------------------------------------------
00028  * EFFS-STD configuration file for Atmel AT49BV163D flash chip.  This file is
00029  * part of an example that allocates 512kB of flash space to the file system,
00030  * and the rest to the application.
00031  *
00032  * To modify the amount of space allocated to the file system:
00033  *
00034  * 1. Change the definition in this file:  #define FS_SIZE ( 1024 * 1024 )
00035  * 2. Change the compcode memory address range for the application in your
00036  *    NBEclipse project settings so that the end of the application space does
00037  *    not exceed the start of the file system space.  See the EFFS Programmer's
00038  *    Guide for details, and the header comments in main.cpp of this example
00039  *    on how to make the changes in the NBEclipse project.
00040  * 3. Be sure to add the /nburn/platform/<platform>/original/lib/libStdFFile.a library
00041  *    to your NBEclipse project C/C++ build linker library options.  See the header
00042  *    comments in main.cpp for this example on how to add the library in the
00043  *    NBEclipse project.
00044  *----------------------------------------------------------------------------*/
00045
00046 #ifndef _ONCHIPFLASH_H_
00047 #define _ONCHIPFLASH_H_
00048
00049 #include "basictypes.h"
00050 #include "hal.h"
00051 #include "file/fsf.h"
00052
00056 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00057
00058 #define FLASH_NAME "AT49BV163"
00059
00063 #define FS_FLASHBASE (0xFFC00000)
00064
00123 #define BLOCKSIZE (64 * 1024)     // Use only the 64k sectors
00124 #define SECTORSIZE (16 * 1024)    // 4 sectors per block
00125 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00126
00127 /*
00128  * Specify the total amount of flash memory in the system, and the amount
00129  * allocated to be used by the file system (the rest is used by the
00130  * application).
00131  */
00132 #define FLASH_SIZE \
00133     (2 * 1024 * 1024)   // Size of total flash in the
00134                         //    system, 2MB
00135 #define FS_SIZE \
00136     (512 * 1024)   // Amount allocated to file
00137                    //    system, 512kB
00138 #define FIRST_ADDR \
00139     (FLASH_SIZE - FS_SIZE)   // First file system address to
00140                              //    use in the flash
00141 #define BLOCKSTART \
00142     (2)   // First block where file system
00143           //    data starts (first 2
00144           //    blocks are DESCRIPTORS)
00145
00146 /*
00147  * Descriptor Blocks:
00148  * These blocks contain critical information about the file system, block
00149  * allocation, wear information, and file/directory information.  At least two
```

```
00150  * descriptor blocks must be included in the system, which can be erased
00151  * independently.  An optional descriptor write cache may be configured which
00152  * improves the performance of the file system.  Please refer to the EFFS-STD
00153  * implementation guide for additional information.
00154  */
00155 #define DESCSIZE (8 * 1024)    // Size of one descriptor
00156 #define DESCBLOCKSTART (0)     // Position of first descriptor
00157 #define DESCBLOCKEND (1)       // Position of last descriptor
00158 #define DESCCACHE (2048)
00159
00160 #endif /* _ONCHIPFLASH_H_ */
```

24.138 MCF5282Flash.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 /*---------------------------------------------------------------------------
00028  * EFFS-STD configuration file for MOD5282 processor flash memory.  This file is part
00029  * of an example that allocates 64kB of flash space to the file system, and the
00030  * rest to the application.
00031  *
00032  * To modify the amount of space allocated to the file system:
00033  *
00034  * 1. Change the definition in this file:  #define FS_SIZE ( 64 * 1024 )
00035  * 2. Change the compcode memory address range for the application in your
00036  *    NBEclipse project settings so that the end of the application space does
00037  *    not exceed the start of the file system space.  See the EFFS Programmer's
00038  *    Guide for details, and the header comments in main.cpp of this example
00039  *    on how to make the changes in the NBEclipse project.
00040  * 3. Be sure to add the /nburn/platform/<platform>/original/lib/libStdFFile.a library
00041  *    to your NBEclipse project C/C++ build linker library options.  See the header
00042  *    comments in main.cpp for this example on how to add the library in the
00043  *    NBEclipse project.
00044  *---------------------------------------------------------------------------*/
00045
00046 #ifndef _ONCHIPFLASH_H_
00047 #define _ONCHIPFLASH_H_
00048
00049 #include "basictypes.h"
00050 #include "hal.h"
00051 #include "file/fsf.h"
00052
00056 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00057
00058 #define FLASH_NAME "MCF5282"   // Processor name, not module name
00059
00063 #define FS_FLASHBASE (0xFFC00000)
00064
00120 #define BLOCKSIZE (4 * 1024)   // All sectors are 4KB
00121 #define SECTORSIZE (512)       // 8 sectors per block
00122 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00123
00124 /*
00125  * Specify the total amount of flash memory in the system, and the amount
00126  * allocated to be used by the file system (the rest is used by the
00127  * application).
00128  */
00129 #define FLASH_SIZE \
00130     (512 * 1024)   // Size of total flash in the
00131                    //    system, 512kB
00132 #define FS_SIZE \
```

```
00133     (64 * 1024)   // Amount allocated to file
00134                   //    system, 64kB
00135 #define FIRST_ADDR \
00136     (FLASH_SIZE - FS_SIZE)   // First file system address to
00137                              //    use in the flash
00138 #define BLOCKSTART \
00139     (2)   // First block where file system
00140           //    data starts (first 2
00141           //    blocks are DESCRIPTORS)
00142
00143 /*
00144  * Descriptor Blocks:
00145  * These blocks contain critical information about the file system, block
00146  * allocation, wear information, and file/directory information.  At least two
00147  * descriptor blocks must be included in the system, which can be erased
00148  * independently.  An optional descriptor write cache may be configured which
00149  * improves the performance of the file system.  Please refer to the EFFS-STD
00150  * implementation guide for additional information.
00151  */
00152 #define DESCSIZE (4 * 1024)   // Size of one descriptor
00153 #define DESCBLOCKSTART (0)    // Position of first descriptor
00154 #define DESCBLOCKEND (1)      // Position of last descriptor
00155 #define DESCCACHE (1024)
00156
00157 #endif /* _ONCHIPFLASH_H_ */
```

24.139 _common/EFFS/STD/src/flashChip/MX25L6406E.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-------------------------------------------------------------------------
00006  * EFFS-STD configuration file for 8MB SPI flash chip used on the NANO54415 and
00007  * SB800EX.
00008  *
00009  * Macronix MX25L6406 8MB SPI Flash chip.
00010  *-------------------------------------------------------------------------*/
00011
00012 #ifndef _ONCHIPFLASH_H_
00013 #define _ONCHIPFLASH_H_
00014
00015 #include "basictypes.h"
00016 #include "hal.h"
00017 #include "file/fsf.h"
00018
00019 // Function implemented
00020 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00021
00022 #define FLASH_NAME "MX25L6406E"
00023
00075 //#define FS_FLASHBASE   ( 0x00000000 )
00076
00077 /*
00078  * CHANGES TO COMPCODE FLAGS
00079  * In NBEclipse, or your command line makefile, change the following line
00080  * so the application will only occupy the specified application space.
00081  * The first parameter is the start of application space, and the second
00082  * is the address just below the file system space.
00083  *
00084  *    COMPCODEFLAGS = 0x04000 0x800000      // Original
00085  *
00086  *    COMPCODEFLAGS = 0x04000 0x700000      // Space for file system
00087  */
00088
00089 #define BLOCKSIZE (4 * 1024)   // Flash physical "sector" size, smallest erasable unit
00090 #define SECTORSIZE (512)       // Flash sector size in bytes, smallest usable unit
00091 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00092
00093 /*
00094  * Specify the total amount of flash memory in the system, and the amount
00095  * allocated to be used by the file system (the rest is used by the
00096  * application).
00097  */
00098 /* Example for a 64k file system */
00099 //#define FLASH_SIZE   (        8*1024 * 1024 )   // Total flash space
00100 //#define FS_SIZE      (            64 * 1024 )   // Amount of Flash allocated to file system
00101 //#define FIRST_ADDR   ( FLASH_SIZE - FS_SIZE )   // First Flash file system address
00102 //#define BLOCKSTART   (                    2 )   // First block where file system data starts (first
      2 blocks are DESCRIPTORS)
00103
00104 /* Example for a 1MB file system */
00105 #define FLASH_SIZE (8 * 1024 * 1024)        // Total flash space
00106 #define FS_SIZE (1 * 1024 * 1024)           // Amount of Flash allocated to file system
00107 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE)   // First Flash file system address
```

```
00108 #define BLOCKSTART (16)                     // First block where file system data starts, after blocks
      reserved for Descriptors
00109
00110 /*
00111  * Descriptor Blocks:
00112  * These blocks contain critical information about the file system, block
00113  * allocation, wear information, and file/directory information.  At least two
00114  * descriptor blocks must be included in the system, which can be erased
00115  * independently.  An optional descriptor write cache may be configured which
00116  * improves the performance of the file system.  Please refer to the EFFS-STD
00117  * implementation guide for additional information.
00118  */
00119 /* Example for a 64k file system */
00120 //#define DESCSIZE          ( 4 * 1024 )   // Size of one descriptor
00121 //#define DESCBLOCKSTART    (        0 )   // Position of first descriptor
00122 //#define DESCBLOCKEND      (        1 )   // Position of last descriptor
00123 //#define DESCCACHE         (     1024 )
00124
00125 /* Example for a 1MB file system */
00126 #define DESCSIZE (64 * 1024)    // Size of one descriptor, 8 BLOCKS
00127 #define DESCBLOCKSTART (0)      // Position of first descriptor
00128 #define DESCBLOCKEND (1)        // Position of last descriptor
00129 #define DESCCACHE (1024)
00130
00131 #endif /* _ONCHIPFLASH_H_ */
```

24.140 Parallax/src/flashChip/MX25L6406E.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 /*---------------------------------------------------------------------------
00028  * EFFS-STD configuration file for 8MB SPI flash chip used on the NANO54415 and
00029  * SB800EX.
00030  *
00031  * To modify the amount of space allocated to the file system:
00032  *
00033  * 1. Change the definition in this file:  #define FS_SIZE ( 64 * 1024 )
00034  * 2. Change the compcode memory address range for the application in your
00035  *    NBEclipse project settings (or makefile) so that the end of the
00036  *    application space does not exceed the start of the file system space.
00037  *    Refer to the EFFS Programming Guide section of the manual for more details.
00038  * 3. Be sure to add the \Nburn\lib\StdFFile.a library to your NBEclipse project
00039  *    C/C++ build linker library options.  See the header comments in main.cpp
00040  *    for this example on how to add the library in the NBEclipse project.
00041  *---------------------------------------------------------------------------*/
00042
00043 #ifndef _ONCHIPFLASH_H_
00044 #define _ONCHIPFLASH_H_
00045
00046 #include "basictypes.h"
00047 #include "hal.h"
00048 #include "file/fsf.h"
00049
00053 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00054
00055 #define FLASH_NAME "MX25L6406E"
00056
00108 //#define FS_FLASHBASE    ( 0x00000000 )
00109
00110 /*
00111  * CHANGES TO COMPCODE FLAGS
```

```
00112  * In NBEclipse, or your command line makefile, change the following line
00113  * so the application will only occupy the specified application space.
00114  * The first parameter is the start of application space, and the second
00115  * is the address just below the file system space.
00116  *
00117  *    COMPCODEFLAGS = 0x04000 0x800000      // Original
00118  *
00119  *    COMPCODEFLAGS = 0x04000 0x700000      // space for file system
00120  */
00121
00122 #define BLOCKSIZE (4 * 1024)   // All sectors are 4KB
00123 #define SECTORSIZE (512)       // 8 sectors per block
00124 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00125
00126 /*
00127  * Specify the total amount of flash memory in the system, and the amount
00128  * allocated to be used by the file system (the rest is used by the
00129  * application).
00130  */
00131 /* Example for a 64k file system */
00132 //#define FLASH_SIZE   (        8*1024 * 1024 )   // Total flash space
00133 //#define FS_SIZE      (           64 * 1024 )   // Amount of Flash allocated to file system
00134 //#define FIRST_ADDR   ( FLASH_SIZE - FS_SIZE )   // First Flash file system address
00135 //#define BLOCKSTART   (                    2 )   // First block where file system data starts (first
      2 blocks are DESCRIPTORS)
00136
00137 /* Example for a 1MB file system */
00138 #define FLASH_SIZE (8 * 1024 * 1024)        // Total flash space
00139 #define FS_SIZE (1 * 1024 * 1024)           // Amount of Flash allocated to file system
00140 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE)   // First Flash file system address
00141 #define BLOCKSTART (16)                     // First block where file system data starts, after blocks
      reserved for Descriptors
00142
00143 /*
00144  * Descriptor Blocks:
00145  * These blocks contain critical information about the file system, block
00146  * allocation, wear information, and file/directory information.  At least two
00147  * descriptor blocks must be included in the system, which can be erased
00148  * independently.  An optional descriptor write cache may be configured which
00149  * improves the performance of the file system.  Please refer to the EFFS-STD
00150  * implementation guide for additional information.
00151  */
00152 /* Example for a 64k file system */
00153 //#define DESCSIZE         ( 4 * 1024 )   // Size of one descriptor
00154 //#define DESCBLOCKSTART   (        0 )   // Position of first descriptor
00155 //#define DESCBLOCKEND     (        1 )   // Position of last descriptor
00156 //#define DESCCACHE        (     1024 )
00157
00158 /* Example for a 1MB file system */
00159 #define DESCSIZE (64 * 1024)   // Size of one descriptor, 8 BLOCKS
00160 #define DESCBLOCKSTART (0)     // Position of first descriptor
00161 #define DESCBLOCKEND (1)       // Position of last descriptor
00162 #define DESCCACHE (1024)
00163
00164 #endif /* _ONCHIPFLASH_H_ */
```

24.141 _common/EFFS/STD/src/flashChip/MX29GL256F.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-------------------------------------------------------------------
00006  * EFFS-STD configuration file for 32MB Flash chip used on the MOD54415
00007  * and MOD54417.
00008  *
00009  * Macronix MX29GL256F 32MB Flash Chip.
00010  *-------------------------------------------------------------------*/
00011
00012 #ifndef _ONCHIPFLASH_H_
00013 #define _ONCHIPFLASH_H_
00014
00015 #include "file/fsf.h"
00016 #include "basictypes.h"
00017 #include "hal.h"
00018
00019 #define FLASH_NAME "MX29GL256"
00020
00021 // functions implemented
00022 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00023
00024 // Start of Flash memory base address
00025 #define FS_FLASHBASE (0xC0000000)
00026
00027 /*
```

```
00028  * BLOCKSIZE
00029  * This defines the size of the blocks to be used in the file storage area.
00030  * This must be an erasable unit of the flash chip. All blocks in the file
00031  * storage area must be the same size. This maybe different from the DESCSIZE
00032  * where the flash chip has different size erasable units available.
00033  *
00034  * SECTORSIZE
00035  * This defines the sector size. Each block is divided into a number of sectors.
00036  * This number is the smallest usable unit in the system and thus represents the
00037  * minimum file storage area. For best usage of the flash blocks the sector size
00038  * should always be a power of 2. For more information see sector section below.
00039  *
00040  * SECTORPERBLOCK
00041  * This defines the number of sectors in a block. It must always be true that:
00042  * SECTORPERBLOCK = BLOCKSIZE/SECTORSIZE
00043  *
00044  *
00045  * The memory map below is for a MOD5441x with a 32MB bottom boot block flash.
00046  * This example will allocate 1.3MB for the file system.
00047  * - 2 blocks are reserved for the file descriptors
00048  * - 1 block is reserved as a free sector for the file system to operate
00049  *
00050  * A total of 10 blocks are allocated for the file system. Subtracting the 3 reserved
00051  * as described above, the total file system free space is: 7 * 128k = 896k (917,504 bytes).
00052  *
00053  * Macronix MX29GL256F, 256 blocks of 128KB each, Total of 32MBytes
00054  *
00055  * Note: this header file is also valid for the Spansion/Cypress 32MB flash
00056  *
00057  *                            Address
00058  *                           ----------
00059  *    ---------------------- C1FFF FFFF (End of flash space)
00060  *    |  File System Data  |
00061  *    |  1MB               |
00062  *    |  128K x 8 Blocks   |
00063  *    |--------------------| C1F0 0000 (Start of File System Data)
00064  *    |  DESC BLOCK 0/1    |
00065  *    | 128K x 2 Blocks    |
00066  *    |--------------------| C1EC 0000 (Start of File System)
00067  *    |                    |
00068  *    | Application        |
00069  *    | 31.232MB           |
00070  *    | 128K x 244 Blocks  |
00071  *    |                    |
00072  *    |--------------------| C004 0000
00073  *    | 128K User Params   |
00074  *    |--------------------| C002 0000
00075  *    | 128K System Params |
00076  *    |--------------------| C000 0000 (Start of Flash space)
00077  *
00078  *
00079  * CHANGES TO COMPCODE FLAGS
00080  * In NBEclipse, or your command line makefile, change the following line
00081  * so the application will only occupy the specified application space.
00082  * The first parameter is the start of application space, and the second
00083  * is the address just below the file system space.
00084  *
00085  *    COMPCODEFLAGS = 0xC0040000 0xC2000000     // Original
00086  *    COMPCODEFLAGS = 0xC0040000 0xC1F00000     // Space for file system, 1MB
00087  *
00088  */
00089
00090 /* WARNING: These settings are for MX29GL256F bottom boot block flash
00091  * components used on the Mod54415
00092  */
00093 #define BLOCKSIZE (128 * 1024)   // Flash physical "sector" size, smallest erasable unit of flash
00094 #define SECTORSIZE (1 * 1024)    // Flash sector size in bytes, smallest usable unit of flash
00095 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00096
00097 /*
00098  * Specify the total amount of flash memory in the system, and the amount
00099  * allocated to be used by the file system (the rest is used by the application.
00100  */
00101 #define FLASH_SIZE (32 * 1024 * 1024)       // Total physical flash, 32MB
00102
00103 /*
00104  * Number of sectors allocated to file system. Includes 2 Descriptor Blocks, and 1 block reserved
00105  * for file system operation. This means the usable storage space will be less than the amount
00106  * calculated below.
00107  */
00108 #define FS_SIZE (1280 * 1024)
00109
00110 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE)   // first file system address
00111 #define BLOCKSTART 2   // Starting file system data block. First 2 blocks are DESCRIPTORS
00112
00113 /*
00114  * Descriptor Blocks:
```

```
00115  * These blocks contain critical information about the file system, block allocation,
00116  * wear information and file/directory information. At least two descriptor blocks
00117  * must be included in the system, which can be erased independently. An optional
00118  * descriptor write cache may be configured which improves the performance of the
00119  * file system. Please refer to the EFFS-STD implementation guide for additional
00120  * information.
00121  */
00122 #define DESCSIZE (128 * 1024)   // size of one descriptor
00123 #define DESCBLOCKSTART 0        // position of first descriptor
00124 #define DESCBLOCKEND 1          // position of last descriptor
00125 #define DESCCACHE 2048
00126
00127 #endif /* _ONCHIPFLASH_H_ */
```

24.142 Parallax/src/flashChip/MX29GL256F.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 /*-------------------------------------------------------------------
00028  * EFFS-STD configuration file for Macronix MX29GL256F Flash Chip.
00029  * This file is part of an example that allocats 512K of flash space
00030  * to the file system, and the rest to the application.
00031  *-------------------------------------------------------------------*/
00032
00033 #ifndef _ONCHIPFLASH_H_
00034 #define _ONCHIPFLASH_H_
00035
00036 #include "file/fsf.h"
00037 #include "basictypes.h"
00038 #include "hal.h"
00039
00040 #define FLASH_NAME "MX29GL256"
00041
00042 /*functions implemented*/
00043 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00044
00045 // Start of Flash memory base address
00046 #define FS_FLASHBASE (0xC0000000)
00047
00048 /*
00049  * BLOCKSIZE
00050  * This defines the size of the blocks to be used in the file storage area.
00051  * This must be an erasable unit of the flash chip. All blocks in the file
00052  * storage area must be the same size. This maybe different from the DESCSIZE
00053  * where the flash chip has different size erasable units available.
00054  *
00055  * SECTORSIZE
00056  * This defines the sector size. Each block is divided into a number of sectors.
00057  * This number is the smallest usable unit in the system and thus represents the
00058  * minimum file storage area. For best usage of the flash blocks the sector size
00059  * should always be a power of 2. For more information see sector section below.
00060  *
00061  * SECTORPERBLOCK
00062  * This defines the number of sectors in a block. It must always be true that:
00063  * SECTORPERBLOCK = BLOCKSIZE/SECTORSIZE
00064  *
00065  *
00066  * The memory map below is for a MOD5441x with a 32MB bottom boot block flash.
00067  * This example will allocate 1.3MB for the file system.
00068  * - 2 blocks are reserved for the file descriptors
00069  * - 1 block is reserved as a free sector for the file system to operate
```

```
00070  *
00071  * A total of 10 blocks are allocated for the file system. Subtracting the 3 reserved
00072  * as described above, the total file system free space is: 7 * 128k = 896k (917,504 bytes).
00073  *
00074  * Macronix MX29GL256F, 256 blocks of 128KB each, Total of 32MBytes
00075  *
00076  * Note: this header file is also valid for the Spansion/Cypress 32MB flash
00077  *
00078  *                            Address
00079  *                           ----------
00080  *    ---------------------- C1FFF FFFF (End of flash space)
00081  *    |  File System Data  |
00082  *    |  1MB               |
00083  *    |  128K x 8 Blocks   |
00084  *    |--------------------| C1F0 0000 (Start of File System Data)
00085  *    |  DESC BLOCK 0/1    |
00086  *    | 128K x 2 Blocks    |
00087  *    |--------------------| C1EC 0000 (Start of File System)
00088  *    |                    |
00089  *    | Application        |
00090  *    | 31.232MB           |
00091  *    | 128K x 244 Blocks  |
00092  *    |                    |
00093  *    |--------------------| C004 0000
00094  *    | 128K User Params   |
00095  *    |--------------------| C002 0000
00096  *    | 128K System Params |
00097  *    |--------------------| C000 0000 (Start of Flash space)
00098  *
00099  *
00100  * CHANGES TO COMPCODE FLAGS
00101  * In NBEclipse, or your command line makefile, change the following line
00102  * so the application will only occupy the specified application space.
00103  * The first parameter is the start of application space, and the second
00104  * is the address just below the file system space.
00105  *
00106  *    COMPCODEFLAGS = 0xC0040000 0xC1FC0000
00107  *
00108  * If using NBEclipse:
00109  * - Right-click on the project and select "Properties"
00110  * - Select "NetBurner" in the left side of the dialog box
00111  * - Verify the Platform is set to Mod5234, then check the "Use Custom Platform Settings" checkbox
00112  * - Modify the "Compcode Memory Range" to the above values
00113  *
00114  *
00115  * If using NBEclipse, you will also need to tell the linker to include the
00116  * /nburn/platform/<platform>/original/lib/libStdFFile.a library. To do this right-click on your
00117  * project, select properties, GNU Linker, then add the library.
00118  *
00119  */
00120
00121 /* WARNING: These settings are for MX29GL256F bottom boot block flash
00122  * components used on the Mod54415
00123  */
00124 #define BLOCKSIZE (128 * 1024)   // flash physical "sector" size
00125 #define SECTORSIZE (1 * 1024)    // file system sectors per BLOCK
00126 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00127
00128 /*
00129  * Specify the total amount of flash memory in the system, and the amount
00130  * allocated to be used by the file system (the rest is used by the
00131  * application.
00132  */
00133 #define FLASH_SIZE (32 * 1024 * 1024)   // size of total flash in the system, 32MB
00134 #define FS_SIZE \
00135     (1280 * 1024)                          // amount allocated to file system: 2 Desc. plus 1MB for
      data (8 x 128k)
00136                                            // note that 1 block of file data will be reserved for the
      file system
00137 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE)   // first file system address to use in the flash
00138 #define BLOCKSTART \
00139     2   // first block where file system data starts
00140         // (first 2 blocks are DESCRIPTORS)
00141
00142 /*
00143  * Descriptor Blocks:
00144  * These blocks contain critical information about the file system, block allocation,
00145  * wear information and file/directory information. At least two descriptor blocks
00146  * must be included in the system, which can be erased independently. An optional
00147  * descriptor write cache may be configured which improves the performance of the
00148  * file system. Please refer to the EFFS-STD implementation guide for additional
00149  * information.
00150  */
00151 #define DESCSIZE (128 * 1024)   // size of one descriptor
00152 #define DESCBLOCKSTART 0        // position of first descriptor
00153 #define DESCBLOCKEND 1          // position of last descriptor
00154 #define DESCCACHE 2048
```

```
00155
00156 #endif /* _ONCHIPFLASH_H_ */
```

24.143 S29GL032.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 /*-----------------------------------------------------------------------------
00028  * EFFS-STD configuration file for Spansion S29GL032A 4MByte flash chip.  This file is
00029  * part of an example that allocates 1MB of flash space to the file system,
00030  * and the rest to the application.
00031  *
00032  * To modify the amount of space allocated to the file system:
00033  *
00034  * 1. Change the definition in this file:  #define FS_SIZE ( 1024 * 1024 )
00035  * 2. Change the compcode memory address range for the application in your
00036  *    NBEclipse project settings so that the end of the application space does
00037  *    not exceed the start of the file system space.  See the EFFS Programmer's
00038  *    Guide for details, and the header comments in main.cpp of this example
00039  *    on how to make the changes in the NBEclipse project.
00040  * 3. Be sure to add the /nburn/platform/<platform>/original/lib/libStdFFile.a library
00041  *    to your NBEclipse project C/C++ build linker library options.  See the header
00042  *    comments in main.cpp for this example on how to add the library in the
00043  *    NBEclipse project.
00044  *----------------------------------------------------------------------------*/
00045
00046 #ifndef _ONCHIPFLASH_H_
00047 #define _ONCHIPFLASH_H_
00048
00049 #include "file/fsf.h"
00050 #include "basictypes.h"
00051 #include "hal.h"
00052
00053 #define FLASH_NAME "S29GL032A"
00054
00055 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00056
00057 /* Start of flash memory base address */
00058 #define FS_FLASHBASE (0xFF800000)
00059
00115 #define BLOCKSIZE (64 * 1024)    // Use only the 64k sectors
00116 #define SECTORSIZE (16 * 1024)   // 4 sectors per block
00117 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00118
00119 /*
00120  * Specify the total amount of flash memory in the system, and the amount
00121  * allocated to be used by the file system (the rest is used by the
00122  * application).
00123  */
00124 #define FLASH_SIZE (4 * 1024 * 1024)        // Size of total flash in the system, 4MB
00125 #define FS_SIZE (1024 * 1024)               // Amount allocated to file system, 1MB
00126 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE)   // First file system address to use in the flash
00127 #define BLOCKSTART (2)                      // First block where file system data starts (first 2
      blocks are DESCRIPTORS)
00128
00129 /*
00130  * Descriptor Blocks:
00131  * These blocks contain critical information about the file system, block
00132  * allocation, wear information, and file/directory information.  At least two
00133  * descriptor blocks must be included in the system, which can be erased
00134  * independently.  An optional descriptor write cache may be configured which
```

```
00135  * improves the performance of the file system.  Please refer to the EFFS-STD
00136  * implementation guide for additional information.
00137  */
00138 #define DESCSIZE (8 * 1024)    // Size of one descriptor
00139 #define DESCBLOCKSTART (0)     // Position of first descriptor
00140 #define DESCBLOCKEND (1)       // Position of last descriptor
00141 #define DESCCACHE (2048)
00142
00143 #endif /* _ONCHIPFLASH_H_ */
```

24.144 same70q21.h File Reference

```
#include <stdint.h>
#include <cm_core_config.h>
#include <core_cm7.h>
#include "system_same70.h"
#include <same70q21_sim.h>
```

Macros

- #define _U_(x) x ## U
- #define _L_(x) x ## L
- #define _UL_(x) x ## UL
- #define **ID_SUPC** (0)

  Supply Controller (SUPC)
- #define **ID_RSTC** (1)

  Reset Controller (RSTC)
- #define **ID_RTC** (2)

  Real Time Clock (RTC)
- #define **ID_RTT** (3)

  Real Time Timer (RTT)
- #define **ID_WDT** (4)

  Watchdog Timer (WDT)
- #define **ID_PMC** (5)

  Power Management Controller (PMC)
- #define **ID_EFC** (6)

  Enhanced Embedded Flash Controller (EFC)
- #define **ID_UART0** (7)

  UART 0 (UART0)
- #define **ID_UART1** (8)

  UART 1 (UART1)
- #define **ID_SMC** (9)

  Static Memory Controller (SMC)
- #define **ID_PIOA** (10)

  Parallel I/O Controller A (PIOA)
- #define **ID_PIOB** (11)

  Parallel I/O Controller B (PIOB)
- #define **ID_PIOC** (12)

  Parallel I/O Controller C (PIOC)
- #define **ID_USART0** (13)

  USART 0 (USART0)
- #define **ID_USART1** (14)

  USART 1 (USART1)
- #define **ID_USART2** (15)

USART 2 (USART2)

- #define **ID_PIOD** (16)

  Parallel I/O Controller D (PIOD)
- #define **ID_PIOE** (17)

  Parallel I/O Controller E (PIOE)
- #define **ID_HSMCI** (18)

  Multimedia Card Interface (HSMCI)
- #define **ID_TWIHS0** (19)

  Two Wire Interface 0 HS (TWIHS0)
- #define **ID_TWIHS1** (20)

  Two Wire Interface 1 HS (TWIHS1)
- #define **ID_SPI0** (21)

  Serial Peripheral Interface 0 (SPI0)
- #define **ID_SSC** (22)

  Synchronous Serial Controller (SSC)
- #define **ID_TC0** (23)

  Timer/Counter 0 (TC0)
- #define **ID_TC1** (24)

  Timer/Counter 1 (TC1)
- #define **ID_TC2** (25)

  Timer/Counter 2 (TC2)
- #define **ID_TC3** (26)

  Timer/Counter 3 (TC3)
- #define **ID_TC4** (27)

  Timer/Counter 4 (TC4)
- #define **ID_TC5** (28)

  Timer/Counter 5 (TC5)
- #define **ID_AFEC0** (29)

  Analog Front End 0 (AFEC0)
- #define **ID_DACC** (30)

  Digital To Analog Converter (DACC)
- #define **ID_PWM0** (31)

  Pulse Width Modulation 0 (PWM0)
- #define **ID_ICM** (32)

  Integrity Check Monitor (ICM)
- #define **ID_ACC** (33)

  Analog Comparator (ACC)
- #define **ID_USBHS** (34)

  USB Host / Device Controller (USBHS)
- #define **ID_MCAN0** (35)

  MCAN Controller 0 (MCAN0)
- #define **ID_MCAN1** (37)

  MCAN Controller 1 (MCAN1)
- #define **ID_GMAC** (39)

  Ethernet MAC (GMAC)
- #define **ID_AFEC1** (40)

  Analog Front End 1 (AFEC1)
- #define **ID_TWIHS2** (41)

  Two Wire Interface 2 HS (TWIHS2)
- #define **ID_SPI1** (42)

  Serial Peripheral Interface 1 (SPI1)

- #define **ID_QSPI** (43)

  Quad I/O Serial Peripheral Interface (QSPI)
- #define **ID_UART2** (44)

  UART 2 (UART2)
- #define **ID_UART3** (45)

  UART 3 (UART3)
- #define **ID_UART4** (46)

  UART 4 (UART4)
- #define **ID_TC6** (47)

  Timer/Counter 6 (TC6)
- #define **ID_TC7** (48)

  Timer/Counter 7 (TC7)
- #define **ID_TC8** (49)

  Timer/Counter 8 (TC8)
- #define **ID_TC9** (50)

  Timer/Counter 9 (TC9)
- #define **ID_TC10** (51)

  Timer/Counter 10 (TC10)
- #define **ID_TC11** (52)

  Timer/Counter 11 (TC11)
- #define **ID_AES** (56)

  AES (AES)
- #define **ID_TRNG** (57)

  True Random Generator (TRNG)
- #define **ID_XDMAC** (58)

  DMA (XDMAC)
- #define **ID_ISI** (59)

  Camera Interface (ISI)
- #define **ID_PWM1** (60)

  Pulse Width Modulation 1 (PWM1)
- #define **ID_SDRAMC** (62)

  SDRAM Controller (SDRAMC)
- #define **ID_RSWDT** (63)

  Reinforced Secure Watchdog Timer (RSWDT)
- #define **ID_PERIPH_COUNT** (64)

  Number of peripheral IDs.
- #define QSPIMEM_ADDR (0x80000000u)
- #define AXIMX_ADDR (0xA0000000u)
- #define ITCM_ADDR (0x00000000u)
- #define IFLASH_ADDR (0x00400000u)
- #define IROM_ADDR (0x00800000u)
- #define DTCM_ADDR (0x20000000u)
- #define IRAM_ADDR (0x20400000u)
- #define EBI_CS0_ADDR (0x60000000u)
- #define EBI_CS1_ADDR (0x61000000u)
- #define EBI_CS2_ADDR (0x62000000u)
- #define EBI_CS3_ADDR (0x63000000u)
- #define SDRAM_CS_ADDR (0x70000000u)
- #define **CHIP_FREQ_FWS_0** (20000000UL)

  Maximum operating frequency when FWS is 0.
- #define **CHIP_FREQ_FWS_1** (40000000UL)

Maximum operating frequency when FWS is 1.

- #define **CHIP_FREQ_FWS_2** (60000000UL)

  Maximum operating frequency when FWS is 2.

- #define **CHIP_FREQ_FWS_3** (80000000UL)

  Maximum operating frequency when FWS is 3.

- #define **CHIP_FREQ_FWS_4** (100000000UL)

  Maximum operating frequency when FWS is 4.

- #define **CHIP_FREQ_FWS_5** (123000000UL)

  Maximum operating frequency when FWS is 5.

Typedefs

- typedef volatile const uint32_t RoReg
- typedef volatile const uint16_t RoReg16
- typedef volatile const uint8_t RoReg8
- typedef volatile uint32_t WoReg
- typedef volatile uint16_t WoReg16
- typedef volatile uint32_t WoReg8
- typedef volatile uint32_t RwReg
- typedef volatile uint16_t RwReg16
- typedef volatile uint8_t RwReg8

24.144.1 Detailed Description

24.144.2 Macro Definition Documentation

24.144.2.1 _L_

```
#define _L_(
              x ) x ## L
```

C code: Long integer literal constant value

24.144.2.2 _U_

```
#define _U_(
             x ) x ## U
```

C code: Unsigned integer literal constant value

24.144.2.3 _UL_

```
#define _UL_(
             x ) x ## UL
```

C code: Unsigned Long integer literal constant value

24.144.2.4 AXIMX_ADDR

```
#define AXIMX_ADDR (0xA0000000u)
```

AXI Bus Matrix base address

24.144.2.5 DTCM_ADDR

```
#define DTCM_ADDR (0x20000000u)
```

Data Tightly Coupled Memory base address

24.144.2.6 EBI_CS0_ADDR

```
#define EBI_CS0_ADDR (0x60000000u)
```

EBI Chip Select 0 base address

24.144.2.7 EBI_CS1_ADDR

```
#define EBI_CS1_ADDR (0x61000000u)
```

EBI Chip Select 1 base address

24.144.2.8 EBI_CS2_ADDR

```
#define EBI_CS2_ADDR (0x62000000u)
```

EBI Chip Select 2 base address

24.144.2.9 EBI_CS3_ADDR

```
#define EBI_CS3_ADDR (0x63000000u)
```

EBI Chip Select 3 base address

24.144.2.10 IFLASH_ADDR

```
#define IFLASH_ADDR (0x00400000u)
```

Internal Flash base address

24.144.2.11 IRAM_ADDR

```
#define IRAM_ADDR (0x20400000u)
```

Internal RAM base address

24.144.2.12 IROM_ADDR

```
#define IROM_ADDR (0x00800000u)
```

Internal ROM base address

24.144.2.13 ITCM_ADDR

```
#define ITCM_ADDR (0x00000000u)
```

Instruction Tightly Coupled Memory base address

24.144.2.14 QSPIMEM_ADDR

```
#define QSPIMEM_ADDR (0x80000000u)
```
QSPI Memory base address

24.144.2.15 SDRAM_CS_ADDR

```
#define SDRAM_CS_ADDR (0x70000000u)
```
SDRAM Chip Select base address

24.144.3 Typedef Documentation

24.144.3.1 RoReg

```
typedef volatile const uint32_t RoReg
```
Read only 32-bit register (volatile const unsigned int)

24.144.3.2 RoReg16

```
typedef volatile const uint16_t RoReg16
```
Read only 16-bit register (volatile const unsigned int)

24.144.3.3 RoReg8

```
typedef volatile const uint8_t RoReg8
```
Read only 8-bit register (volatile const unsigned int)

24.144.3.4 RwReg

```
typedef volatile uint32_t RwReg
```
Read-Write 32-bit register (volatile unsigned int)

24.144.3.5 RwReg16

```
typedef volatile uint16_t RwReg16
```
Read-Write 16-bit register (volatile unsigned int)

24.144.3.6 RwReg8

```
typedef volatile uint8_t RwReg8
```
Read-Write 8-bit register (volatile unsigned int)

24.144.3.7 WoReg

```
typedef volatile uint32_t WoReg
```
Write only 32-bit register (volatile unsigned int)

24.144.3.8 WoReg16

```
typedef volatile uint16_t WoReg16
```
Write only 16-bit register (volatile unsigned int)

24.144.3.9 WoReg8

```
typedef volatile uint32_t WoReg8
```
Write only 8-bit register (volatile unsigned int)

24.145 arch/cortex-m7/cpu/SAME70/include/SAME70Q21.h

Go to the documentation of this file.

```
00001
00035 /*
00036  * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel Support</a>
00037  */
00038
00039 #ifndef _SAME70Q21_
00040 #define _SAME70Q21_
00041
00042 /* \addtogroup SAME70Q21_definitions SAME70Q21 definitions
00043
00044   Defines structures and symbols for SAME70q21:
00045   @{
00046 */
00047
00048 #ifdef __cplusplus
00049  extern "C" {
00050 #endif
00051
00052 #if !(defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_ASM__))
00053 #include <stdint.h>
00054 #endif
00055
00056 /* IO definitions (access restrictions to peripheral registers) */
00057 #ifndef __cplusplus
00058 typedef volatile const uint32_t RoReg;
00059 typedef volatile const uint16_t RoReg16;
00060 typedef volatile const uint8_t  RoReg8;
00061 #else
00062 typedef volatile       uint32_t RoReg;
00063 typedef volatile       uint16_t RoReg16;
00064 typedef volatile       uint8_t  RoReg8;
00065 #endif
00066 typedef volatile       uint32_t WoReg;
00067 typedef volatile       uint16_t WoReg16;
00068 typedef volatile       uint32_t WoReg8;
00069 typedef volatile       uint32_t RwReg;
00070 typedef volatile       uint16_t RwReg16;
00071 typedef volatile       uint8_t  RwReg8;
00073 typedef volatile unsigned char  vubyte;
00074 typedef volatile unsigned short vuword;
00075 typedef volatile unsigned long  vudword;
00076
00077 #if !defined(SKIP_INTEGER_LITERALS)
00078
00079 #if defined(_U_) || defined(_L_) || defined(_UL_)
00080   #error "Integer Literals macros already defined elsewhere"
00081 #endif
00082
00083 #if !(defined(__ASSEMBLER__) || defined(__IAR_SYSTEMS_ASM__))
00084 /* Macros that deal with adding suffixes to integer literal constants for C/C++ */
00085 #define _U_(x) x ## U
00086 #define _L_(x) x ## L
00087 #define _UL_(x) x ## UL
00089 #else /* Assembler */
00090
00091 #define _U_(x) x
00092 #define _L_(x) x
00093 #define _UL_(x) x
00094 #endif /* !(defined(__ASSEMBLER__) || defined(__IAR_SYSTEMS_ASM__)) */
00095
00096 #endif /* SKIP_INTEGER_LITERALS */
00097 /*  @}  end of Atmel Global Defines */
00098
00099
00100 /*
00101  * \brief CMSIS DEFINITIONS FOR SAME70Q21
00102  *
00103  * \addtogroup SAME70Q21_cmsis CMSIS Definitions
00104  *
00105  * @{
00106  */
00107
00108
00109 typedef void (*IsrFn) (void);
00110
00111 typedef struct _DeviceVectors
00112 {
00113   /* Stack pointer */
00114   IsrFn pvStack;
00115
00116   /* Cortex-M handlers */
00117   IsrFn pfnReset;
00118   IsrFn pfnNMI;
```

```
00119   IsrFn pfnHardFault;
00120   IsrFn pfnMemManage;
00121   IsrFn pfnBusFault;
00122   IsrFn pfnUsageFault;
00123   IsrFn pfnReserved1;
00124   IsrFn pfnReserved2;
00125   IsrFn pfnReserved3;
00126   IsrFn pfnReserved4;
00127   IsrFn pfnSVC;
00128   IsrFn pfnDebugMon;
00129   IsrFn pfnReserved5;
00130   IsrFn pfnPendSV;
00131   IsrFn pfnSysTick;
00132
00133   /* Peripheral handlers */
00134   IsrFn pfnSUPC;   /*  0 Supply Controller */
00135   IsrFn pfnRSTC;   /*  1 Reset Controller */
00136   IsrFn pfnRTC;    /*  2 Real Time Clock */
00137   IsrFn pfnRTT;    /*  3 Real Time Timer */
00138   IsrFn pfnWDT;    /*  4 Watchdog Timer */
00139   IsrFn pfnPMC;    /*  5 Power Management Controller */
00140   IsrFn pfnEFC;    /*  6 Enhanced Embedded Flash Controller */
00141   IsrFn pfnUART0;  /*  7 UART 0 */
00142   IsrFn pfnUART1;  /*  8 UART 1 */
00143   IsrFn pvReserved9;
00144   IsrFn pfnPIOA;   /* 10 Parallel I/O Controller A */
00145   IsrFn pfnPIOB;   /* 11 Parallel I/O Controller B */
00146   IsrFn pfnPIOC;   /* 12 Parallel I/O Controller C */
00147   IsrFn pfnUSART0; /* 13 USART 0 */
00148   IsrFn pfnUSART1; /* 14 USART 1 */
00149   IsrFn pfnUSART2; /* 15 USART 2 */
00150   IsrFn pfnPIOD;   /* 16 Parallel I/O Controller D */
00151   IsrFn pfnPIOE;   /* 17 Parallel I/O Controller E */
00152   IsrFn pfnHSMCI;  /* 18 Multimedia Card Interface */
00153   IsrFn pfnTWIHS0; /* 19 Two Wire Interface 0 HS */
00154   IsrFn pfnTWIHS1; /* 20 Two Wire Interface 1 HS */
00155   IsrFn pfnSPI0;   /* 21 Serial Peripheral Interface 0 */
00156   IsrFn pfnSSC;    /* 22 Synchronous Serial Controller */
00157   IsrFn pfnTC0;    /* 23 Timer/Counter 0 */
00158   IsrFn pfnTC1;    /* 24 Timer/Counter 1 */
00159   IsrFn pfnTC2;    /* 25 Timer/Counter 2 */
00160   IsrFn pfnTC3;    /* 26 Timer/Counter 3 */
00161   IsrFn pfnTC4;    /* 27 Timer/Counter 4 */
00162   IsrFn pfnTC5;    /* 28 Timer/Counter 5 */
00163   IsrFn pfnAFEC0;  /* 29 Analog Front End 0 */
00164   IsrFn pfnDACC;   /* 30 Digital To Analog Converter */
00165   IsrFn pfnPWM0;   /* 31 Pulse Width Modulation 0 */
00166   IsrFn pfnICM;    /* 32 Integrity Check Monitor */
00167   IsrFn pfnACC;    /* 33 Analog Comparator */
00168   IsrFn pfnUSBHS;  /* 34 USB Host / Device Controller */
00169   IsrFn pfnMCAN0;  /* 35 MCAN Controller 0 */
00170   IsrFn pvReserved36;
00171   IsrFn pfnMCAN1;  /* 37 MCAN Controller 1 */
00172   IsrFn pvReserved38;
00173   IsrFn pfnGMAC;   /* 39 Ethernet MAC */
00174   IsrFn pfnAFEC1;  /* 40 Analog Front End 1 */
00175   IsrFn pfnTWIHS2; /* 41 Two Wire Interface 2 HS */
00176   IsrFn pfnSPI1;   /* 42 Serial Peripheral Interface 1 */
00177   IsrFn pfnQSPI;   /* 43 Quad I/O Serial Peripheral Interface */
00178   IsrFn pfnUART2;  /* 44 UART 2 */
00179   IsrFn pfnUART3;  /* 45 UART 3 */
00180   IsrFn pfnUART4;  /* 46 UART 4 */
00181   IsrFn pfnTC6;    /* 47 Timer/Counter 6 */
00182   IsrFn pfnTC7;    /* 48 Timer/Counter 7 */
00183   IsrFn pfnTC8;    /* 49 Timer/Counter 8 */
00184   IsrFn pfnTC9;    /* 50 Timer/Counter 9 */
00185   IsrFn pfnTC10;   /* 51 Timer/Counter 10 */
00186   IsrFn pfnTC11;   /* 52 Timer/Counter 11 */
00187   IsrFn pvReserved53;
00188   IsrFn pvReserved54;
00189   IsrFn pvReserved55;
00190   IsrFn pfnAES;    /* 56 AES */
00191   IsrFn pfnTRNG;   /* 57 True Random Generator */
00192   IsrFn pfnXDMAC;  /* 58 DMA */
00193   IsrFn pfnISI;    /* 59 Camera Interface */
00194   IsrFn pfnPWM1;   /* 60 Pulse Width Modulation 1 */
00195   IsrFn pvReserved61;
00196   IsrFn pfnSDRAMC; /* 62 SDRAM Controller */
00197   IsrFn pfnRSWDT;  /* 63 Reinforced Secure Watchdog Timer */
00198 } __attribute__((packed)) DeviceVectors;
00199
00200 /* Cortex-M7 core handlers */
00201 void Reset_Handler      ( void );
00202 void NMI_Handler        ( void );
00203 void HardFault_Handler  ( void );
00204 void MemManage_Handler  ( void );
00205 void BusFault_Handler   ( void );
```

```
00206 void UsageFault_Handler ( void );
00207 void SVC_Handler        ( void );
00208 void DebugMon_Handler   ( void );
00209 void PendSV_Handler     ( void );
00210 void SysTick_Handler    ( void );
00211
00212 /* Peripherals handlers */
00213 void ACC_Handler        ( void );
00214 void AES_Handler        ( void );
00215 void AFEC0_Handler      ( void );
00216 void AFEC1_Handler      ( void );
00217 void DACC_Handler       ( void );
00218 void EFC_Handler        ( void );
00219 void GMAC_Handler       ( void );
00220 void HSMCI_Handler      ( void );
00221 void ICM_Handler        ( void );
00222 void ISI_Handler        ( void );
00223 void MCAN0_Handler      ( void );
00224 void MCAN1_Handler      ( void );
00225 void PIOA_Handler       ( void );
00226 void PIOB_Handler       ( void );
00227 void PIOC_Handler       ( void );
00228 void PIOD_Handler       ( void );
00229 void PIOE_Handler       ( void );
00230 void PMC_Handler        ( void );
00231 void PWM0_Handler       ( void );
00232 void PWM1_Handler       ( void );
00233 void QSPI_Handler       ( void );
00234 void RSTC_Handler       ( void );
00235 void RSWDT_Handler      ( void );
00236 void RTC_Handler        ( void );
00237 void RTT_Handler        ( void );
00238 void SDRAMC_Handler     ( void );
00239 void SPI0_Handler       ( void );
00240 void SPI1_Handler       ( void );
00241 void SSC_Handler        ( void );
00242 void SUPC_Handler       ( void );
00243 void TC0_Handler        ( void );
00244 void TC1_Handler        ( void );
00245 void TC2_Handler        ( void );
00246 void TC3_Handler        ( void );
00247 void TC4_Handler        ( void );
00248 void TC5_Handler        ( void );
00249 void TC6_Handler        ( void );
00250 void TC7_Handler        ( void );
00251 void TC8_Handler        ( void );
00252 void TC9_Handler        ( void );
00253 void TC10_Handler       ( void );
00254 void TC11_Handler       ( void );
00255 void TRNG_Handler       ( void );
00256 void TWIHS0_Handler     ( void );
00257 void TWIHS1_Handler     ( void );
00258 void TWIHS2_Handler     ( void );
00259 void UART0_Handler      ( void );
00260 void UART1_Handler      ( void );
00261 void UART2_Handler      ( void );
00262 void UART3_Handler      ( void );
00263 void UART4_Handler      ( void );
00264 void USART0_Handler     ( void );
00265 void USART1_Handler     ( void );
00266 void USART2_Handler     ( void );
00267 void USBHS_Handler      ( void );
00268 void WDT_Handler        ( void );
00269 void XDMAC_Handler      ( void );
00270
00271 /*
00272  * brief CMSIS includes
00273  */
00274
00275 #include <cm_core_config.h>
00276 #include <core_cm7.h>
00277 #if !defined DONT_USE_CMSIS_INIT
00278 #include "system_same70.h"
00279 #endif /* DONT_USE_CMSIS_INIT */
00280
00281 /* @}  CMSIS group */
00282
00283
00284 #include <same70q21_sim.h>
00285
00286 /*
00287  * \brief PERIPHERAL ID DEFINITIONS FOR SAME70Q21
00288  *
00289  * \addtogroup SAME70Q21_id Peripheral Ids Definitions
00290  *
00291  * @{
00292  */
```

```
00293
00294 #define ID_SUPC   ( 0)
00295 #define ID_RSTC   ( 1)
00296 #define ID_RTC    ( 2)
00297 #define ID_RTT    ( 3)
00298 #define ID_WDT    ( 4)
00299 #define ID_PMC    ( 5)
00300 #define ID_EFC    ( 6)
00301 #define ID_UART0  ( 7)
00302 #define ID_UART1  ( 8)
00303 #define ID_SMC    ( 9)
00304 #define ID_PIOA   (10)
00305 #define ID_PIOB   (11)
00306 #define ID_PIOC   (12)
00307 #define ID_USART0 (13)
00308 #define ID_USART1 (14)
00309 #define ID_USART2 (15)
00310 #define ID_PIOD   (16)
00311 #define ID_PIOE   (17)
00312 #define ID_HSMCI  (18)
00313 #define ID_TWIHS0 (19)
00314 #define ID_TWIHS1 (20)
00315 #define ID_SPI0   (21)
00316 #define ID_SSC    (22)
00317 #define ID_TC0    (23)
00318 #define ID_TC1    (24)
00319 #define ID_TC2    (25)
00320 #define ID_TC3    (26)
00321 #define ID_TC4    (27)
00322 #define ID_TC5    (28)
00323 #define ID_AFEC0  (29)
00324 #define ID_DACC   (30)
00325 #define ID_PWM0   (31)
00326 #define ID_ICM    (32)
00327 #define ID_ACC    (33)
00328 #define ID_USBHS  (34)
00329 #define ID_MCAN0  (35)
00330 #define ID_MCAN1  (37)
00331 #define ID_GMAC   (39)
00332 #define ID_AFEC1  (40)
00333 #define ID_TWIHS2 (41)
00334 #define ID_SPI1   (42)
00335 #define ID_QSPI   (43)
00336 #define ID_UART2  (44)
00337 #define ID_UART3  (45)
00338 #define ID_UART4  (46)
00339 #define ID_TC6    (47)
00340 #define ID_TC7    (48)
00341 #define ID_TC8    (49)
00342 #define ID_TC9    (50)
00343 #define ID_TC10   (51)
00344 #define ID_TC11   (52)
00345 #define ID_AES    (56)
00346 #define ID_TRNG   (57)
00347 #define ID_XDMAC  (58)
00348 #define ID_ISI    (59)
00349 #define ID_PWM1   (60)
00350 #define ID_SDRAMC (62)
00351 #define ID_RSWDT  (63)
00353 #define ID_PERIPH_COUNT (64)
00354 /* @} SAME70Q21_id */
00355
00356 /* ************************************************************************** */
00357 /*   MEMORY MAPPING DEFINITIONS FOR SAME70Q21 */
00358 /* ************************************************************************** */
00359
00360 #define IFLASH_SIZE             (0x200000u)
00361 #define IFLASH_PAGE_SIZE        (512u)
00362 #define IFLASH_LOCK_REGION_SIZE (16384u)
00363 #define IFLASH_NB_OF_PAGES      (4096u)
00364 #define IFLASH_NB_OF_LOCK_BITS  (128u)
00365 #define IRAM_SIZE               (0x60000u)
00366
00367 #define QSPIMEM_ADDR  (0x80000000u)
00368 #define AXIMX_ADDR    (0xA0000000u)
00369 #define ITCM_ADDR     (0x00000000u)
00370 #define IFLASH_ADDR   (0x00400000u)
00371 #define IROM_ADDR     (0x00800000u)
00372 #define DTCM_ADDR     (0x20000000u)
00373 #define IRAM_ADDR     (0x20400000u)
00374 #define EBI_CS0_ADDR  (0x60000000u)
00375 #define EBI_CS1_ADDR  (0x61000000u)
00376 #define EBI_CS2_ADDR  (0x62000000u)
00377 #define EBI_CS3_ADDR  (0x63000000u)
00378 #define SDRAM_CS_ADDR (0x70000000u)
00380 /* ************************************************************************** */
00381 /*   MISCELLANEOUS DEFINITIONS FOR SAME70Q21 */
```

```
00382 /* ************************************************************************** */
00383
00384 #define CHIP_JTAGID (0x05B3D03FUL)
00385 #define CHIP_CIDR   (0xA1020E00UL)
00386 #define CHIP_EXID   (0x00000002UL)
00387
00388 /* ************************************************************************** */
00389 /*   ELECTRICAL DEFINITIONS FOR SAME70Q21 */
00390 /* ************************************************************************** */
00391
00392 /* %ATMEL_ELECTRICAL% */
00393
00394 /* Device characteristics */
00395 #define CHIP_FREQ_SLCK_RC_MIN           (20000UL)
00396 #define CHIP_FREQ_SLCK_RC               (32000UL)
00397 #define CHIP_FREQ_SLCK_RC_MAX           (44000UL)
00398 #define CHIP_FREQ_MAINCK_RC_4MHZ        (4000000UL)
00399 #define CHIP_FREQ_MAINCK_RC_8MHZ        (8000000UL)
00400 #define CHIP_FREQ_MAINCK_RC_12MHZ       (12000000UL)
00401 #define CHIP_FREQ_CPU_MAX               (300000000UL)
00402 #define CHIP_FREQ_XTAL_32K              (32768UL)
00403 #define CHIP_FREQ_XTAL_12M              (12000000UL)
00404
00405 /* Embedded Flash Read Wait State (VDDCORE set at 1.20V) */
00406 #define CHIP_FREQ_FWS_0                 (20000000UL)
00407 #define CHIP_FREQ_FWS_1                 (40000000UL)
00408 #define CHIP_FREQ_FWS_2                 (60000000UL)
00409 #define CHIP_FREQ_FWS_3                 (80000000UL)
00410 #define CHIP_FREQ_FWS_4                 (100000000UL)
00411 #define CHIP_FREQ_FWS_5                 (123000000UL)
00413 #ifdef __cplusplus
00414 }
00415 #endif
00416
00419 #endif /* _SAME70Q21_ */
```

24.146 examples/_common/EFFS/STD/src/flashChip/SAME70Q21.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-------------------------------------------------------------------
00006  * EFFS-STD configuration file for Microchip SAME70Q21 processor
00007  * onboard 2MB Flash memory. Used on the MODM7AE70 and SBE70LC.
00008  *-------------------------------------------------------------------*/
00009
00010 #ifndef _ONCHIPFLASH_H_
00011 #define _ONCHIPFLASH_H_
00012
00013 #include "file/fsf.h"
00014 #include "basictypes.h"
00015 #include "hal.h"
00016
00017 #define FLASH_NAME "SAME70Q21"
00018
00019 // functions implemented
00020 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00021
00022 // Start of Flash memory base address
00023 #define FS_FLASHBASE (0x00400000)
00024
00025 /*
00026  * BLOCKSIZE
00027  * This defines the size of the blocks to be used in the file storage area.
00028  * This must be an erasable unit of the flash chip. All blocks in the file
00029  * storage area must be the same size. This maybe different from the DESCSIZE
00030  * where the flash chip has different size erasable units available.
00031  *
00032  * SECTORSIZE
00033  * This defines the sector size. Each block is divided into a number of sectors.
00034  * This number is the smallest usable unit in the system and thus represents the
00035  * minimum file storage area. For best usage of the flash blocks the sector size
00036  * should always be a power of 2. For more information see sector section below.
00037  *
00038  * SECTORPERBLOCK
00039  * This defines the number of sectors in a block. It must always be true that:
00040  * SECTORPERBLOCK = BLOCKSIZE/SECTORSIZE
00041  *
00042  *
00043  * The memory map below is for a MODM7AE70 with a 2MB bottom boot block flash.
00044  * This example will allocate 1.5MB for the application space, and 512KB for
00045  * the file system.
00046  *
00047  * Microchip SAME70Q21:
```

```
00048  * The memory is organized in sectors. Each sector has a size of 128KB. The
00049  * first sector is divided into 3 smaller sectors. The three smaller sectors are
00050  * organized in 2 sectors of 8KB and 1 sector of 112KB.
00051  * Total number of bytes for storage: 2,097,152. Application space: 1.97MB max.
00052  *
00053  *                            Address
00054  *                           ----------
00055  *    ---------------------- 005F FFFF (End of flash space)
00056  *    |  File System Data  |
00057  *    |  256KB             |
00058  *    |  128K x 2 Blocks   |
00059  *    |--------------------| 005C 0000 (Start of File System Data)
00060  *    |  DESC BLOCK 0/1    |
00061  *    |  128K x 2 Blocks   |
00062  *    |--------------------| 0058 0000 (Start of File System)
00063  *    |                    |
00064  *    | Application        |
00065  *    | 1.375MB            |
00066  *    | 128K x 11 Blocks   |
00067  *    |                    |
00068  *    |--------------------| 0040 6004
00069  *    |   8K User Params   |
00070  *    |--------------------| 0040 4000
00071  *    |  10K Configuration |
00072  *    |--------------------| 0040 1800
00073  *    |   6K Monitor       |
00074  *    |--------------------| 0040 0000 (Start of Flash space)
00075  *
00076  *
00077  * DISABLING BACKGROUND ERASE
00078  * To use the EFFS-STD file system on the MODM7AE70 and SBE70LC, you must enable the feature:
00079  * "Locate Application at Fixed Address in Flash". This disables the background flash erase
00080  * feature, would will erase the flash file system sectors.
00081  *
00082  * If using NBEclipse:
00083  * - Right-click on the project and select "Properties"
00084  * - Navigate to C/C++ Build > Settings > NB Flashpack > General
00085  * - Check the checkbox for "Locate Application at Fixed Address in Flash"
00086  *
00087  * If you used the NBEclipse import example in the project creation, the StdFFile.a library will
00088  * be included automatically. Otherwise, the library must be added manually.
00089  * - Navigate to C/C++ Build > Settings > GNC C/C++ Linker > Libraries
00090  * - In the Libraries section, add "StdFFile"
00091  */
00092
00093 #define BLOCKSIZE      (16 * 512)    // Flash physical "sector" size, smallest erasable unit of flash
00094 #define SECTORSIZE     (512)         // Flash sector size in bytes, smallest usable unit of flash
00095 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00096
00097 /*
00098  * Specify the total amount of flash memory in the system, and the amount
00099  * allocated to be used by the file system (the rest is used by the
00100  * application.
00101  */
00102 #define FLASH_SIZE (2 * 1024 * 1024)      // Total physical flash, 2MB
00103
00104 /*
00105  * Number of sectors allocated to file system. Includes 2 Descriptor Blocks, and 1 block reserved
00106  * for file system operation. This means the usable storage space will be less than the amount
00107  * calculated below.
00108  * Total space reserved for EFFS: 256 * 512 bytes/sector = 131072 bytes.
00109  */
00110 #define FS_SIZE    (256 * 512)
00111
00112 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE) // First file system address
00113 #define BLOCKSTART 2                      // Starting file system data block. First 2 blocks are
      DESCRIPTORS
00114
00115 /*
00116  * Descriptor Blocks:
00117  * These blocks contain critical information about the file system, block allocation,
00118  * wear information and file/directory information. At least two descriptor blocks
00119  * must be included in the system, which can be erased independently. An optional
00120  * descriptor write cache may be configured which improves the performance of the
00121  * file system. Please refer to the EFFS-STD implementation guide for additional
00122  * information.
00123  */
00124 #define DESCSIZE (16 * 512)   // size of one descriptor
00125 #define DESCBLOCKSTART 0      // position of first descriptor
00126 #define DESCBLOCKEND 1        // position of last descriptor
00127 #define DESCCACHE 1024
00128
00129 #endif /* _ONCHIPFLASH_H_ */
```

24.147 examples/Parallax/src/flashChip/SAME70Q21.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 /*-------------------------------------------------------------------
00028  * EFFS-STD configuration file for MicroChip SAME70Q21 Flash Chip.
00029  * This file is part of an example that allocates 512K of flash space
00030  * to the file system, and the rest to the application.
00031  *-------------------------------------------------------------------*/
00032
00033 #ifndef _ONCHIPFLASH_H_
00034 #define _ONCHIPFLASH_H_
00035
00036 #include "file/fsf.h"
00037 #include "basictypes.h"
00038 #include "hal.h"
00039
00040 #define FLASH_NAME "SAME70Q21"
00041
00042 /*functions implemented*/
00043 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00044
00045 // Start of Flash memory base address
00046 #define FS_FLASHBASE (0x00400000)
00047
00048 /*
00049  * BLOCKSIZE
00050  * This defines the size of the blocks to be used in the file storage area.
00051  * This must be an erasable unit of the flash chip. All blocks in the file
00052  * storage area must be the same size. This maybe different from the DESCSIZE
00053  * where the flash chip has different size erasable units available.
00054  *
00055  * SECTORSIZE
00056  * This defines the sector size. Each block is divided into a number of sectors.
00057  * This number is the smallest usable unit in the system and thus represents the
00058  * minimum file storage area. For best usage of the flash blocks the sector size
00059  * should always be a power of 2. For more information see sector section below.
00060  *
00061  * SECTORPERBLOCK
00062  * This defines the number of sectors in a block. It must always be true that:
00063  * SECTORPERBLOCK = BLOCKSIZE/SECTORSIZE
00064  *
00065  *
00066  * The memory map below is for a MODM7AE70 with a 2MB bottom boot block flash.
00067  * This example will allocate 1.5MB for the application space, and 512KB for
00068  * the file system.
00069  *
00070  * Microchip SAME70Q21:
00071  * The memory is organized in sectors. Each sector has a size of 128KB. The
00072  * first sector is divided into 3 smaller sectors. The three smaller sectors are
00073  * organized in 2 sectors of 8KB and 1 sector of 112KB.
00074  * Each 128k sector is organized in pages of 512 bytes. so each 128k sector has 256 pages of 512
      bytes.
00075  * Total number of bytes for storage: 2,097,152. Application space: 1.97MB max.
00076  *
00077  *                             Address
00078  *                            ----------
00079  *    ---------------------- 005F FFFF (End of flash space)
00080  *    |  File System Data  |
00081  *    |  256KB             |
00082  *    |  128K x 2 Blocks   |
00083  *    |--------------------| 005C 0000 (Start of File System Data)
00084  *    |  DESC BLOCK 0/1    |
```

```
00085  *    |  128K x 2 Blocks   |
00086  *    |--------------------| 0058 0000 (Start of File System)
00087  *    |                    |
00088  *    | Application        |
00089  *    | 1.375MB            |
00090  *    | 128K x 11 Blocks   |
00091  *    |                    |
00092  *    |--------------------| 0040 6004
00093  *    |   8K User Params   |
00094  *    |--------------------| 0040 4000
00095  *    |  10K Configuration |
00096  *    |--------------------| 0040 1800
00097  *    |   6K Monitor       |
00098  *    |--------------------| 0040 0000 (Start of Flash space)
00099  *
00100  *
00101  * CHANGES TO COMPCODE FLAGS
00102  * In NBEclipse, or your command line makefile, change the following line
00103  * so the application will only occupy the specified application space.
00104  * The first parameter is the start of application space, and the second
00105  * is the address just below the file system space.
00106  *
00107  *    COMPCODEFLAGS = 0x00406004 0x00580000
00108  *
00109  * If using NBEclipse:
00110  * - Right-click on the project and select "Properties"
00111  * - Select "NetBurner" in the left side of the dialog box
00112  * - Verify the Platform is set to Mod5234, then check the "Use Custom Platform Settings" checkbox
00113  * - Modify the "Compcode Memory Range" to the above values
00114  *
00115  *
00116  * If using NBEclipse, you will also need to tell the linker to include the
00117  * /nburn/platform/<platform>/original/lib/libStdFFile.a library. To do this right-click on your
00118  * project, select properties, GNU Linker, then add the library.
00119  *
00120  */
00121
00122 /* WARNING: These settings are for MX29GL256F bottom boot block flash
00123  * components used on the Mod54415
00124  */
00125 #define BLOCKSIZE (16 * 512)   // flash physical "sector" size
00126 #define SECTORSIZE (512)       // file system sectors per BLOCK
00127 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00128
00129 /*
00130  * Specify the total amount of flash memory in the system, and the amount
00131  * allocated to be used by the file system (the rest is used by the
00132  * application.
00133  */
00134 #define FLASH_SIZE (2 * 1024 * 1024)        // size of total flash in the system, 2MB
00135 #define FS_SIZE    (7 * 256 * 512)          // amount allocated to file system: 2 Descriptor blocks +
      file storage
00136                                            // note that 1 block of file data will be reserved for the
      file system
00137 #define FIRST_ADDR (FLASH_SIZE - FS_SIZE)   // first file system address to use in the flash
00138 #define BLOCKSTART 2                        // first block where file system data starts
00139                                            // (first 2 blocks are DESCRIPTORS)
00140
00141 /*
00142  * Descriptor Blocks:
00143  * These blocks contain critical information about the file system, block allocation,
00144  * wear information and file/directory information. At least two descriptor blocks
00145  * must be included in the system, which can be erased independently. An optional
00146  * descriptor write cache may be configured which improves the performance of the
00147  * file system. Please refer to the EFFS-STD implementation guide for additional
00148  * information.
00149  */
00150 #define DESCSIZE (16 * 512)   // size of one descriptor
00151 #define DESCBLOCKSTART 0      // position of first descriptor
00152 #define DESCBLOCKEND 1        // position of last descriptor
00153 #define DESCCACHE 1024
00154
00155 #endif /* _ONCHIPFLASH_H_ */
```

24.148 SST39VF040.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
```

```
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 ************************************************************************/
00026
00027 /*-------------------------------------------------------------------------
00028  * EFFS-STD configuration file for SST39V040 flash memory chip.  This file is part
00029  * of an example that allocates 64KB of flash space to the file system, and the
00030  * rest to the application.
00031  *
00032  * To modify the amount of space allocated to the file system:
00033  *
00034  * 1. Change the definition in this file:  #define FS_SIZE ( 64 * 1024 )
00035  * 2. Change the compcode memory address range for the application in your
00036  *    NBEclipse project settings so that the end of the application space does
00037  *    not exceed the start of the file system space.  See the EFFS Programmer's
00038  *    Guide for details, and the header comments in main.cpp of this example
00039  *    on how to make the changes in the NBEclipse project.
00040  * 3. Be sure to add the /nburn/platform/<platform>/original/lib/libStdFFile.a library
00041  *    to your NBEclipse project C/C++ build linker library options.  See the header
00042  *    comments in main.cpp for this example on how to add the library in the
00043  *    NBEclipse project.
00044  *-------------------------------------------------------------------------*/
00045
00046 #ifndef _ONCHIPFLASH_H_
00047 #define _ONCHIPFLASH_H_
00048
00049 #include "basictypes.h"
00050 #include "hal.h"
00051 #include "file/fsf.h"
00052
00056 extern int fs_phy_OnChipFlash(FS_FLASH *flash);
00057
00058 #define FLASH_NAME "SST39VF040"
00059
00063 #define FS_FLASHBASE (0xFFC00000)
00064
00121 #define BLOCKSIZE (4 * 1024)   // Use only the 64k sectors
00122 #define SECTORSIZE (512)       // 8 sectors per block
00123 #define SECTORPERBLOCK (BLOCKSIZE / SECTORSIZE)
00124
00125 /*
00126  * Specify the total amount of flash memory in the system, and the amount
00127  * allocated to be used by the file system (the rest is used by the
00128  * application).
00129  */
00130 #define FLASH_SIZE \
00131     (512 * 1024)   // Size of total flash in the
00132                    //    system, 512kB
00133 #define FS_SIZE \
00134     (64 * 1024)   // Amount allocated to file
00135                   //    system, 64kB
00136 #define FIRST_ADDR \
00137     (FLASH_SIZE - FS_SIZE)   // First file system address to
00138                              //    use in the flash
00139 #define BLOCKSTART \
00140     (2)   // First block where file system
00141           //    data starts (first 2
00142           //    blocks are DESCRIPTORS)
00143
00144 /*
00145  * Descriptor Blocks:
00146  * These blocks contain critical information about the file system, block
00147  * allocation, wear information, and file/directory information.  At least two
00148  * descriptor blocks must be included in the system, which can be erased
00149  * independently.  An optional descriptor write cache may be configured which
00150  * improves the performance of the file system.  Please refer to the EFFS-STD
00151  * implementation guide for additional information.
00152  */
00153 #define DESCSIZE (4 * 1024)    // Size of one descriptor
00154 #define DESCBLOCKSTART (0)     // Position of first descriptor
00155 #define DESCBLOCKEND (1)       // Position of last descriptor
00156 #define DESCCACHE (1024)
00157
00158 #endif /* _ONCHIPFLASH_H_ */
```

24.149 Parallax/src/formtools.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef FORM_TOOL_H
00006 #define FORM_TOOL_H
00007
00008
00009 /* Functiosn to manage HTML form creation and data extraction */
00010
00011 /* Output a select */
00012 /* Item 1 = first selection item !!! */
00013 void FormOutputSelect( int sock, const char *name, int selnum, const char **list );
00014
00015
00016 /* Outout a Check box */
00017 void FormOutputCheckbox( int sock, const char *name, BOOL checked );
00018
00019 /* Output an input box */
00020 void FormOutputInput( int sock, const char *name, int siz, const char *val );
00021
00022 /* Output an input box for numbers */
00023 void FormOutputNumInput( int sock, const char *name, int siz, int val );
00024
00025 /*Output an input box for IP addresses */
00026 void FormOutputIPInput( int sock, const char *name, IPADDR ip );
00027
00028
00029 void ShowIP2Sock( int sock, IPADDR ip );
00030
00031 #endif
00032
```

24.150 serial/SerialBurner/src/formtools.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FORM_TOOL_H_
00006 #define _FORM_TOOL_H_
00007
00008 void ShowIP2Sock(int sock, IPADDR ip);
00009
00010 /* ---------------------------------------------------------------------------
00011  *   Functions to manage HTML form creation.
00012  * ------------------------------------------------------------------------- */
00013
00014 // Output a selection
00015 // Item 1 = First selection item!
00016 void FormOutputSelect(int sock, const char *name, int selnum, const char **list);
00017 void FormOutputSelectValueOnClick(int sock,
00018                                   const char *name,
00019                                   int selnum,
00020                                   const char **labellist,
00021                                   const char **valuelist,
00022                                   const char **onclicklist);
00023
00024 // Output a check box
00025 void FormOutputCheckbox(int sock, const char *name, BOOL checked);
00026
00027 // Output an input box
00028 void FormOutputInput(int sock, const char *name, int siz, const char *val);
00029
00030 // Output an input box for numbers
00031 void FormOutputNumInput(int sock, const char *name, int siz, int val);
00032
00033 // Output an input box for IP addresses
00034 void FormOutputIPInput(int sock, const char *name, IPADDR ip);
00035
00036 /* ---------------------------------------------------------------------------
00037  *   Functions to manage HTML data extraction.
00038  *-------------------------------------------------------------------------- */
00039
00040 // Extract an IP address from the post data
00041 IPADDR FormExtractIP(const char *name, char *pData, IPADDR def_val);
00042
00043 // Extract a number from the post data
00044 long FormExtractNum(const char *name, char *pData, long def_val);
00045
00046 // Extract a check box state from the post data
00047 BOOL FormExtractCheck(const char *name, char *pData, BOOL def_val);
```

```
00048
00049 // Extract a selection from a select box
00050 // Item 1 = First selection item!
00051 int FormExtractSel(const char *name, char *pdata, const char **pList, int defsel);
00052
00053 #endif /* _FORM_TOOL_H_ */
```

24.151 SSH/SecureSerToEthFactoryApp/src/formtools.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FORM_TOOL_H_
00006 #define _FORM_TOOL_H_
00007
00008 #include <config_obj.h>
00009 #include <nettypes.h>
00010
00011 #define PROTOCOL_HTML_INPUT_NAME "Protocol"
00012 #define DEVICE_ADDRESS_HTML_INPUT_NAME "DeviceAddress"
00013 #define DEVICE_MASK_HTML_INPUT_NAME "DeviceMask"
00014 #define DEVICE_GATE_HTML_INPUT_NAME "DeviceGate"
00015 #define DEVICE_DNS_HTML_INPUT_NAME "DeviceDns"
00016 #define DEVICE_NAME_HTML_INPUT_NAME "DeviceName"
00017 #define NTP_NAME_HTML_INPUT_NAME "NTPName"
00018
00019 void ShowIP2Sock(int sock, IPADDR ip);
00020
00021 /* ------------------------------------------------------------------------
00022  *   Functions to manage HTML form creation.
00023  * ------------------------------------------------------------------------ */
00024
00025 unsigned char asciiString2Byte(unsigned char *value);
00026 // Output a selection
00027 // Item 1 = First selection item!
00028 void FormOutputSelect(int sock, const char *name, int selnum, const char **list);
00029 void FormOutputSelectValueOnClick(int sock,
00030                                   const char *name,
00031                                   int selnum,
00032                                   const char **labellist,
00033                                   const char **valuelist,
00034                                   const char **onclicklist);
00035
00036 // Output a check box
00037 void FormOutputCheckbox(int sock, const char *name, bool checked);
00038
00039 // // Output an input box
00040 // void FormOutputInput( int sock, const char* name, int siz, const char* val );
00041
00042 // // Output an input box for numbers
00043 // void FormOutputNumInput( int sock, const char* name, int siz, int val );
00044
00045 // // Output an input box for bytes
00046 // void FormOutputByteInput(int sock, const char* name, int siz, int val);
00047
00048 // // Output an input box for IP addresses
00049 // void FormOutputIPInput( int sock, const char* name, IPADDR4 ip );
00050
00051 // Output value and choices of a config_chooser as a drop down select box
00052 // void FormOutputConfigChooser(int sock, const char* name, config_chooser chooser);
00053
00054 /* ------------------------------------------------------------------------
00055  *   Functions to manage HTML data extraction.
00056  *------------------------------------------------------------------------ */
00057
00058 // // Extract an IP address from the post data
00059 // IPADDR4 FormExtractIP( const char* name, char* pData, IPADDR4 def_val );
00060
00061 // // Extract a number from the post data
00062 // long FormExtractNum( const char* name, char* pData, long def_val );
00063
00064 // // Extract a byte from the post data
00065 // long FormExtractByte(const char* name, char* pData, long def_val);
00066
00067 // // Extract a check box state from the post data
00068 // bool FormExtractCheck( const char* name, char* pData, bool def_val );
00069
00070 int ExtractPostData(PCSTR name, PCSTR data, PSTR dest_buffer, int maxlen);
00071
00072 #endif /* _FORM_TOOL_H_ */
```

24.152 SSL/HttpsUploadCert/src/formtools.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FORM_TOOL_H_
00006 #define _FORM_TOOL_H_
00007
00014 void ShowIP2Sock(int sock, IPADDR ip);
00015
00030 void FormOutputSelect(int sock, const char *name, int selnum, const char **list);
00031
00042 void FormOutputSelectValueOnClick(int sock,
00043                                   const char *name,
00044                                   int selnum,
00045                                   const char **labellist,
00046                                   const char **valuelist,
00047                                   const char **onclicklist);
00048
00056 void FormOutputCheckbox(int sock, const char *name, BOOL checked);
00057
00066 void FormOutputInput(int sock, const char *name, int siz, const char *val);
00067
00076 void FormOutputNumInput(int sock, const char *name, int siz, int val);
00077
00085 void FormOutputIPInput(int sock, const char *name, IPADDR ip);
00086
00087 #endif /* _FORM_TOOL_H_ */
```

24.153 _common/EFFS/STD/src/fs_main.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __FS_MAIN_H
00006 #define __FS_MAIN_H
00007
00008 #define USE_NOR
00009
00010 /* Drive numbers */
00011 #define NOR_DRV_NUM 0
00012 #define STDRAM_DRV_NUM 1
00013 #define MMC_DRV_NUM 2
00014 #define CFC_DRV_NUM 3
00015 #define HDD_DRV_NUM 3
00016 #define FATRAM_DRV_NUM 4
00017
00018 #if ((defined USE_NOR) | (defined USE_STDRAM)) & ((defined USE_CFC) | (defined USE_HDD) | (defined
      USE_MMC) | (defined USE_FATRAM))
00019 #define FS_WRAPPER
00020 #elif (defined USE_NOR) | (defined USE_STDRAM)
00021 #define FS_STD
00022 #elif (defined USE_CFC) | (defined USE_HDD) | (defined USE_MMC) | (defined USE_FATRAM)
00023 #define FS_FAT
00024 #endif
00025
00026 #include "file/fsf.h"
00027 #define fd_mountstd fs_mountdrive
00028 #define fd_format(d, t) fs_format(d)
00029 #define fd_hardformat(d, t) fs_format(d)
00030 #define fd_getdrive fs_getdrive
00031 #define fd_chdrive fs_chdrive
00032 #define fd_getcwd fs_getcwd
00033 #define fd_chdir fs_chdir
00034 #define fd_mkdir fs_mkdir
00035 #define fd_rmdir fs_rmdir
00036 #define fd_getfreespace fs_getfreespace
00037 #define fd_findfirst fs_findfirst
00038 #define fd_findnext fs_findnext
00039 #define fd_filelength fs_filelength
00040 #define fd_delete fs_delete
00041 #define fd_rename fs_rename
00042 #define fd_open fs_open
00043 #define fd_close fs_close
00044 #define fd_read fs_read
00045 #define fd_write fs_write
00046 #define FD_FIND FS_FIND
00047 #define FD_FILE FS_FILE
00048 #define FD_SPACE FS_SPACE
00049 #define FD_ATTR_DIR FS_ATTR_DIR
00050
00051 #endif /* __FS_MAIN_H */
```

24.154 Parallax/src/fs_main.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 #ifndef __FS_MAIN_H
00028 #define __FS_MAIN_H
00029
00030 #define USE_NOR
00031
00032 /* Drive numbers */
00033 #define NOR_DRV_NUM 0
00034 #define STDRAM_DRV_NUM 1
00035 #define MMC_DRV_NUM 2
00036 #define CFC_DRV_NUM 3
00037 #define HDD_DRV_NUM 3
00038 #define FATRAM_DRV_NUM 4
00039
00040 #if ((defined USE_NOR) | (defined USE_STDRAM)) & ((defined USE_CFC) | (defined USE_HDD) | (defined
      USE_MMC) | (defined USE_FATRAM))
00041 #define FS_WRAPPER
00042 #elif (defined USE_NOR) | (defined USE_STDRAM)
00043 #define FS_STD
00044 #elif (defined USE_CFC) | (defined USE_HDD) | (defined USE_MMC) | (defined USE_FATRAM)
00045 #define FS_FAT
00046 #endif
00047
00048 #include "file/fsf.h"
00049 #define fd_mountstd fs_mountdrive
00050 #define fd_format(d, t) fs_format(d)
00051 #define fd_hardformat(d, t) fs_format(d)
00052 #define fd_getdrive fs_getdrive
00053 #define fd_chdrive fs_chdrive
00054 #define fd_getcwd fs_getcwd
00055 #define fd_chdir fs_chdir
00056 #define fd_mkdir fs_mkdir
00057 #define fd_rmdir fs_rmdir
00058 #define fd_getfreespace fs_getfreespace
00059 #define fd_findfirst fs_findfirst
00060 #define fd_findnext fs_findnext
00061 #define fd_filelength fs_filelength
00062 #define fd_delete fs_delete
00063 #define fd_rename fs_rename
00064 #define fd_open fs_open
00065 #define fd_close fs_close
00066 #define fd_read fs_read
00067 #define fd_write fs_write
00068 #define FD_FIND FS_FIND
00069 #define FD_FILE FS_FILE
00070 #define FD_SPACE FS_SPACE
00071 #define FD_ATTR_DIR FS_ATTR_DIR
00072
00073 #endif /* __FS_MAIN_H */
```

24.155 _common/EFFS/STD/src/ftp_fs.h

```
00001 /*NB_REVISION*/
00002
00003 /*****************************************************************************
00004  *
00005  *   Copyright (c) 2003 by HCC Embedded
```

```
00006  *
00007  *   This software is copyrighted by and is the sole property of HCC.  All
00008  *   rights, title, ownership, or other interests in the software remain the
00009  *   property of HCC.  This software may only be used in accordance with the
00010  *   corresponding license agreement.  Any unauthorized use, duplication,
00011  *   transmission, distribution, or disclosure of this software is expressly
00012  *   forbidden.
00013  *
00014  *   This copyright notice may not be removed or modified without prior written
00015  *   consent of HCC.
00016  *
00017  *   HCC reserves the right to modify this software without notice.
00018  *
00019  *   HCC Embedded
00020  *   Budapest 1132
00021  *   Victor Hugo Utca 11-15
00022  *   Hungary
00023  *
00024  *   Tel:    +36 (1) 450 1302
00025  *   Fax:    +36 (1) 450 1303
00026  *   http:   www.hcc-embedded.com
00027  *   E-mail: info@hcc-embedded.com
00028  *
00029  ******************************************************************************/
00030
00031 #ifndef _FTP_F_H
00032 #define _FTP_F_H
00033
00034 #if ((defined USE_NOR) | (defined USE_STDRAM)) & ((defined USE_CFC) | (defined USE_HDD) | (defined
      USE_MMC) | (defined USE_FATRAM))
00035 #define FS_WRAPPER
00036 #elif (defined USE_NOR) | (defined USE_STDRAM)
00037 #define FS_STD
00038 #elif (defined USE_CFC) | (defined USE_HDD) | (defined USE_MMC) | (defined USE_FATRAM)
00039 #define FS_FAT
00040 #endif
00041
00042 #endif /* _FTP_F_H */
```

24.156 Parallax/src/ftp_fs.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /******************************************************************************
00004  *
00005  *   Copyright (c) 2003 by HCC Embedded
00006  *
00007  *   This software is copyrighted by and is the sole property of HCC.  All
00008  *   rights, title, ownership, or other interests in the software remain the
00009  *   property of HCC.  This software may only be used in accordance with the
00010  *   corresponding license agreement.  Any unauthorized use, duplication,
00011  *   transmission, distribution, or disclosure of this software is expressly
00012  *   forbidden.
00013  *
00014  *   This copyright notice may not be removed or modified without prior written
00015  *   consent of HCC.
00016  *
00017  *   HCC reserves the right to modify this software without notice.
00018  *
00019  *   HCC Embedded
00020  *   Budapest 1132
00021  *   Victor Hugo Utca 11-15
00022  *   Hungary
00023  *
00024  *   Tel:    +36 (1) 450 1302
00025  *   Fax:    +36 (1) 450 1303
00026  *   http:   www.hcc-embedded.com
00027  *   E-mail: info@hcc-embedded.com
00028  *
00029  ******************************************************************************/
00030
00031 #ifndef _FTP_F_H
00032 #define _FTP_F_H
00033
00034 #if ((defined USE_NOR) | (defined USE_STDRAM)) & ((defined USE_CFC) | (defined USE_HDD) | (defined
      USE_MMC) | (defined USE_FATRAM))
00035 #define FS_WRAPPER
00036 #elif (defined USE_NOR) | (defined USE_STDRAM)
00037 #define FS_STD
00038 #elif (defined USE_CFC) | (defined USE_HDD) | (defined USE_MMC) | (defined USE_FATRAM)
00039 #define FS_FAT
00040 #endif
00041
00042 #endif /* _FTP_F_H */
```

24.157 _common/EFFS/FAT/src/http_f.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _HTTP_F_H
00006 #define _HTTP_F_H
00007
00008 void RegisterWebFuncs();
00009
00010 #endif /* _HTTP_F_H */
```

24.158 _common/EFFS/STD/src/http_f.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _HTTP_F_H
00006 #define _HTTP_F_H
00007
00008 void RegisterWebFuncs();
00009
00010 #endif /* _HTTP_F_H */
```

24.159 Parallax/src/http_f.h

```
00001 /* Revision: 3.3.8 */
00002
00003 /*****************************************************************************
00004 * Copyright 1998-2022 NetBurner, Inc.  ALL RIGHTS RESERVED
00005 *
00006 *    Permission is hereby granted to purchasers of NetBurner Hardware to use or
00007 *    modify this computer program for any use as long as the resultant program
00008 *    is only executed on NetBurner provided hardware.
00009 *
00010 *    No other rights to use this program or its derivatives in part or in
00011 *    whole are granted.
00012 *
00013 *    It may be possible to license this or other NetBurner software for use on
00014 *    non-NetBurner Hardware. Contact sales@Netburner.com for more information.
00015 *
00016 *    NetBurner makes no representation or warranties with respect to the
00017 *    performance of this computer program, and specifically disclaims any
00018 *    responsibility for any damages, special or consequential, connected with
00019 *    the use of this program.
00020 *
00021 * NetBurner
00022 * 16855 W Bernardo Dr
00023 * San Diego, CA 92127
00024 * www.netburner.com
00025 *****************************************************************************/
00026
00027 #ifndef _HTTP_F_H
00028 #define _HTTP_F_H
00029
00030 void RegisterWebFuncs();
00031
00032 #endif /* _HTTP_F_H */
```

24.160 Parallax/src/nvsettings.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005
00006 #define REG_VAR_SIZE 64     // Standard register variable size in bytes
00007 #define DEV_NAME_SIZE 40    // Device name size in bytes
00008 #define HTTP_PORT_SIZE 10   // HTTP port size in bytes
00009
00010
00011 /* The structure that holds the configuration data */
00012 struct NV_SettingsStruct
00013 {
00014       uint32_t DataBaudRate;
00015       char DeviceName[DEV_NAME_SIZE];
00016       uint8_t Output_Bits;
```

```
00017       uint8_t Output_Stop;
00018       uint8_t Output_Parity;
00019       uint8_t IP_Addr_mode;
00020       uint8_t Var_ET[REG_VAR_SIZE];
00021       uint8_t Var_EF[REG_VAR_SIZE];
00022       uint8_t Var_ES[REG_VAR_SIZE];
00023       uint8_t Var_EC[REG_VAR_SIZE];
00024       uint8_t Var_EV[REG_VAR_SIZE];
00025       uint8_t Var_EU[REG_VAR_SIZE];
00026       uint8_t Var_EP[REG_VAR_SIZE];
00027       uint8_t Var_EA[REG_VAR_SIZE];
00028       uint8_t Var_EW[REG_VAR_SIZE];
00029       uint8_t Var_BI[REG_VAR_SIZE];
00030       uint8_t Var_BM[REG_VAR_SIZE];
00031       uint8_t Var_BP[REG_VAR_SIZE];
00032       uint8_t Var_XX[20][REG_VAR_SIZE];
00033       uint8_t Config_User[20];
00034       uint8_t Config_Pass[20];
00035       uint8_t Web_User[20];
00036       uint8_t Web_Pass[20];
00037       uint8_t Web_Text[20];
00038       uint8_t Var_HP[HTTP_PORT_SIZE];          // HTTP port number for var interface
00039       uint32_t VerifyKey;
00040 };
00041
00042
00043 /* Declare global variable */
00044 extern struct NV_SettingsStruct NV_Settings;
00045 extern volatile BOOL Settings_Changed;
00046
00047
00048 /* Constants that go with IP_Addr_mode */
00049 #define IP_ADDR_MODE_DHCP (1)
00050 #define IP_ADDR_MODE_STATIC (2)
00051
00052 /* Number of NB_Vars */
00053 #define NB_VAR_CNT 250
00054
00055 extern     uint8_t Var_ET[REG_VAR_SIZE];
00056 extern     uint8_t Var_EF[REG_VAR_SIZE];
00057 extern     uint8_t Var_ES[REG_VAR_SIZE];
00058 extern     uint8_t Var_EC[REG_VAR_SIZE];
00059 extern     uint8_t Var_EV[REG_VAR_SIZE];
00060 extern     uint8_t Var_EU[REG_VAR_SIZE];
00061 extern     uint8_t Var_EP[REG_VAR_SIZE];
00062 extern     uint8_t Var_EA[REG_VAR_SIZE];
00063 extern     uint8_t Var_EW[REG_VAR_SIZE];
00064 extern     uint8_t Var_BI[REG_VAR_SIZE];
00065 extern     uint8_t Var_BM[REG_VAR_SIZE];
00066 extern     uint8_t Var_BP[REG_VAR_SIZE];
00067 extern     uint8_t Var_XX[NB_VAR_CNT][REG_VAR_SIZE];
00068
00069 extern uint8_t Var_SI[REG_VAR_SIZE];  // Current IP Address
00070 extern uint8_t Var_SN[REG_VAR_SIZE];  // Current Network MASK
00071 extern uint8_t Var_SG[REG_VAR_SIZE];  // Current Gateway Address
00072 extern uint8_t Var_SD[REG_VAR_SIZE];  // Current DNS Server Address
00073 extern uint8_t Var_SU[REG_VAR_SIZE];  // Last UDP IP received
00074 extern uint8_t Var_HP[REG_VAR_SIZE];  // HTTP port number
00075 extern uint8_t PostVar;
00076 extern uint8_t StatusVar;
00077
00078 #define BIT_VALID_LINK (1)
00079 #define BIT_POST_UPDATE (2)
00080 #define BIT_READY_TO_SEND_EMAIL (4)
00081 #define BIT_EMAIL_SUCCESS (16)
00082 #define BIT_UDP_RECEIVED (32);
00083
00084 #define PARALLAX_UDP_PORT (10000)
00085
00086
00087 /*---------------------------------------------------------------------
00088  Check NV Settings.  Assign default values if VerifyKey is not
00089  valid.
00090  ---------------------------------------------------------------------*/
00091 void CheckNVSettings();
00092
```

24.161 serial/SerialBurner/src/nvsettings.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NVSETTINGS_H_
00006 #define _NVSETTINGS_H_
```

```
00007
00008 #define VERIFY_KEY (0x48666050)   // NV Settings key code
00009
00010 /*
00011  * The default number of seconds between receiving TCP chars before a timeout occurs.
00012  * The system default timeout, TCP_WRITE_TIMEOUT, is 10 seconds, which is the minimum
00013  * timeout value. A timeout value of 0 will disable the timeout feature.
00014  * */
00015 #define DEF_INACTIVITY_TIMEOUT (60)   // inactivity timeout in seconds
00016
00017 /* If a new client TCP connection is attempted while one is active,
00018  * one of the following three actions can be taken:
00019  *      a) Ignore the incoming connection (leave current connection active)
00020  *           (set override timeout to 0xFFFFFFFF)
00021  *      b) Replace the existing connection if it has been idle for a specified number of seconds.
00022  *           (set override to the number of seconds to wait)
00023  *      c) Always replace the existing connection.
00024  *           (set override to 0 seconds)
00025  *
00026  *    This is done with the override timeout setting below:
00027  *    The default number of seconds to wait before a new connection can override an
00028  *    existing connection.
00029  */
00030 #define DEF_OVERRIDE_TIMEOUT (20)
00031
00032 struct NV_SettingsStruct
00033 {
00034     uint32_t VerifyKey;
00035     uint16_t ServerListenPort;
00036     uint16_t ClientTimeout;
00037     uint16_t ClientOverrideTimeout;
00038     uint32_t DataBaudRate;
00039     uint16_t SerialDataFlowControl;
00040 };
00041
00042 extern NV_SettingsStruct NV_Settings;   // Non-volatile settings to store in flash memory
00043
00044 #endif
```

24.162 SSH/SshServerUserKey/src/nvsettings.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NVSETTINGS_H_
00006 #define NVSETTINGS_H_
00007
00008 #include <basictypes.h>
00009 #include "nbfactory.h"
00010
00011 // Configuration verify key (increment if data changed, added, reorganized)
00012 #define NB_FACTORY_VERIFY_KEY (0x5e545064)
00013 #define NTP_NAME_LENGTH (35)
00014 #define DEVICE_NAME_LENGTH (15)
00015
00016 struct NV_SettingsStruct
00017 {
00018     char DeviceName[(DEVICE_NAME_LENGTH + 1)];
00019     char NTPName[NTP_NAME_LENGTH + 1];
00020     // IPADDR NTP_Addr;
00021
00022     /* SSH key source and lengths (default and user installed) */
00023     uint8_t SshKeyEccSource;   // Library default, app default, or user installed
00024     uint16_t SshKeyEccLength;
00025     uint8_t SshKeyRsaSource;   // Library default, app default, or user installed
00026     uint16_t SshKeyRsaLength;
00027
00028     /* Version verification key */
00029     uint32_t VerifyKey;
00030 };
00031
00032 extern void CheckNVSettings(BOOL returnToFactory);
00033
00034 #endif /* NVSETTINGS_H_ */
```

24.163 ow.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
```

```
00005 #include <predef.h>
00006 #include <nbrtos.h>
00007 #include <init.h>
00008 #include <stdio.h>
00009 #include <ctype.h>
00010 #include <sim.h>
00011 #include <pins.h>
00012 #include <cfinter.h>
00013
00014 #define OW_CR_PST 0x40   // Presence Set bit
00015 #define OW_CR_RPP 0x80   // Reset/Presence-detect pulse
00016 #define OW_IER_ERBF \
00017     0x10   // Enable receiver buffer full interrupt, and
00018            // Receive buffer full flag
00019 #define OW_IER_ETSE 0x08   // Enable transmit shift register empty interrupt
00020 #define OW_IER_ETBE 0x04   // Enable transmit buffer empty interrupt
00021
00022 #ifndef OW_HPP_
00023 #define OW_HPP_
00024
00025 /*
00026  * OWState
00027  *     An enum type that defines the possible states of the 1-Wire module.
00028  */
00029 enum OWState
00030 {
00031     NOT_INIT = 0,
00032     OK = 1,
00033     READ = 2,
00034     WRITE = 3
00035 };
00036
00037 /*
00038  *   owDriverStruct
00039  *       This struct contains the state variables used for the 1-Wire driver.
00040  *
00041  *   OW_State state - The state of the 1-Wire module.  See the enumerated type
00042  *       above.
00043  *
00044  *   uint8_t* buf - Points to the data buffer.  This is the memory location
00045  *       from which data will be read or written to the peripheral.
00046  *
00047  *   8int8_t remLength - Counter that keeps track of the number of bytes
00048  *       that still need to be read or written.
00049  *
00050  *   OS_SEM* sem - Pointer to the external semaphore for the OW_read and
00051  *       OW_write functions.
00052  */
00053 struct owDriverStruct
00054 {
00055     volatile OWState state;
00056     volatile uint8_t *buf;
00057     volatile int remLength;
00058     OS_SEM *sem;
00059
00060     // Default constructor
00061     owDriverStruct() : state(NOT_INIT), buf(NULL), remLength(0), sem(NULL) {}
00062 };
00063
00064 /*
00065  * 1-Wire interrupt service routine.
00066  *     Called by hardware when event occurs on the 1-Wire Module
00067  */
00068 void OWMasterIsr();
00069
00070 /*
00071  * OWInit()
00072  *     This function initializes the 1-Wire module registers and sets up the
00073  *     interrupt service routine
00074  */
00075 void OWInit();
00076
00077 /*
00078  * OWDetectDevices()
00079  *     Generates a reset pulse and listens for a response from external
00080  *     devices.  Returns true if a device is detected, false otherwise
00081  */
00082 bool OWDetectDevices();
00083
00084 /*
00085  * OWRead(uint8_t* data, int len, OS_SEM *sem)
00086  *     Reads len bytes from the 1-Wire Bus.  Data points to the buffer
00087  *     to which the bytes are read. The function  posts to the semaphore
00088  *     sem when complete (optional).
00089  */
00090 OWState OWRead(uint8_t *data, int len, OS_SEM *sem = NULL);
00091
```

```
00092 /*
00093  * OWWrite(uint8_t* data, int len, OS_SEM *sem)
00094  *     Writes len bytes to the 1-Wire Bus.  Data points to the buffer from
00095  *     which bytes are written.  The function posts to the semaphore
00096  *     sem when complete (optional).
00097  */
00098 OWState OWWrite(uint8_t *data, int len, OS_SEM *sem = NULL);
00099
00100 /*
00101  * OWDone()
00102  *     Returns true if the driver is ready to be used/read/write data on the bus
00103  */
00104 bool OWDone();
00105
00106 #endif /* OW_HPP_ */
```

24.164 PeriodicAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #define ADC_CR1_STOP 0x4000
00006 #define ADC_CR1_START 0x2000
00007 #define ADC_CR1_LOOP 0x0002
00008 #define ADC_CR2_INIT_OFF 0x0000
00009 #define ADC_ZCCR_OFF 0x0000
00010 #define ADC_SDIS_ENABLE_ALL 0x0000
00011 #define ADC_SDIS_DISABLE_ALL 0x0000
00012
00013 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00014
00015 void StartAD();   // Start A/D conversion set.
00016
00017 void StartADLoop();   // Start A/D conversion set, continuous loop mode.
00018
00019 void StopAD();   // Stop A/D conversion set.
00020
00021 bool ADDone();   // Return true if the conversion is complete
00022
00023 uint16_t GetADResult(int ch);   // Return the AD Result
```

24.165 PeriodicAD_DMA.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <basictypes.h>
00006
00007 #define DMA_CH_DTER_0 8
00008 #define DMA_CH_DTER_1 9
00009 #define DMA_CH_DTER_2 10
00010 #define DMA_CH_DTER_3 11
00011
00012 #define DMA_ERR_NOERR 0
00013 #define DMA_ERR_CHNOTSET 1
00014 #define DMA_ERR_TMRNOTSET 2
00015 #define DMA_ERR_TMRRUNNING 3
00016
00017 // Initialize the dma timer to sample the ADC
00018 int initADC_DMA(int channel = 0, int timerNum = 3);
00019
00020 // Set the sample period for the ADC (in us)
00021 double setSamplePeriod(uint32_t period);
00022
00023 // Set the sample frequency (in Hz)
00024 double setSampleFreq(uint32_t freq);
00025
00026 // Start logging the ADC using DMA. The period is in units of 1 us
00027 int startADC_DMA(void *buffer, uint32_t bufferSize, void (*isrfunc)(void *, void *));
00028
00029 // Stop the DMA Timer and ADC
00030 void stopADC_DMA();
```

24.166 wavWriter.h

```
00001 /*NB_REVISION*/
00002
```

```
00003 /*NB_COPYRIGHT*/
00004
00005 #include <basictypes.h>
00006
00007 #define NUM_WAV_WRITERS 5
00008 #define SWPBUFFSIZE 512
00009
00010 // wavWriter expects data from onboard ADC straight from the result reg, with offset 0
00011 int openNewWav(char *fileName, uint16_t channels, uint32_t sampleRate);
```

24.167 examples/PlatformSpecific/MCF5441X/ADC/SimpleADC/src/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00006
00007 void StartAD();   // Start A/D conversion set.
00008
00009 bool ADDone();   // Return true if the conversion is complete
00010
00011 uint16_t GetADResult(int ch);   // Return the AD Result
```

24.168 examples/PlatformSpecific/MCF5441X/MOD5441X/Mod5441xFactoryApp/src/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void InitSingleEndAD();         // Setup the A/D converter to be ready to run
00006 void StartAD();                 // Start A/D conversion set
00007 bool ADDone();                  // Return true if the conversion is complete
00008 uint16_t GetADResult(int ch);   // Return the AD Result
```

24.169 examples/PlatformSpecific/MCF5441X/NANO54415/NANO54415FactoryApp/src/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00006
00007 void StartAD();   // Start A/D conversion set.
00008
00009 bool ADDone();   // Return true if the conversion is complete
00010
00011 uint16_t GetADResult(int ch);   // Return the AD Result
```

24.170 examples/PlatformSpecific/MIMXRT1061/ADC_Simple/src/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00006
00007 // Enable or disable individual AFEC channels
00008 void ADConfigCh(unsigned nAfec, unsigned ch, bool enable);
00009
00010 void StartAD();   // Start A/D conversion set.
00011
00012 bool ADDone();   // Return true if the conversion is complete
00013
00014 uint16_t GetADResult(unsigned nAfec, unsigned ch);   // Return the AD Result
```

24.171 examples/PlatformSpecific/SAME70/MODM7AE70/ADC_↩ Simple/src/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00006
00007 // Enable or disable individual AFEC channels
00008 void ADConfigCh(unsigned nAfec, unsigned ch, bool enable);
00009
00010 void StartAD();   // Start A/D conversion set.
00011
00012 bool ADDone();   // Return true if the conversion is complete
00013
00014 uint16_t GetADResult(unsigned nAfec, unsigned ch);   // Return the AD Result
```

24.172 examples/PlatformSpecific/SOMRT1061/SEMC_Simple/src/↩ SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00006
00007 // Enable or disable individual AFEC channels
00008 void ADConfigCh(unsigned nAfec, unsigned ch, bool enable);
00009
00010 void StartAD();   // Start A/D conversion set.
00011
00012 bool ADDone();   // Return true if the conversion is complete
00013
00014 uint16_t GetADResult(unsigned nAfec, unsigned ch);   // Return the AD Result
```

24.173 examples/WebSockets/DIPSwitches/src/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <basictypes.h>
00006
00007 // Setup the A/D converter to be ready to run
00008 void InitSingleEndAD();
00009
00010 // Start A/D conversion set.
00011 void StartAD();
00012
00013 // Return true if the conversion is complete
00014 bool ADDone();
00015
00016 // Return the AD Result
00017 uint16_t GetADResult(int ch);
```

24.174 platform/NANO54415/include/SimpleAD.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 // Set up the A/D converter to be ready to Run.
00006 void InitSingleEndAD();
00007
00008 void StartAD();
00009 ;   // Start A/D conversion set.
00010
00011 bool ADDone();   // Return true if the conversion is done.
00012
00013 uint16_t GetADResult(int ch);   // Get the AD Result
```

24.175 platform/SB800EX/include/SimpleAD.h

```
00001
00002
00003 // Set up the A/D converter to be ready to Run.
00004 void InitSingleEndAD();
00005
00006 void StartAD();
00007 ;   // Start A/D conversion set.
00008
00009 bool ADDone();   // Return true if the conversion is done.
00010
00011 uint16_t GetADResult(int ch);   // Get the AD Result
```

24.176 dev_test.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Device tests implementation */
00006 #ifndef _DEV_TEST_H
00007 #define _DEV_TEST_H
00008
00009 #include <effs_fat/fat.h>
00010
00011 #ifdef __cplusplus
00012 extern "C"
00013 {
00014 #endif
00015
00016     int SpeedTest(char *pFileName, unsigned int size = 1024);
00017
00018 #ifdef __cplusplus
00019 }
00020 #endif
00021
00022 #endif /* _DEV_TEST_H */
```

24.177 _common/EFFS/FAT/src/ftp_f.h

```
00001 /*NB_REVISION*/
00002
00003 /***************************************************************************
00004  *
00005  *   Copyright (c) 2003 by HCC Embedded
00006  *
00007  *   This software is copyrighted by and is the sole property of HCC.  All
00008  *   rights, title, ownership, or other interests in the software remain the
00009  *   property of HCC.  This software may only be used in accordance with the
00010  *   corresponding license agreement.  Any unauthorized use, duplication,
00011  *   transmission, distribution, or disclosure of this software is expressly
00012  *   forbidden.
00013  *
00014  *   This copyright notice may not be removed or modified without prior written
00015  *   consent of HCC.
00016  *
00017  *   HCC reserves the right to modify this software without notice.
00018  *
00019  *   HCC Embedded
00020  *   Budapest 1132
00021  *   Victor Hugo Utca 11-15
00022  *   Hungary
00023  *
00024  *   Tel:    +36 (1) 450 1302
00025  *   Fax:    +36 (1) 450 1303
00026  *   http:   www.hcc-embedded.com
00027  *   E-mail: info@hcc-embedded.com
00028  *
00029  ***************************************************************************/
00030
00031 #ifndef _FTP_F_H
00032 #define _FTP_F_H
00033
00034 #if ((defined USE_NOR) | (defined USE_STDRAM)) & ((defined USE_CFC) | (defined USE_HDD) | (defined
      USE_MMC) | (defined USE_FATRAM))
00035 #define FS_WRAPPER
00036 #elif (defined USE_NOR) | (defined USE_STDRAM)
00037 #define FS_STD
00038 #elif (defined USE_CFC) | (defined USE_HDD) | (defined USE_MMC) | (defined USE_FATRAM)
00039 #define FS_FAT
00040 #endif
```

```
00041
00042 #endif /* _FTP_F_H */
```

24.178 PlatformSpecific/MCF5441X/EffsSDHC/src/ftp_f.h

```
00001 /*NB_REVISION*/
00002
00003 /*****************************************************************************
00004  *
00005  *   Copyright (c) 2003 by HCC Embedded
00006  *
00007  *   This software is copyrighted by and is the sole property of HCC.  All
00008  *   rights, title, ownership, or other interests in the software remain the
00009  *   property of HCC.  This software may only be used in accordance with the
00010  *   corresponding license agreement.  Any unauthorized use, duplication,
00011  *   transmission, distribution, or disclosure of this software is expressly
00012  *   forbidden.
00013  *
00014  *   This copyright notice may not be removed or modified without prior written
00015  *   consent of HCC.
00016  *
00017  *   HCC reserves the right to modify this software without notice.
00018  *
00019  *   HCC Embedded
00020  *   Budapest 1132
00021  *   Victor Hugo Utca 11-15
00022  *   Hungary
00023  *
00024  *   Tel:    +36 (1) 450 1302
00025  *   Fax:    +36 (1) 450 1303
00026  *   http:   www.hcc-embedded.com
00027  *   E-mail: info@hcc-embedded.com
00028  *
00029  *****************************************************************************/
00030
00031 #ifndef _FTP_F_H
00032 #define _FTP_F_H
00033
00034 #if ((defined USE_NOR) | (defined USE_STDRAM)) & ((defined USE_CFC) | (defined USE_HDD) | (defined
      USE_MMC) | (defined USE_FATRAM))
00035 #define FS_WRAPPER
00036 #elif (defined USE_NOR) | (defined USE_STDRAM)
00037 #define FS_STD
00038 #elif (defined USE_CFC) | (defined USE_HDD) | (defined USE_MMC) | (defined USE_FATRAM)
00039 #define FS_FAT
00040 #endif
00041
00042 #endif /* _FTP_F_H */
```

24.179 tests.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _TESTS_H_
00006 #define _TESTS_H_
00007
00008 void displayEffsErrorCode(int code);
00009 void doSingleMemCardTest(int mmc_drive);
00010 void doDualMemCardTest(void);
00011 void doManfTest(void);
00012
00013 #endif /* _TESTS_H_ */
```

24.180 webfuncs.cpp File Reference

This code module contains the web functions for the POP3 mail using SSL example program.

```
#include <crypto/ssl_pop3.h>
#include <dns.h>
#include <effs_fat/fat.h>
#include <httppost.h>
#include <mailto.h>
#include <netinterface.h>
#include <stdlib.h>
```

```
#include "FileSystemUtils.h"
#include "cardtype.h"
```

Functions

- void WebDisplayDhcpSelect (int sock, PCSTR url)

  Displays if the network values are statically set or assigned by DHCP.
- void WebShowDhcpDeviceIpAddress (int sock, PCSTR url)

  Displays the active IP value.
- void WebShowStaticDeviceIpAddress (int sock, PCSTR url)

  Displays the static IP value.
- void WebShowDhcpDeviceIpMask (int sock, PCSTR url)

  Displays the active IP mask value.
- void WebShowStaticDeviceIpMask (int sock, PCSTR url)

  Displays the static IP mask value.
- void WebShowDhcpDeviceGateway (int sock, PCSTR url)

  Displays the active gateway value.
- void WebShowStaticDeviceGateway (int sock, PCSTR url)

  Displays the static gateway value.
- void WebShowDhcpDeviceDnsServer (int sock, PCSTR url)

  Displays the active DNS server value.
- void WebShowStaticDeviceDnsServer (int sock, PCSTR url)

  Displays the static DNS server value.
- void WebShowDeviceName (int sock, PCSTR url)

  Displays the device's name form value.
- void WebShowUserValue (int sock, PCSTR url)

  Displays the current username form value.
- void WebShowUserPass (int sock, PCSTR url)

  Displays the current user password form value.
- void WebShowServer (int sock, PCSTR url)

  Displays the current destination server form value.
- void LastResult (int sock, PCSTR url)

  Displays the results of the request and any associated information with the request. If email messages were successfully returned, then links to those pages will be added to the page.
- void POP3_GetMessages (int session)

  POP3_GetMessages.
- void HandleGetMailPost (int sock, PostEvents event, const char *pName, const char *pValue)

  Handles the post for the get mail post request. This callback is called for each field of a post form.
- int HandleMailGet (int sock, HTTP_Request &pr)

  A GET callback function that displays the email saved on the file system in the browser.

24.180.1 Detailed Description

This code module contains the web functions for the POP3 mail using SSL example program.

24.180.2 Function Documentation

24.180.2.1 HandleGetMailPost()

```
void HandleGetMailPost (
            int sock,
            PostEvents event,
            const char * pName,
            const char * pValue )
```

Handles the post for the get mail post request. This callback is called for each field of a post form.

Parameters

sock	HTTP socket.
event	The kind of post event that is currently being handled with this callback.
pName	The name of the post element that is currently being handled.
pValue	The value of the post element that is currently being handled.

24.180.2.2 HandleMailGet()

```
int HandleMailGet (
            int sock,
            HTTP_Request & pr )
```

A GET callback function that displays the email saved on the file system in the browser.

Parameters

sock	HTTP socket.
pr	The HTTP request object associated with the request.

Return values

0	If the file was found and sent.
-1	If the file was not found or unable to be sent.

24.180.2.3 LastResult()

```
void LastResult (
            int sock,
            PCSTR url )
```

Displays the results of the request and any associated information with the request. If email messages were successfully returned, then links to those pages will be added to the page.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.4 POP3_GetMessages()

```
void POP3_GetMessages (
            int session )
```

POP3_GetMessages.

Parameters

session	The session number associated with the connection.

24.180.2.5 WebDisplayDhcpSelect()

```
void WebDisplayDhcpSelect (
            int sock,
            PCSTR url )
```

Displays if the network values are statically set or assigned by DHCP.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.6 WebShowDeviceName()

```
void WebShowDeviceName (
            int sock,
            PCSTR url )
```

Displays the device's name form value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.7 WebShowDhcpDeviceDnsServer()

```
void WebShowDhcpDeviceDnsServer (
            int sock,
            PCSTR url )
```

Displays the active DNS server value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.8 WebShowDhcpDeviceGateway()

```
void WebShowDhcpDeviceGateway (
            int sock,
            PCSTR url )
```

Displays the active gateway value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.9 WebShowDhcpDeviceIpAddress()

```
void WebShowDhcpDeviceIpAddress (
            int sock,
            PCSTR url )
```

Displays the active IP value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.10 WebShowDhcpDeviceIpMask()

```
void WebShowDhcpDeviceIpMask (
            int sock,
            PCSTR url )
```

Displays the active IP mask value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.11 WebShowServer()

```
void WebShowServer (
            int sock,
            PCSTR url )
```

Displays the current destination server form value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.12 WebShowStaticDeviceDnsServer()

```
void WebShowStaticDeviceDnsServer (
            int sock,
            PCSTR url )
```

Displays the static DNS server value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.13 WebShowStaticDeviceGateway()

```
void WebShowStaticDeviceGateway (
            int sock,
            PCSTR url )
```

Displays the static gateway value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.14 WebShowStaticDeviceIpAddress()

```
void WebShowStaticDeviceIpAddress (
            int sock,
            PCSTR url )
```

Displays the static IP value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.15 WebShowStaticDeviceIpMask()

```
void WebShowStaticDeviceIpMask (
            int sock,
            PCSTR url )
```

Displays the static IP mask value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.16 WebShowUserPass()

```
void WebShowUserPass (
            int sock,
            PCSTR url )
```

Displays the current user password form value.

Parameters

sock	HTTP Socket
url	Calling page

24.180.2.17 WebShowUserValue()

```
void WebShowUserValue (
            int sock,
            PCSTR url )
```

Displays the current username form value.

Parameters

sock	HTTP Socket
url	Calling page

24.181 MCF5441X/MOD5441X/Mod5441xFactoryApp/src/webfuncs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _WEBFUNCS_H_
00006 #define _WEBFUNCS_H_
00007
00008 const char FirmwareVersion[] = "MOD54415 Factory Demo Program v1.13 " __DATE__ "";
00009
00010 #endif /* _WEBFUNCS_H_ */
```

24.182 MCF5441X/NANO54415/NANO54415FactoryApp/src/webfuncs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _WEBFUNCS_H_
00006 #define _WEBFUNCS_H_
00007
00008 const char FirmwareVersion[] = "NANO54415 Factory Demo Program v1.0 " __DATE__ "";
00009
00010 #endif /* _WEBFUNCS_H_ */
```

24.183 MCF5441X/PulseGenerator-Counter/src/webfuncs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _WEBFUNCS_H_
00006 #define _WEBFUNCS_H_
00007
00008 const char FirmwareVersion[] = "v1.0 " __DATE__ "";
00009
00010 #endif /* _WEBFUNCS_H_ */
```

24.184 SAME70/MODM7AE70/MODM7AE70FactoryApp/src/webfuncs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _WEBFUNCS_H_
```

```
00006 #define _WEBFUNCS_H_
00007
00008 const char FirmwareVersion[] =
00009     "MODM7AE70 Factory Demo Program v1.3 "__DATE__
00010     "";
00011
00012 #endif /* _WEBFUNCS_H_ */
```

24.185 examples/PlatformSpecific/MCF5441X/RTC-External/src/rtc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*****************************************************************************
00006  * The following is a description of the basic time structure defined in the
00007  * time.h standard C library, which is used by the RTC functions.
00008  *
00009  * struct tm
00010  * {
00011  *    int tm_sec;     // Second (0-59)
00012  *    int tm_min;     // Minute (0-59)
00013  *    int tm_hour;    // Hour (0-23)
00014  *    int tm_mday;    // Day of the month (1-31)
00015  *    int tm_mon;     // Month of the year [January(0)-December(11)]
00016  *    int tm_year;    // Years since 1900
00017  *    int tm_wday;    // Day of the week [Sunday(0)-Saturday(6)]
00018  *    int tm_yday;    // Days since January 1st (0-365)
00019  *    int tm_isdst;   // Daylight Saving Time flag (in effect if greater than
00020  *                    // zero, not in effect if equal to zero, information not
00021  *                    // available if less than zero)
00022  * };
00023  *****************************************************************************/
00024
00025 #ifndef _RTC_H
00026 #define _RTC_H
00027
00028 #include <time.h>
00029
00030 #ifdef __cplusplus
00031 extern "C"
00032 {
00033 #endif /* __cplusplus */
00034
00041     int8_t rtcInitExternalRtc(uint8_t i2cModuleNum);
00042
00051     int rtcGetTime(struct tm &bts);
00052
00060     int rtcSetTime(struct tm &bts);
00061
00067     int rtcSetSystemTimeFromRtc(void);
00068
00074     int rtcSetRtcFromSystemTime(void);
00075
00076     void rtcSetRtcManual(int sec, int min, int hour, int mday, int mon, int year);   // , int wday,
      int yday);
00077
00078 #ifdef __cplusplus
00079 }
00080 #endif /* __cplusplus */
00081
00082 #endif /* _RTC_H */
```

24.186 examples/PlatformSpecific/SAME70/MODM7AE70/RTC-External/src/rtc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*****************************************************************************
00006  * The following is a description of the basic time structure defined in the
00007  * time.h standard C library, which is used by the RTC functions.
00008  *
00009  * struct tm
00010  * {
00011  *    int tm_sec;     // Second (0-59)
00012  *    int tm_min;     // Minute (0-59)
00013  *    int tm_hour;    // Hour (0-23)
00014  *    int tm_mday;    // Day of the month (1-31)
00015  *    int tm_mon;     // Month of the year [January(0)-December(11)]
```

```
00016  *    int tm_year;    // Years since 1900
00017  *    int tm_wday;    // Day of the week [Sunday(0)-Saturday(6)]
00018  *    int tm_yday;    // Days since January 1st (0-365)
00019  *    int tm_isdst;   // Daylight Saving Time flag (in effect if greater than
00020  *                    // zero, not in effect if equal to zero, information not
00021  *                    // available if less than zero)
00022  * };
00023  ******************************************************************************/
00024
00025 #ifndef _RTC_H
00026 #define _RTC_H
00027
00028 #include <time.h>
00029
00030 #ifdef __cplusplus
00031 extern "C"
00032 {
00033 #endif /* __cplusplus */
00034
00041     int8_t rtcInitExternalRtc(uint8_t i2cModuleNum);
00042
00051     int rtcGetTime(struct tm &bts);
00052
00060     int rtcSetTime(struct tm &bts);
00061
00067     int rtcSetSystemTimeFromRtc(void);
00068
00074     int rtcSetRtcFromSystemTime(void);
00075
00076     void rtcSetRtcManual(int sec, int min, int hour, int mday, int mon, int year);   // , int wday,
      int yday);
00077
00078 #ifdef __cplusplus
00079 }
00080 #endif /* __cplusplus */
00081
00082 #endif /* _RTC_H */
```

24.187 platform/MOD5441X/include/rtc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /******************************************************************************
00006  * The following is a description of the basic time structure defined in the
00007  * time.h standard C library, which is used by the RTC functions.
00008  *
00009  * struct tm
00010  * {
00011  *    int tm_sec;     // Second (0-59)
00012  *    int tm_min;     // Minute (0-59)
00013  *    int tm_hour;    // Hour (0-23)
00014  *    int tm_mday;    // Day of the month (1-31)
00015  *    int tm_mon;     // Month of the year [January(0)-December(11)]
00016  *    int tm_year;    // Years since 1900
00017  *    int tm_wday;    // Day of the week [Sunday(0)-Saturday(6)]
00018  *    int tm_yday;    // Days since January 1st (0-365)
00019  *    int tm_isdst;   // Daylight Saving Time flag (in effect if greater than
00020  *                    // zero, not in effect if equal to zero, information not
00021  *                    // available if less than zero)
00022  * };
00023  ******************************************************************************/
00024
00025 #ifndef _RTC_H
00026 #define _RTC_H
00027
00028 #include <time.h>
00029
00030 #ifdef __cplusplus
00031 extern "C"
00032 {
00033 #endif /* __cplusplus */
00034
00043     int RTCGetTime(struct tm &bts);
00044
00052     int RTCSetTime(struct tm &bts);
00053
00059     int RTCSetSystemFromRTCTime(void);
00060
00066     int RTCSetRTCfromSystemTime(void);
00067
00068 #ifdef __cplusplus
00069 }
00070 #endif /* __cplusplus */
```

```
00071
00072 #endif /* _RTC_H */
```

24.188 platform/NANO54415/include/rtc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*****************************************************************************
00006  * The following is a description of the basic time structure defined in the
00007  * time.h standard C library, which is used by the RTC functions.
00008  *
00009  * struct tm
00010  * {
00011  *    int tm_sec;     // Second (0-59)
00012  *    int tm_min;     // Minute (0-59)
00013  *    int tm_hour;    // Hour (0-23)
00014  *    int tm_mday;    // Day of the month (1-31)
00015  *    int tm_mon;     // Month of the year [January(0)-December(11)]
00016  *    int tm_year;    // Years since 1900
00017  *    int tm_wday;    // Day of the week [Sunday(0)-Saturday(6)]
00018  *    int tm_yday;    // Days since January 1st (0-365)
00019  *    int tm_isdst;   // Daylight Saving Time flag (in effect if greater than
00020  *                    // zero, not in effect if equal to zero, information not
00021  *                    // available if less than zero)
00022  * };
00023  *****************************************************************************/
00024
00025 #ifndef _RTC_H
00026 #define _RTC_H
00027
00028 #include <time.h>
00029
00030 #ifdef __cplusplus
00031 extern "C"
00032 {
00033 #endif /* __cplusplus */
00034
00043     int RTCGetTime(struct tm &bts);
00044
00052     int RTCSetTime(struct tm &bts);
00053
00059     int RTCSetSystemFromRTCTime(void);
00060
00066     int RTCSetRTCfromSystemTime(void);
00067
00068 #ifdef __cplusplus
00069 }
00070 #endif /* __cplusplus */
00071
00072 #endif /* _RTC_H */
```

24.189 platform/SB800EX/include/rtc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*****************************************************************************
00006  * The following is a description of the basic time structure defined in the
00007  * time.h standard C library, which is used by the RTC functions.
00008  *
00009  * struct tm
00010  * {
00011  *    int tm_sec;     // Second (0-59)
00012  *    int tm_min;     // Minute (0-59)
00013  *    int tm_hour;    // Hour (0-23)
00014  *    int tm_mday;    // Day of the month (1-31)
00015  *    int tm_mon;     // Month of the year [January(0)-December(11)]
00016  *    int tm_year;    // Years since 1900
00017  *    int tm_wday;    // Day of the week [Sunday(0)-Saturday(6)]
00018  *    int tm_yday;    // Days since January 1st (0-365)
00019  *    int tm_isdst;   // Daylight Saving Time flag (in effect if greater than
00020  *                    // zero, not in effect if equal to zero, information not
00021  *                    // available if less than zero)
00022  * };
00023  *****************************************************************************/
00024
00025 #ifndef _RTC_H
00026 #define _RTC_H
00027
```

```
00028 #include <time.h>
00029
00030 #ifdef __cplusplus
00031 extern "C"
00032 {
00033 #endif /* __cplusplus */
00034
00043     int RTCGetTime(struct tm &bts);
00044
00052     int RTCSetTime(struct tm &bts);
00053
00059     int RTCSetSystemFromRTCTime(void);
00060
00066     int RTCSetRTCfromSystemTime(void);
00067
00068 #ifdef __cplusplus
00069 }
00070 #endif /* __cplusplus */
00071
00072 #endif /* _RTC_H */
```

24.190 edma.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef ___EDMA_H
00006 #define ___EDMA_H
00007
00008 #define TCD_ATTR_8BIT_TRANS 0x0000
00009 #define TCD_ATTR_16BIT_TRANS 0x0101
00010
00011 #define TCD_XOFF_0BYTE 0x0000
00012 #define TCD_XOFF_1BYTE 0x0001
00013 #define TCD_XOFF_2BYTE 0x0002
00014
00015 #define TCD_XITER_CNT_MASK 0x7FFF
00016
00017 #define TCD_CSR_DONE 0x0080
00018 #define TCD_CSR_ACTIVE 0x0040
00019 #define TCD_CSR_MAJ_LINK 0x0020
00020 #define TCD_CSR_EN_SG 0x0010
00021 #define TCD_CSR_DREQ 0x0008
00022 #define TCD_CSR_INT_HALF 0x0004
00023 #define TCD_CSR_INT_MAJOR 0x0002
00024 #define TCD_CSR_START 0x0001
00025
00026 #define TCD_CSR_DISABLE_REQ 0xC008
00027 #define TCD_CSR_DREQ_INT_MAJOR 0xC00A
00028
00029 #define TCD_CITER_MAX_COUNT 0x7FFF
00030
00031 #endif /* ----- #ifndef ___EDMA_H  ----- */
```

24.191 wavPlayer.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef ___WAV_PLAYER_H
00006 #define ___WAV_PLAYER_H
00007
00008 #include <nbrtos.h>
00009 #include <sim.h>
00010
00011 // Comment out this macro to remove filesystem requirement
00012 #define WAV_PLAYER_FILESYSTEM (1)
00013
00014 // Sim1 Miscellaneous Control Register
00015 #define MISCCR2_DAC1SEL 0x0040   // Enable DAC1 drive output
00016 #define MISCCR2_DAC0SEL 0x0020   // Enable DAC0 drive output
00017 #define MISCCR2_ADCEN 0x0010     // Enable ADC 6-4 and 2-0
00018 #define MISCCR2_ADC7EN 0x0008    // Enable ADC 7
00019 #define MISCCR2_ADC3EN 0x0004    // Enable ADC 3
00020
00021 // ADC Calibration Register
00022 #define ADC_CAL_DAC1 0x0002   // Selects the source of the ADCA7 input as DAC1 output.
00023 #define ADC_CAL_DAC0 0x0001   // Selects the source of the ADCA3 input as DAC0 output.
00024
00025 // DAC Control Register
```

```
00026 #define DAC_CR_RESET 0x1101
00027 #define DAC_CR_FILT 0x1000
00028 #define DAC_CR_WMLVL_M 0x0300
00029 #define DAC_CR_WMLVL_S 8
00030 #define DAC_CR_DMA 0x0080
00031 #define DAC_CR_HSLS 0x0040
00032 #define DAC_CR_UP 0x0020
00033 #define DAC_CR_DOWN 0x0010
00034 #define DAC_CR_AUTO 0x0008
00035 #define DAC_CR_SYNC 0x0004
00036 #define DAC_CR_FMT 0x0002
00037 #define DAC_CR_PDN 0x0001   // Power down. 0 = power on, 1 = power down
00038
00039 // DAC Trigger Select Register DACTSR
00040 // Channels
00041 #define DAC1_CH0 0x0000   // Source channel 0
00042 #define DAC1_CH1 0x0800   // Source channel 1
00043 #define DAC1_CH2 0x1000   // Source channel 2
00044 #define DAC1_CH3 0x1800   // Source channel 3
00045
00046 #define DAC0_CH0 0x0000   // Source channel 0
00047 #define DAC0_CH1 0x0008   // Source channel 1
00048 #define DAC0_CH2 0x0010   // Source channel 2
00049 #define DAC0_CH3 0x0018   // Source channel 3
00050
00051 // DAC Trigger Sources
00052 #define DAC1_SRC_PWMA 0x0000   // PWMs
00053 #define DAC1_SRC_PWMB 0x0100
00054 #define DAC1_SRC_PWMX 0x0200
00055 #define DAC1_SRC_PWMTRIG1 0x0300
00056 #define DAC1_SRC_PWMTRIG0 0x0400
00057 #define DAC1_SRC_TnOUT 0x0500   // Timers
00058 #define DAC1_SRC_TnIN 0x0600
00059
00060 #define DAC0_SRC_PWMA 0x0000   // PWMs
00061 #define DAC0_SRC_PWMB 0x0001
00062 #define DAC0_SRC_PWMX 0x0002
00063 #define DAC0_SRC_PWMTRIG1 0x0003
00064 #define DAC0_SRC_PWMTRIG0 0x0004
00065 #define DAC0_SRC_TnOUT 0x0005   // Timers
00066 #define DAC0_SRC_TnIN 0x0006
00067
00068 #define EDMA_CH_DAC0 62
00069 #define EDMA_CH_DAC1 63
00070 #define EDMA_CH_DAC0_INT 0x40000000
00071 #define EDMA_CH_DAC1_INT 0x80000000
00072
00073 #define TCD_POOL_SIZE 20
00074 #define DAC_COUNT 2
00075 #define MAX_SAMPLE_RATE 100000
00076
00077 namespace WAVFILE
00078 {
00079 struct Chunk
00080 {
00081     char id[4];
00082     uint32_t size;
00083 };
00084 struct RIFFChunk
00085 {
00086     char ChunkID[4];   // 'R', 'I', 'F', 'F'
00087     uint32_t ChunkSize;
00088     char Format[4];   // 'W', 'A', 'V', 'E'
00089 };
00090 struct formatChunk
00091 {
00092     char ChunkID[4];   // 'f', 'm', 't', ' '
00093     uint32_t SubChunkSize;
00094     uint16_t AudioFormat;      // 1 = PCM, others are handled
00095     uint16_t ChannelCount;     // How many channels are recorded in this file?
00096     uint32_t SampleRate;       // Samples per second
00097     uint32_t ByteRate;         // Bytes per second = Sample Rate * ChannelCount * BitsPerSample
00098     uint16_t BlockAlign;       // Bytes per sample set = BitsPerSample * ChannelCount
00099     uint16_t BitsPerSample;    // Bits of data per sample
00100                                // Note: this is rounded up to multiples of 8-bits
00101                                // for BlockAlign and ByteRate
00102 };
00103 struct dataChunk
00104 {
00105     char ChunkID[4];   // 'd', 'a', 't', 'a'
00106     uint32_t SubChunkSize;
00107     uint8_t data[];   // Arbitrary length array definition
00108 };
00109
00110 // Structure definition for the 'canonical' WAV file
00111 struct WavFile
00112 {
```

```
00113     RIFFChunk riff;     // Leading chunk defining the data block as a RIFF chunk
00114     formatChunk fmt;    // First subchunk, describing the format of the data section
00115     dataChunk data;     // The actual data section
00116 };
00117 }   // namespace WAVFILE
00118
00119 class WavPlayer
00120 {
00121    public:
00122     // Public enum definitions
00123     enum playState
00124     {
00125         STATE_NO_DAC,
00126         STATE_NO_TIMER,
00127         STATE_NOT_LOADED,
00128         STATE_NOT_PROCESSED,
00129         STATE_BUFFER_RESET,
00130         STATE_PROCESSED,
00131         STATE_READY,
00132         STATE_PLAYING,
00133         STATE_UNMUTE,
00134         STATE_MUTE,
00135         STATE_PAUSED_UNMUTE,
00136         STATE_PAUSED_MUTE,
00137         STATE_PAUSED,
00138         STATE_FINISHED
00139     };
00140
00141     enum wavError
00142     {
00143         ERROR_NONE,
00144         ERROR_PLAYING,
00145         ERROR_FILE,
00146         ERROR_FILE_SIZE,
00147         ERROR_TYPE,
00148         ERROR_SHORT,
00149         ERROR_LONG,
00150         ERROR_FORMAT,
00151         ERROR_RATE,
00152         ERROR_BITSIZE,
00153         ERROR_SIZE_MISMATCH,
00154         ERROR_CHANNEL,
00155         ERROR_DACNUM,
00156         ERROR_TIMER,
00157         ERROR_IN_USE,
00158         ERROR_OTHER
00159     };
00160
00161    private:
00162     enum readMode
00163     {
00164         MODE_FILE,
00165         MODE_BUFFER,
00166         MODE_NONE
00167     };
00168
00169     struct channelControl
00170     {
00171         int dacNum;
00172         uint32_t dataRem;
00173         uint16_t transfersRem;
00174         uint16_t finalTransferSize;
00175         bool finished;
00176         volatile dacstruct *dac;
00177         volatile edma_tcdstruct *tcd;
00178     };
00179
00180     struct wavData
00181     {
00182         uint32_t SampleRate;
00183         uint16_t BitsPerSample;
00184         uint16_t ChannelCount;
00185         uint32_t dataSize;
00186     };
00187
00188     struct initialPlaySettings
00189     {
00190         uint16_t channel;
00191         uint16_t transfersRem;
00192         uint16_t transferSize;
00193     };
00194
00195     static WavPlayer *s_players[DAC_COUNT];
00196
00197     readMode m_mode;
00198     playState m_state;
00199     wavData m_wavInfo;
```

```
00200     channelControl m_channel[DAC_COUNT];
00201     initialPlaySettings m_initSettings[DAC_COUNT];
00202     uint32_t m_timer;
00203     int m_playsRem[DAC_COUNT];
00204     timerstruct timerSettings;
00205     WAVFILE::WavFile *m_pWav;
00206     WAVFILE::RIFFChunk *m_pWav_riff;    // Leading chunk defining the data block as a RIFF chunk
00207     WAVFILE::formatChunk *m_pWav_fmt;   // First subchunk, describing the format of the data section
00208     WAVFILE::dataChunk *m_pWav_data;    // The actual data section
00209     OS_SEM *m_finishedSem;
00210
00211     // Prepare the data for sending to DAC
00212     wavError PrepareData();
00213
00214     const uint8_t *FindChunk(const char *chunkID, const uint8_t *data, uint32_t dataLen);
00215     wavError ParseHeader();
00216     // Helper functions for swapping endianess of header segments
00217     void PrepChunk_RIFF();
00218     void PrepChunk_fmt();
00219     void PrepChunk_data();
00220
00221     void SetLengths();
00222     void ConfigDAC();
00223     void ConfigDMA();
00224     void ConfigTimer();
00225     void ISR(int channelNum);
00226     void SoftMuteISR();
00227
00228   public:
00229     WavPlayer();
00230     ~WavPlayer();
00231
00232     static void RunISR();
00233     static void RunSoftMuteISR(uint32_t vector);
00234
00235 #ifdef WAV_PLAYER_FILESYSTEM
00236     /* OpenFile
00237      * Opens and readies a WAV file from disk
00238      *
00239      *  Args:
00240      *  - fileName: pointer to C string containing the name of the file to be opened
00241      *  - dataBuffer: pointer to a buffer to load the file into
00242      *  - bufferSize: the size of the buffer being used
00243      *
00244      *  Returns:
00245      *  - ERROR_NONE:   File loaded successfully
00246      *  - Other:        Failure while loading file, value indicates type
00247      */
00248     wavError OpenFile(const char *fileName, uint8_t *dataBuffer, uint32_t bufferSize);
00249 #endif /* #ifdef WAV_PLAYER_FILESYSTEM */
00250     /* OpenBuffer
00251      * Readies a WAV file stored in a buffer.
00252      * WARNING: This function mutates the passed buffer if successful. Until
00253      * either a new WAV file is loaded, ResetBuffer() is called, or the
00254      * WavPlayer is destroyed, the buffer should not be read or modified.
00255      *
00256      *  Args:
00257      *  - data: pointer to a buffer containing a WAV file
00258      *
00259      *  Returns:
00260      *  - ERROR_NONE:   File loaded successfully
00261      *  - Other:        Failure while loading file, value indicates type
00262      */
00263     wavError OpenBuffer(uint8_t *data);
00264
00265     /* SetChannelDAC
00266      * Sets the association between an audio channel and the DAC that is used
00267      * to play it. Note: Only 2 channel WAVs are supported.
00268      *
00269      *  Args:
00270      *  - channel:  WAV sample channel to read from
00271      *  - dacNum:   DAC channel to output to
00272      *
00273      *  Returns:
00274      *  - ERROR_NONE:   Channel Associated
00275      *  - Other:        Failure, value indicates type
00276      */
00277     wavError SetChannelDAC(int channel = 0, int dacNum = 0);
00278     /* SetTimer
00279      * Selects which DMA Timer is used as the trigger source for the DAC(s)
00280      *
00281      *  Args:
00282      *  - timerNum: the dma timer to use
00283      *
00284      *  Returns:
00285      *  - ERROR_NONE:   Channel Associated
00286      *  - Other:        Failure, value indicates type
```

```
00287      */
00288     wavError SetTimer(int timerNum = 3);
00289
00290     /* Play
00291      * Plays the previously loaded WAV file
00292      *
00293      *  Args:
00294      *  - wavFinishedSem:   Optional semaphore posted to upon completion
00295      *
00296      *  Returns:
00297      *  NONE
00298      */
00299     wavError Play(OS_SEM *wavFinishedSem = NULL);
00300
00301     /* Loop
00302      * Repeatedly plays the open file.
00303      *
00304      *  Args:
00305      *  - playCount: the number of times to play the file, 0 = loop forever
00306      *
00307      *  Returns:
00308      *
00309      */
00310     wavError Loop(uint32_t playCount = 0, OS_SEM *wavFinishedSem = NULL);
00311
00312     /* Stop
00313      * Immediately stops playback
00314      *
00315      */
00316     wavError Stop();
00317
00318     /* StopGraceful
00319      * Stops playback once current loop iteration is finished.
00320      */
00321     wavError StopGraceful();
00322
00323     wavError Pause();
00324
00325     wavError Resume();
00326
00327     /* ResetBuffer
00328      * Reset the data buffer back to its unmutated state. Only needed when the
00329      * WAV file was loaded using OpenBuffer and the buffer is not reverted from
00330      * other sources.
00331      *
00332      *  Args:
00333      *  NONE
00334      *
00335      *  Returns:
00336      *  NONE
00337      */
00338     wavError ResetBuffer();
00339
00340     /* GetState
00341      * Gets the current state of the WavPlayer.
00342      *
00343      *  Args:
00344      *  NONE
00345      *
00346      *  Returns:
00347      *  NONE
00348      */
00349     playState GetState();
00350 };
00351
00352 #endif /* ----- #ifndef ___WAV_PLAYER_H  ----- */
```

24.192 PlatformSpecific/SAME70/GpioServer/src/analog.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __ANALOG_H__
00006 #define __ANALOG_H__
00007
00008 struct ch_t
00009 {
00010     uint8_t pinNum;   // assumes P2 header for MODM7AE70, and J1 header for SBE70LC
00011     uint8_t module;
00012     uint8_t ch;
00013 };
00014
00015 #ifdef SBE70LC
00016 #define ADC_CH_COUNT (5)
```

```
00017 ch_t ch[ADC_CH_COUNT] = {{13, 0, 1}, {11, 0, 10}, {5, 0, 2}, {14, 0, 5}, {10, 1, 0}};
00018 #elif (defined MODM7AE70)
00019 #define ADC_CH_COUNT (7)
00020 ch_t ch[ADC_CH_COUNT] = {{7, 0, 0}, {21, 0, 1}, {38, 0, 2}, {29, 0, 5}, {8, 0, 6}, {3, 0, 10}, {6, 1,
      3}};
00021 #endif
00022
00023 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00024
00025 // Enable or disable individual AFEC channels
00026 void ADConfigCh(unsigned nAfec, unsigned ch, bool enable);
00027
00028 void StartAD();   // Start A/D conversion set.
00029
00030 bool ADDone();   // Return true if the conversion is complete
00031
00032 uint16_t GetADResult(unsigned nAfec, unsigned ch);   // Return the AD Result
00033
00034 #endif   // end __ANALOG_H__
```

24.193 SSH/SecureSerToEthFactoryApp/src/analog.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __ANALOG_H__
00006 #define __ANALOG_H__
00007
00008 #ifdef SAME70
00009 struct ch_t
00010 {
00011     uint8_t pinNum;   // assumes P2 header for MODM7AE70, and J1 header for SBE70LC
00012     uint8_t module;
00013     uint8_t ch;
00014 };
00015
00016 #ifdef SBE70LC
00017 #define ADC_CH_COUNT (5)
00018 ch_t ch[ADC_CH_COUNT] = {{13, 0, 1}, {11, 0, 10}, {5, 0, 2}, {14, 0, 5}, {10, 1, 0}};
00019 #elif (defined MODM7AE70)
00020 #define ADC_CH_COUNT (7)
00021 ch_t ch[ADC_CH_COUNT] = {{7, 0, 0}, {21, 0, 1}, {38, 0, 2}, {29, 0, 5}, {8, 0, 6}, {3, 0, 10}, {6, 1,
      3}};
00022 #endif
00023
00024 void InitSingleEndAD();   // Setup the A/D converter to be ready to run
00025
00026 // Enable or disable individual AFEC channels
00027 void ADConfigCh(unsigned nAfec, unsigned ch, bool enable);
00028
00029 void StartAD();   // Start A/D conversion set.
00030
00031 bool ADDone();   // Return true if the conversion is complete
00032
00033 uint16_t GetADResult(unsigned nAfec, unsigned ch);   // Return the AD Result
00034 #endif                                               // SAME70
00035
00036 #endif   // end __ANALOG_H__
```

24.194 PlatformSpecific/SAME70/GpioServer/src/gpioserver.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __GPIO_SERVER_H__
00006 #define __GPIO_SERVER_H__
00007
00008 #include <string.h>
00009 #include <stdlib.h>
00010 #include <tcp.h>
00011 #include <command.h>
00012 #include <pins.h>
00013 #include <cpu_pins.h>
00014 #include <fdprintf.h>
00015
00016 #define CMD_ERR_NOT_GPIO_MODE -1
00017 #define CMD_ERR_BAD_PIN_NUM -2
00018 #define CMD_ERR_SYNTAX -3
00019
00020 extern const char *AppName;
```

```
00021 extern const char *VersionNumber;
00022
00023 /* returns true if the pin number is valid within the pin header, retruns false if the pin is not
      valid */
00024 bool pinIsValid(uint8_t pinNumber);
00025
00026 /* returns the index in the analog "ch" struct cooresponding to the analogPinNum, returns -1 if the
00027    analogPinNum is not found within the "ch" struct (therefore is not a valid analog pin number). */
00028 int8_t getAnalogPinIndex(uint8_t analogPinNum);
00029
00030 /* returns true if the pin output was set, false if the pin output was not set */
00031 bool SetPinOutput(uint8_t pinNumber, uint8_t output, FILE *fp);
00032
00033 /* Set the pin as a digital input */
00034 bool SetPinAsInput(uint8_t pinNumber, FILE *fp);
00035
00036 /* Read the digital pin value */
00037 uint32_t ReadPin(uint8_t pinNumber);
00038
00039 /* Read the analog value of the cooresponding ADC channel in the "ch" struct. The index is
00040    obtained by using the getAnalogPinIndex() function.  */
00041 uint16_t ReadAnalogPin(uint8_t channelIndex);
00042
00043 /* Parse a whole command string for the pin number. returns the number to the pinNumber variable by
      reference.
00044    returns true if parsing the pin number was successful. returns false otherwise. The function
00045    prints the error response to the fp variable in the case that parsing fails.
00046 */
00047 bool parsePinNumber(const char *command, uint8_t &pinNumber, FILE *fp);
00048
00049 /* Command parser. returns CMD_CLOSE if requested to close the connection, returns CMD_OK otherwise */
00050 int32_t ProcessGpioCommand(const char *command, FILE *fp);
00051
00052 /* The User Authentication Callback */
00053 int ProcessAuth(const char *name, const char *pass);
00054
00055 /* The Command Processing Callback */
00056 int ProcessCommand(const char *command, FILE *fp, void *pData);
00057
00058 /* The Connection Processing Callback */
00059 void *ProcessConnect(FILE *fp);
00060
00061 /* The Prompt Processing Callback */
00062 void ProcessPrompt(FILE *fp, void *pData);
00063
00064 /* The Disconnect Processing Callback */
00065 void ProcessDisconnect(FILE *fp, int cause, void *pData);
00066
00067 #endif   // end __GPIO_SERVER_H__
```

24.195 SSH/SecureSerToEthFactoryApp/src/gpioserver.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __GPIO_SERVER_H__
00006 #define __GPIO_SERVER_H__
00007
00008 #include <string.h>
00009 #include <stdlib.h>
00010 #include <tcp.h>
00011 #include <command.h>
00012 #include <pins.h>
00013 #include <cpu_pins.h>
00014 #include <fdprintf.h>
00015
00016 #define SERIALPORT_TO_USE (0)
00017 #define BAUDRATE_TO_USE (115200)
00018 #define STOP_BITS (1)
00019 #define DATA_BITS (8)
00020 #define TCP_PORT_TO_USE (23)   // The Telnet port
00021 #define IDLE_TIMEOUT (60 * TICKS_PER_SECOND)
00022 #define DO_TELNET_PROCESSING (true)
00023 #define MAX_TCP_CONNECTIONS (5)
00024
00025 #define CMD_ERR_NOT_GPIO_MODE -1
00026 #define CMD_ERR_BAD_PIN_NUM -2
00027 #define CMD_ERR_SYNTAX -3
00028
00029 extern const char FirmwareVersion[];
00030
00031 #ifdef SUPPORTED_GPIO_SERVER_PLATFORM
00032 /* returns true if the pin number is valid within the pin header, retruns false if the pin is not
      valid */
```

```
00033 bool pinIsValid(uint8_t pinNumber);
00034
00035 /* returns the index in the analog "ch" struct cooresponding to the analogPinNum, returns -1 if the
00036    analogPinNum is not found within the "ch" struct (therefore is not a valid analog pin number). */
00037 int8_t getAnalogPinIndex(uint8_t analogPinNum);
00038
00039 /* returns true if the pin output was set, false if the pin output was not set */
00040 bool SetPinOutput(uint8_t pinNumber, uint8_t output, FILE *fp);
00041
00042 /* Set the pin as a digital input */
00043 bool SetPinAsInput(uint8_t pinNumber, FILE *fp);
00044
00045 /* Read the digital pin value */
00046 uint32_t ReadPin(uint8_t pinNumber);
00047
00048 /* Read the analog value of the cooresponding ADC channel in the "ch" struct. The index is
00049    obtained by using the getAnalogPinIndex() function.  */
00050 uint16_t ReadAnalogPin(uint8_t channelIndex);
00051
00052 /* Parse a whole command string for the pin number. returns the number to the pinNumber variable by
      reference.
00053    returns true if parsing the pin number was successful. returns false otherwise. The function
00054    prints the error response to the fp variable in the case that parsing fails.
00055 */
00056 bool parsePinNumber(const char *command, uint8_t &pinNumber, FILE *fp);
00057
00058 /* Command parser. returns CMD_CLOSE if requested to close the connection, returns CMD_OK otherwise */
00059 int ProcessGpioCommand(const char *command, FILE *fp, void *pData);
00060
00061 /* The User Authentication Callback */
00062 int ProcessAuth(const char *name, const char *pass);
00063
00064 /* The Command Processing Callback */
00065 int ProcessCommand(const char *command, FILE *fp, void *pData);
00066
00067 /* The Connection Processing Callback */
00068 void *ProcessConnect(FILE *fp);
00069
00070 /* The Prompt Processing Callback */
00071 void ProcessPrompt(FILE *fp, void *pData);
00072
00073 /* The Disconnect Processing Callback */
00074 void ProcessDisconnect(FILE *fp, int cause, void *pData);
00075 #endif   // SUPPORTED_GPIO_SERVER_PLATFORM
00076
00077 #endif   // end __GPIO_SERVER_H__
```

24.196 ebi_pager.h

```
00001 #ifndef __EBI_PAGER_H
00002 #define __EBI_PAGER_H
00003
00004 #include <predef.h>
00005 #include <pins.h>
00006
00007 extern volatile uint8_t ebi_0_base[];
00008 extern volatile uint8_t ebi_1_base[];
00009 extern volatile uint8_t ebi_2_base[];
00010 extern volatile uint8_t ebi_3_base[];
00011
00025 void ConfigAddressPager(uint32_t csNum, uint32_t pageSize, PinIO *pageSelectPins, uint32_t
      pageSelectPinCount);
00026
00027 #endif /* ----- #ifndef __EBI_PAGER_H  ----- */
```

24.197 hd44780.h

```
00001 #ifndef __HD44780_H
00002 #define __HD44780_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 #include <predef.h>
00008
00009 class HD44780_LCD_Bus
00010 {
00011     volatile uint8_t bus[2];
00012     class reg
00013     {
00014         uint8_t addr;
00015         volatile uint8_t *lcd;
```

```
00016
00017         public:
00018          reg(uint8_t addr, volatile uint8_t *lcd) : addr(addr), lcd(lcd) {}
00019          operator uint8_t() { return read(); }
00020          uint8_t operator=(uint8_t val)
00021          {
00022              write(val);
00023              return val;
00024          }
00025
00026          uint8_t read() { return lcd[addr]; }
00027          void write(uint8_t val)
00028          {
00029              while (lcd[0] & 0x80)
00030              {
00031                  asm("dsb");
00032              }
00033              lcd[addr] = val;
00034          }
00035      };
00036
00037    public:
00038      HD44780_LCD_Bus();
00039
00040      void init(uint8_t cols, uint8_t rows);
00041      void putchar(char c);
00042      void putstr(const char *str);
00043      void setCursor(int row, int col);
00044      void clr();
00045
00046      reg operator[](int i) { return reg(i & 0x1, bus); }
00047      friend class HD44780_LCD;
00048 };
00049
00050 class HD44780_LCD
00051 {
00052    public:
00053      enum cursorDisp_t
00054      {
00055          CURSOR_OFF = 0,
00056          CURSOR_ON = 2,
00057          CURSOR_BLINK = 3
00058      };
00059
00060    private:
00061      HD44780_LCD_Bus &bus;
00062      uint8_t cols;
00063      uint8_t rows;
00064      cursorDisp_t cursorDisp;
00065      bool dispState;
00066
00067    public:
00068      HD44780_LCD(HD44780_LCD_Bus &bus, uint8_t cols, uint8_t rows);
00069
00070      void init() { bus.init(cols, rows); }
00071      void putchar(char c) { bus.putchar(c); }
00072      void putstr(const char *str) { bus.putstr(str); }
00073      void setCursor(int row, int col) { bus.setCursor(row, col); }
00074      void enableCursor(cursorDisp_t disp);
00075      void enableDisplay(bool enable);
00076      void clr();
00077
00078      HD44780_LCD_Bus::reg operator[](int i) { return bus[i]; }
00079 };
00080
00081 #endif /* ----- #ifndef __HD44780_H  ----- */
```

24.198 ssc_i2s.h

```
00001 #ifndef __SSC_I2S_H
00002 #define __SSC_I2S_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 #include <predef.h>
00008 #include <sim.h> /*on-chip register definitions*/
00009 #include <pins.h>
00010 #include <nbrtos.h>
00011 #include <xdmac.h>
00012
00013 // class SSC_I2S : private Ssc {
00014 //
00015 //
00016 // public:
```

```
00017 //     inline void Reset() { SSC_CR = SSC_CR_SWRST; }
00018 //     inline void EnableTx(bool enable) {
00019 //         SSC_CR = (enable ? SSC_CR_TXEN : SSC_CR_TXDIS);
00020 //     }
00021 //     inline void EnableRx(bool enable) {
00022 //         SSC_CR = (enable ? SSC_CR_RXEN : SSC_CR_RXDIS);
00023 //     }
00024 //     inline uint16_t SetClkDiv(uint16_t div) {
00025 //         return SSC_CMR = div & SSC_CMR_DIV_Msk;
00026 //     }
00027 //
00028 //     static Ssc *get() { return reinterpret_cast<SSC_I2S*>(SSC); }
00029 //     Ssc &GetRegisers() { return *this; }
00030 //
00031 // } __attribute__((packed));
00032 //
00033 // static_assert( sizeof (SSC_I2S == sizeof(Ssc) );
00034 //
00035
00036 typedef void (*SSC_BufferDoneFn_t)(void *buffer, bool valid);
00037 /****************************************/
00038 /*      SSC Configuration Enums         */
00039 /****************************************/
00043 enum clkSrc_t
00044 {
00045     CLK_SRC_MCK,
00046     CLK_SRC_RK,
00047     CLK_SRC_TK
00048 };
00049
00073 enum startCond_t
00074 {
00075     START_CONTINUOUS,
00076     START_SYNC_RX_TX,
00077     START_FRAME_LOW,
00078     START_FRAME_HIGH,
00079     START_FRAME_FALLING,
00080     START_FRAME_RISING,
00081     START_FRAME_LEVEL,
00082     START_FRAME_EDGE,
00083     START_CMP_0
00084 };
00085
00089 enum clkGate_t
00090 {
00091     CLK_GATE_CONTINUOUS,
00092     CLK_GATE_FRAME_LOW,
00093     CLK_GATE_FRAME_HIGH,
00094 };
00095
00099 enum dataValid_t
00100 {
00101     DATA_VALID_RISING,
00102     DATA_VALID_FALLING,
00103 };
00104
00108 enum frameEdge_t
00109 {
00110     FRAME_SYNC_RISING,
00111     FRAME_SYNC_FALLING
00112 };
00113
00117 enum clkOut_t
00118 {
00119     CLK_OUT_INPUT,
00121     CLK_OUT_CONTINUOUS,
00122     CLK_OUT_TRANSFER
00123 };
00124
00128 enum frameSyncOut_t
00129 {
00130     FRAME_SYNC_INPUT,
00132     FRAME_SYNC_NEGATIVE,
00134     FRAME_SYNC_POSITIVE,
00136     FRAME_SYNC_LOW,
00137     FRAME_SYNC_HIGH,
00138     FRAME_SYNC_TOGGLE
00139 };
00140
00144 enum bitOrder_t
00145 {
00146     LEAST_SIG_FIRST,
00147     MOST_SIG_FIRST
00148 };
00149
00160 enum bufferDepletionBehavior_t
00161 {
```

```
00162     DEPLETED_PAUSE,
00163     DEPLETED_REPEAT_LAST,
00164 };
00165
00166 /*************************************/
00167 /*      SSC Configuration Struct       */
00168 /*************************************/
00169
00212 struct SSC_rxtx_cfg_t
00213 {
00214     bool enable{false};
00215     uint8_t period{64};
00216     uint8_t startDly{1};
00217     startCond_t startCond{START_FRAME_EDGE};
00218     clkGate_t clkGate{CLK_GATE_CONTINUOUS};
00219     dataValid_t dataValid{DATA_VALID_RISING};
00220     clkOut_t clkOut{CLK_OUT_INPUT};
00221     clkSrc_t clkSrc{CLK_SRC_RK};
00222
00223     uint8_t syncLen{0};
00224     frameEdge_t syncEdge{FRAME_SYNC_FALLING};
00225     bool syncDataEnabled{false};
00226     frameSyncOut_t syncOut{FRAME_SYNC_INPUT};
00227     uint8_t wordsPerFrame{1};
00228     bitOrder_t bitOrder{MOST_SIG_FIRST};
00229     bool lineIdleState{1};
00230     uint8_t bitsPerWord{24};
00231
00232     bufferDepletionBehavior_t depletionBehavior{DEPLETED_PAUSE};
00233 };
00234
00248 struct SSC_cfg_t
00249 {
00250     uint16_t clkDiv{150};
00251
00252     SSC_rxtx_cfg_t rx;
00253     SSC_rxtx_cfg_t tx;
00254 };
00255
00296 extern const SSC_cfg_t SSC_I2S_SLAVE_24_TXRX_RK;
00297 extern const SSC_cfg_t SSC_I2S_SLAVE_24_TXRX_TK;
00298 extern const SSC_cfg_t SSC_I2S_SLAVE_24_TXRX_TKRK;
00299 extern const SSC_cfg_t SSC_I2S_SLAVE_16_TXRX_RK;
00300 extern const SSC_cfg_t SSC_I2S_SLAVE_16_TXRX_TK;
00301 extern const SSC_cfg_t SSC_I2S_SLAVE_16_TXRX_TKRK;
00302 extern const SSC_cfg_t SSC_I2S_SLAVE_8_TXRX_RK;
00303 extern const SSC_cfg_t SSC_I2S_SLAVE_8_TXRX_TK;
00304 extern const SSC_cfg_t SSC_I2S_SLAVE_8_TXRX_TKRK;
00305
00306 extern const SSC_cfg_t SSC_LJUST_SLAVE_24_TXRX_RK;
00307 extern const SSC_cfg_t SSC_LJUST_SLAVE_24_TXRX_TK;
00308 extern const SSC_cfg_t SSC_LJUST_SLAVE_24_TXRX_TKRK;
00309 extern const SSC_cfg_t SSC_LJUST_SLAVE_16_TXRX_RK;
00310 extern const SSC_cfg_t SSC_LJUST_SLAVE_16_TXRX_TK;
00311 extern const SSC_cfg_t SSC_LJUST_SLAVE_16_TXRX_TKRK;
00312 extern const SSC_cfg_t SSC_LJUST_SLAVE_8_TXRX_RK;
00313 extern const SSC_cfg_t SSC_LJUST_SLAVE_8_TXRX_TK;
00314 extern const SSC_cfg_t SSC_LJUST_SLAVE_8_TXRX_TKRK;
00315
00316 /*************************************/
00317 /*      SSC Driver Context Object      */
00318 /*************************************/
00319
00326 class SSCCtx_t
00327 {
00328   public:
00329     enum ctxState_t
00330     {
00331         CTX_STATE_STOPPED,
00332         CTX_STATE_INIT,
00333         CTX_STATE_TX,
00334         CTX_STATE_RX,
00335         CTX_STATE_TXRX,
00336     };
00337
00338   private:
00339     XdmaCh_t *txCh;
00340     XdmaCh_t *rxCh;
00341     OS_SEM txSem;
00342     OS_SEM rxSem;
00343     SSC_BufferDoneFn_t txBufDone;
00344     SSC_BufferDoneFn_t rxBufDone;
00345     uint32_t txReadyCount;
00346     uint32_t rxReadyCount;
00347     uint32_t nTxIrq;
00348     uint32_t nTxErr;
00349     uint32_t nTxPost;
```

```
00350     uint32_t nTxCall;
00351     uint32_t nTxStart;
00352     uint32_t nTxAdd;
00353 
00354     uint32_t nRxIrq;
00355     uint32_t nRxErr;
00356     uint32_t nRxPost;
00357     uint32_t nRxCall;
00358     uint32_t nRxStart;
00359     uint32_t nRxAdd;
00360 
00361     Desc0Ring TxRing;
00362     Desc0Ring RxRing;
00363     uint8_t rxByteWidth;
00364     uint8_t txByteWidth;
00365 
00366     ctxState_t state;
00367     void initBDs(bool loopRx, bool loopTx);
00368     void initHw_Pins(const SSC_cfg_t &cfg);
00369     void initHw_SSC(const SSC_cfg_t &cfg);
00370     void initHw_DMA(const SSC_cfg_t &cfg);
00371 
00372    public:
00379     int Init(const SSC_cfg_t &cfg);
00383     void Shutdown();
00384 
00391     int getCurrentConfig(SSC_cfg_t &cfg);
00396     ctxState_t getState();
00397 
00409     int TransmitBuffer(void *buffer, uint32_t bufferLen, bool waitIfNeeded);
00421     int ReadyReceiveBuffer(void *buffer, uint32_t bufferLen, bool waitIfNeeded);
00422 
00426     void RegisterTxBufferDoneCB(SSC_BufferDoneFn_t cb);
00430     void RegisterRxBufferDoneCB(SSC_BufferDoneFn_t cb);
00431 
00432     void TxIsr();
00433     void RxIsr();
00434 
00435     void dump();
00436 };
00437 
00438 extern SSCCtx_t ssc;
00439 
00440 #endif /* ----- #ifndef __SSC_I2S_H  ----- */
```

24.199 wm8904.h

```
00001 #ifndef __WM8904_H
00002 #define __WM8904_H
00003 /*NB_REVISION*/
00004 
00005 /*NB_COPYRIGHT*/
00006 
00007 #include <predef.h>
00008 #include <stdio.h>
00009 #include <init.h>
00010 #include <sim.h> /*on-chip register definitions*/
00011 #include <pins.h>
00012 #include <nbrtos.h>
00013 #include <i2c.h>
00014 #include "ssc_i2s.h"
00015 
00016 #include "wm8904_reg.h"
00017 
00022 class WM8904
00023 {
00024    public:
00025     struct Reg
00026     {
00027         enum RegAddr_t
00028         {
00029             SWRST_ID = 0x00,
00030             BIAS_CTL0 = 0x04,
00031             VMID_CTL0 = 0x05,
00032             MIC_BIAS_CTL0 = 0x06,
00033             MIC_BIAS_CTL1 = 0x07,
00034             ANA_ADC0 = 0x0A,
00035             PWR_MGMNT0 = 0x0C,
00036             PWR_MGMNT2 = 0x0E,
00037             PWR_MGMNT3 = 0x0F,
00038             PWR_MGMNT6 = 0x12,
00039             CLK_RATES0 = 0x14,
00040             CLK_RATES1 = 0x15,
00041             CLK_RATES2 = 0x16,
00042             AUDIO_INTF0 = 0x18,
```

```
00043             AUDIO_INTF1 = 0x19,
00044             AUDIO_INTF2 = 0x1A,
00045             AUDIO_INTF3 = 0x1B,
00046             DAC_DGTL_VOL_L = 0x1E,
00047             DAC_DGTL_VOL_R = 0x1F,
00048             DAC_DGTL0 = 0x20,
00049             DAC_DGTL1 = 0x21,
00050             ADC_DGTL_VOL_L = 0x24,
00051             ADC_DGTL_VOL_R = 0x25,
00052             ADC_DGTL0 = 0x26,
00053             DGTL_MIC0 = 0x27,
00054             DRC0 = 0x28,
00055             DRC1 = 0x29,
00056             DRC2 = 0x2A,
00057             DRC3 = 0x2B,
00058             ANA_L_IN0 = 0x2C,
00059             ANA_R_IN0 = 0x2D,
00060             ANA_L_IN1 = 0x2E,
00061             ANA_R_IN1 = 0x2F,
00062             ANA_OUT1_L = 0x39,
00063             ANA_OUT1_R = 0x3A,
00064             ANA_OUT2_L = 0x3B,
00065             ANA_OUT2_R = 0x3C,
00066             ANA_OUT12_ZC = 0x3D,
00067             DC_SERVO0 = 0x43,
00068             DC_SERVO1 = 0x44,
00069             DC_SERVO2 = 0x45,
00070             DC_SERVO4 = 0x47,
00071             DC_SERVO5 = 0x48,
00072             DC_SERVO6 = 0x49,
00073             DC_SERVO7 = 0x4A,
00074             DC_SERVO8 = 0x4B,
00075             DC_SERVO9 = 0x4C,
00076             DC_SERVO_RDBK0 = 0x4D,
00077             ANA_HP0 = 0x5A,
00078             ANA_LINEOUT0 = 0x5E,
00079             CHARGE_PUMP0 = 0x62,
00080             CLASS_W0 = 0x68,
00081             WR_SEQ0 = 0x6C,
00082             WR_SEQ1 = 0x6D,
00083             WR_SEQ2 = 0x6E,
00084             WR_SEQ3 = 0x6F,
00085             WR_SEQ4 = 0x70,
00086             FLL_CTL1 = 0x74,
00087             FLL_CTL2 = 0x75,
00088             FLL_CTL3 = 0x76,
00089             FLL_CTL4 = 0x77,
00090             FLL_CTL5 = 0x78,
00091             GPIO_CTL1 = 0x79,
00092             GPIO_CTL2 = 0x7A,
00093             GPIO_CTL3 = 0x7B,
00094             GPIO_CTL4 = 0x7C,
00095             DGTL_PULLS = 0x7E,
00096             IRQ_STAT = 0x7F,
00097             IRQ_STAT_MASK = 0x80,
00098             IRQ_POL = 0x81,
00099             IRQ_DEBOUNCE = 0x82,
00100             EQ1 = 0x86,
00101             EQ2 = 0x87,
00102             EQ3 = 0x88,
00103             EQ4 = 0x89,
00104             EQ5 = 0x8A,
00105             EQ6 = 0x8B,
00106             EQ7 = 0x8C,
00107             EQ8 = 0x8D,
00108             EQ9 = 0x8E,
00109             EQ10 = 0x8F,
00110             EQ11 = 0x90,
00111             EQ12 = 0x91,
00112             EQ13 = 0x92,
00113             EQ14 = 0x93,
00114             EQ15 = 0x94,
00115             EQ16 = 0x95,
00116             EQ17 = 0x96,
00117             EQ18 = 0x97,
00118             EQ19 = 0x98,
00119             EQ20 = 0x99,
00120             EQ21 = 0x9A,
00121             EQ22 = 0x9B,
00122             EQ23 = 0x9C,
00123             EQ24 = 0x9D,
00124             ADC_TEST0 = 0xC6,
00125             FLL_NCO_TEST0 = 0xF7,
00126             FLL_NCO_TEST1 = 0xF8
00127         };
00128         struct cmd_t
00129         {
```

```
00130             RegAddr_t reg;
00131             uint16_t val;
00132             uint16_t dly;
00133             cmd_t &operator=(const cmd_t rhs)
00134             {
00135                 reg = rhs.reg;
00136                 val = rhs.val;
00137                 dly = rhs.dly;
00138                 return *this;
00139             }
00140         };
00141         static cmd_t init_cmds_0[];
00142         static cmd_t init_cmds_1[];
00143         static cmd_t init_cmds_2[];
00144         static cmd_t init_cmds_3[];
00145         static cmd_t init_cmds_4[];
00146         static cmd_t init_cmds_5[];
00147         static cmd_t init_cmds_6[];
00148     };
00149
00150     enum DataLen_t
00151     {
00152         DATA_LEN_16,
00153         DATA_LEN_24,
00154         DATA_LEN_32
00155     };
00156
00157     enum DataFmt_t
00158     {
00159         DATA_FMT_16_L_JUSTIFIED,
00160         DATA_FMT_16_I2S,
00161         DATA_FMT_32_L_JUSTIFIED,
00162         DATA_FMT_32_I2S
00163     };
00164
00165     enum AudioChSelect_t
00166     {
00167         CH_SELECT_LEFT,
00168         CH_SELECT_RIGHT,
00169         CH_SELECT_LEFT_RIGHT
00170     };
00171
00172     enum AudioOutSelect_t
00173     {
00174         AUDIO_OUT_NONE,
00175         AUDIO_OUT_HP,
00176         AUDIO_OUT_LINE,
00177         AUDIO_OUT_SPKR
00178     };
00179
00180     enum AudioInSelect_t
00181     {
00182         AUDIO_IN_NONE,
00183         AUDIO_IN_1,
00184         AUDIO_IN_2,
00185         AUDIO_IN_3,
00186     };
00187
00188     struct cfg_t
00189     {
00190         uint16_t sampleRate;
00191         uint32_t refClkRate;
00192         DataFmt_t dataFormat;
00193         AudioChSelect_t inCh;
00194         AudioChSelect_t outCh;
00195         AudioInSelect_t inSrc;
00196         AudioOutSelect_t outSrc;
00197
00198         uint8_t initMicGain;
00199         uint8_t initOutVol;
00200     };
00201
00202   private:
00203     I2C &twi;
00204
00205     void InitInput(const cfg_t &cfg);
00206     void ConfigFLL(const cfg_t &cfg);
00207
00208   public:
00214     WM8904(I2C &module);
00220     void Init(const cfg_t &cfg, const SSC_cfg_t &ssc_cfg);
00224     void Shutdown();
00230     void WriteReg(Reg::RegAddr_t reg, uint16_t dat);
00236     uint16_t ReadReg(Reg::RegAddr_t reg);
00237
00243     void SendCmd(Reg::cmd_t cmd);
00251     void SendCmdList(Reg::cmd_t *cmds, uint32_t len);
```

```
00260     void UpdateCmd(Reg::cmd_t cmd, uint16_t updateMask);
00261
00268     void SetVolume(AudioOutSelect_t out, AudioChSelect_t channel, uint8_t volume);
00275     uint8_t GetVolume(AudioOutSelect_t out, AudioChSelect_t channel);
00282     void Mute(AudioOutSelect_t out, AudioChSelect_t channel, bool mute);
00283
00289     void SetMicGain(AudioChSelect_t channel, uint8_t gain);
00295     uint8_t GetMicGain(AudioChSelect_t channel);
00301     void MuteMic(AudioChSelect_t channel, bool mute);
00302
00314     int TransmitBuffer(void *buffer, uint32_t bufferLen, bool waitIfNeeded);
00326     int ReadyReceiveBuffer(void *buffer, uint32_t bufferLen, bool waitIfNeeded);
00327
00331     void RegisterTxBufferDoneCB(SSC_BufferDoneFn_t cb);
00335     void RegisterRxBufferDoneCB(SSC_BufferDoneFn_t cb);
00336 };
00337 #endif /* ----- #ifndef __WM8904_H  ----- */
```

24.200 wm8904_reg.h

```
00001 #ifndef __WM8904_REG_H
00002 #define __WM8904_REG_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 #define WM8904_I2C_ID 0x1A
00008
00009 #define BIAS_CTL0_ISEL_HI (0x2 « 2)
00010 #define BIAS_CTL0_ISEL_LO (0x0 « 2)
00011 #define BIAS_CTL0_BIAS_ENA (1 « 0)
00012
00013 #define VMID_CTL0_BUF_ENA (1 « 6)
00014 #define VMID_CTL0_RES_DIS (0 « 1)
00015 #define VMID_CTL0_RES_MID (1 « 1)
00016 #define VMID_CTL0_RES_SLOW (2 « 1)
00017 #define VMID_CTL0_RES_FAST (3 « 1)
00018 #define VMID_CTL0_ENA (1 « 0)
00019
00020 #define PWR_MGMNT0_INL_ENA (1 « 1)
00021 #define PWR_MGMNT0_INR_ENA (1 « 0)
00022
00023 #define PWR_MGMNT2_HPL_PGA_ENA (1 « 1)
00024 #define PWR_MGMNT2_HPR_PGA_ENA (1 « 0)
00025
00026 #define PWR_MGMNT6_DACL_ENA (1 « 3)
00027 #define PWR_MGMNT6_DACR_ENA (1 « 2)
00028 #define PWR_MGMNT6_ADCL_ENA (1 « 1)
00029 #define PWR_MGMNT6_ADCR_ENA (1 « 0)
00030
00031 #define CHARGE_PUMP0_CP_ENA (1 « 0)
00032
00033 #define CLK_SYS_RATE_64 (0 « 10)
00034 #define CLK_SYS_RATE_128 (1 « 10)
00035 #define CLK_SYS_RATE_192 (2 « 10)
00036 #define CLK_SYS_RATE_256 (3 « 10)
00037 #define CLK_SYS_RATE_384 (4 « 10)
00038 #define CLK_SYS_RATE_512 (5 « 10)
00039 #define CLK_SYS_RATE_768 (6 « 10)
00040 #define CLK_SYS_RATE_1024 (7 « 10)
00041 #define CLK_SYS_RATE_1408 (8 « 10)
00042 #define CLK_SYS_RATE_1536 (9 « 10)
00043 #define CLK_SAMPLE_RATE_8K (0 « 10)
00044 #define CLK_SAMPLE_RATE_12K (1 « 10)
00045 #define CLK_SAMPLE_RATE_16K (2 « 10)
00046 #define CLK_SAMPLE_RATE_24K (3 « 10)
00047 #define CLK_SAMPLE_RATE_32K (4 « 10)
00048 #define CLK_SAMPLE_RATE_48K (5 « 10)
00049
00050 #define CLK_MCLK_INV (1 « 15)
00051 #define CLK_SYSCLK_SRC_MCLK (0 « 14)
00052 #define CLK_SYSCLK_SRC_FLL (1 « 14)
00053 #define CLK_TOCLK_RATE_DIV2 (0 « 12)
00054 #define CLK_TOCLK_RATE_DIV1 (1 « 12)
00055 #define CLK_OPCLK_ENA (1 « 3)
00056 #define CLK_CLK_SYS_ENA (1 « 2)
00057 #define CLK_CLK_DSP_ENA (1 « 1)
00058 #define CLK_TOCLK_ENA (1 « 0)
00059
00060 #define AUDIO_INTF0_DACL_DATINV (1 « 12)
00061 #define AUDIO_INTF0_DACR_DATINV (1 « 11)
00062 #define AUDIO_INTF0_DAC_BOOST(x) ((x & 3) « 9)
00063 #define AUDIO_INTF0_LOOPBACK (1 « 8)
00064 #define AUDIO_INTF0_ADC_SRCL (1 « 7)
00065 #define AUDIO_INTF0_ADC_SRCR (1 « 6)
```

```
00066 #define AUDIO_INTF0_DAC_SRCL (1 « 5)
00067 #define AUDIO_INTF0_DAC_SRCR (1 « 4)
00068 #define AUDIO_INTF0_ADC_COMP (1 « 3)
00069 #define AUDIO_INTF0_ADC_COMPMODE (1 « 2)
00070 #define AUDIO_INTF0_DAC_COMP (1 « 1)
00071 #define AUDIO_INTF0_DAC_COMPMODE (1 « 0)
00072
00073 #define AUDIO_INTF2_OPCLK_DIV_1 (0 « 8)
00074 #define AUDIO_INTF2_OPCLK_DIV_2 (1 « 8)
00075 #define AUDIO_INTF2_OPCLK_DIV_3 (2 « 8)
00076 #define AUDIO_INTF2_OPCLK_DIV_4 (3 « 8)
00077 #define AUDIO_INTF2_OPCLK_DIV_5_5 (4 « 8)
00078 #define AUDIO_INTF2_OPCLK_DIV_6 (5 « 8)
00079 #define AUDIO_INTF2_OPCLK_DIV_8 (6 « 8)
00080 #define AUDIO_INTF2_OPCLK_DIV_12 (7 « 8)
00081 #define AUDIO_INTF2_OPCLK_DIV_16 (8 « 8)
00082 #define AUDIO_INTF2_BCLK_DIV_1 (0 « 0)
00083 #define AUDIO_INTF2_BCLK_DIV_1_5 (1 « 0)
00084 #define AUDIO_INTF2_BCLK_DIV_2 (2 « 0)
00085 #define AUDIO_INTF2_BCLK_DIV_3 (3 « 0)
00086 #define AUDIO_INTF2_BCLK_DIV_4 (4 « 0)
00087 #define AUDIO_INTF2_BCLK_DIV_5 (5 « 0)
00088 #define AUDIO_INTF2_BCLK_DIV_5_5 (6 « 0)
00089 #define AUDIO_INTF2_BCLK_DIV_6 (7 « 0)
00090 #define AUDIO_INTF2_BCLK_DIV_8 (8 « 0)
00091 #define AUDIO_INTF2_BCLK_DIV_10 (9 « 0)
00092 #define AUDIO_INTF2_BCLK_DIV_11 (10 « 0)
00093 #define AUDIO_INTF2_BCLK_DIV_12 (11 « 0)
00094 #define AUDIO_INTF2_BCLK_DIV_16 (12 « 0)
00095 #define AUDIO_INTF2_BCLK_DIV_20 (13 « 0)
00096 #define AUDIO_INTF2_BCLK_DIV_22 (14 « 0)
00097 #define AUDIO_INTF2_BCLK_DIV_24 (15 « 0)
00098 #define AUDIO_INTF2_BCLK_DIV_25 (16 « 0)
00099 #define AUDIO_INTF2_BCLK_DIV_30 (17 « 0)
00100 #define AUDIO_INTF2_BCLK_DIV_32 (18 « 0)
00101 #define AUDIO_INTF2_BCLK_DIV_44 (19 « 0)
00102 #define AUDIO_INTF2_BCLK_DIV_48 (20 « 0)
00103
00104 #define AUDIO_INTF1_AIFDAC_TDM_NORM (0 « 13)
00105 #define AUDIO_INTF1_AIFDAC_TDM_TDM (1 « 13)
00106 #define AUDIO_INTF1_AIFDAC_TDM_CHAN0 (0 « 12)
00107 #define AUDIO_INTF1_AIFDAC_TDM_CHAN1 (1 « 12)
00108 #define AUDIO_INTF1_AIFADC_TDM_NORM (0 « 11)
00109 #define AUDIO_INTF1_AIFADC_TDM_TDM (1 « 11)
00110 #define AUDIO_INTF1_AIFADC_TDM_CHAN0 (0 « 10)
00111 #define AUDIO_INTF1_AIFADC_TDM_CHAN1 (1 « 10)
00112 #define AUDIO_INTF1_AIF_TRIS (1 « 8)
00113 #define AUDIO_INTF1_BCLK_INV (1 « 7)
00114 #define AUDIO_INTF1_BCLK_DIR_IN (0 « 6)
00115 #define AUDIO_INTF1_BCLK_DIR_OUT (1 « 6)
00116 #define AUDIO_INTF1_LRCLK_INV (1 « 6)
00117 #define AUDIO_INTF1_WL_16BIT (0 « 2)
00118 #define AUDIO_INTF1_WL_20BIT (1 « 2)
00119 #define AUDIO_INTF1_WL_24BIT (2 « 2)
00120 #define AUDIO_INTF1_WL_32BIT (3 « 2)
00121 #define AUDIO_INTF1_FMT_RIGHT (0 « 0)
00122 #define AUDIO_INTF1_FMT_LEFT (1 « 0)
00123 #define AUDIO_INTF1_FMT_I2S (2 « 0)
00124 #define AUDIO_INTF1_FMT_DSP (3 « 0)
00125
00126 #define AUDIO_INTF3_LRCLK_DIR_IN (0 « 11)
00127 #define AUDIO_INTF3_LRCLK_DIR_OUT (1 « 11)
00128 #define AUDIO_INTF3_LRCLK_RATE(x) ((x)&0x7FF)
00129
00130 #define DC_SERVO0_ENA_LNR (1 « 3)
00131 #define DC_SERVO0_ENA_LNL (1 « 2)
00132 #define DC_SERVO0_ENA_HPR (1 « 1)
00133 #define DC_SERVO0_ENA_HPL (1 « 0)
00134
00135 #define DC_SERVO1_TRIG_1_LNR (1 « 15)
00136 #define DC_SERVO1_TRIG_1_LNL (1 « 14)
00137 #define DC_SERVO1_TRIG_1_HPR (1 « 13)
00138 #define DC_SERVO1_TRIG_1_HPL (1 « 12)
00139 #define DC_SERVO1_TRIG_n_LNR (1 « 11)
00140 #define DC_SERVO1_TRIG_n_LNL (1 « 10)
00141 #define DC_SERVO1_TRIG_n_HPR (1 « 9)
00142 #define DC_SERVO1_TRIG_n_HPL (1 « 8)
00143 #define DC_SERVO1_TRIG_START_LNR (1 « 7)
00144 #define DC_SERVO1_TRIG_START_LNL (1 « 6)
00145 #define DC_SERVO1_TRIG_START_HPR (1 « 5)
00146 #define DC_SERVO1_TRIG_START_HPL (1 « 4)
00147 #define DC_SERVO1_TRIG_DAC_WR_LNR (1 « 3)
00148 #define DC_SERVO1_TRIG_DAC_WR_LNL (1 « 2)
00149 #define DC_SERVO1_TRIG_DAC_WR_HPR (1 « 1)
00150 #define DC_SERVO1_TRIG_DAC_WR_HPL (1 « 0)
00151
00152 #define FLL_FRACN_ENA_INT (0 « 2)
```

```
00153 #define FLL_FRACN_ENA_FRAC (1 « 2)
00154 #define FLL_OSC_ENA_DIS (0 « 1)
00155 #define FLL_OSC_ENA_EN (1 « 1)
00156 #define FLL_ENA_DIS (0 « 0)
00157 #define FLL_ENA_EN (1 « 0)
00158
00159 #define FLL_OUTDIV(x) ((uint16_t)(((x - 1) & 0x3F) « 8))
00160 #define FLL_CTRL_RATE_FVCO_1 (0 « 4)
00161 #define FLL_CTRL_RATE_FVCO_2 (1 « 4)
00162 #define FLL_CTRL_RATE_FVCO_3 (2 « 4)
00163 #define FLL_CTRL_RATE_FVCO_4 (3 « 4)
00164 #define FLL_CTRL_RATE_FVCO_5 (4 « 4)
00165 #define FLL_CTRL_RATE_FVCO_6 (5 « 4)
00166 #define FLL_CTRL_RATE_FVCO_7 (6 « 4)
00167 #define FLL_CTRL_RATE_FVCO_8 (7 « 4)
00168 #define FLL_FRATIO_DIV_1 (0 « 0)
00169 #define FLL_FRATIO_DIV_2 (1 « 0)
00170 #define FLL_FRATIO_DIV_4 (2 « 0)
00171 #define FLL_FRATIO_DIV_8 (3 « 0)
00172 #define FLL_FRATIO_DIV_16 (7 « 0)
00173
00174 #define FLL_K(x) ((uint16_t)(((x)&0xFFFF) « 0))
00175
00176 #define FLL_N(x) ((uint16_t)(((x)&0x3FF) « 5))
00177 #define FLL_GAIN_1 (0 « 0)
00178 #define FLL_GAIN_2 (1 « 0)
00179 #define FLL_GAIN_4 (2 « 0)
00180 #define FLL_GAIN_8 (3 « 0)
00181 #define FLL_GAIN_16 (4 « 0)
00182 #define FLL_GAIN_32 (5 « 0)
00183 #define FLL_GAIN_64 (6 « 0)
00184 #define FLL_GAIN_128 (7 « 0)
00185 #define FLL_GAIN_256 (8 « 0)
00186
00187 #define FLL_CLK_REF_DIV_1 (0 « 3)
00188 #define FLL_CLK_REF_DIV_2 (1 « 3)
00189 #define FLL_CLK_REF_DIV_4 (2 « 3)
00190 #define FLL_CLK_REF_DIV_8 (3 « 3)
00191 #define FLL_CLK_REF_DIV_16 (4 « 3)
00192 #define FLL_CLK_REF_SRC_MCLK (0 « 0)
00193 #define FLL_CLK_REF_SRC_BCLK (1 « 0)
00194 #define FLL_CLK_REF_SRC_LRCLK (2 « 0)
00195
00196 #define GPIO_CTL_GPIO_PU (1 « 5)
00197 #define GPIO_CTL_GPIO_PD (1 « 4)
00198 #define GPIO_CTL_GPIO_IN (0 « 0)
00199 #define GPIO_CTL_GPIO_CLK_OUT (1 « 0)
00200 #define GPIO_CTL_GPIO_LO (2 « 0)
00201 #define GPIO_CTL_GPIO_HI (3 « 0)
00202 #define GPIO_CTL_GPIO_IRQ (4 « 0)
00203 #define GPIO_CTL_GPIO_FLL_LOCK (5 « 0)
00204 #define GPIO_CTL_GPIO_MIC_DET (6 « 0)
00205 #define GPIO_CTL_GPIO_MIC_SHORT (7 « 0)
00206 #define GPIO_CTL_GPIO_DMIC_CLKO (8 « 0)
00207 #define GPIO_CTL_GPIO_FLL_CLKO (9 « 0)
00208
00209 #define ANA_IN_MUTE (1 « 7)
00210 #define ANA_IN_VOLUME(x) ((uint16_t)(((x)&0x1F) « 0))
00211
00212 #define ANA_HP0_HPL_RM_SHORT (1 « 7)
00213 #define ANA_HP0_HPL_ENA_OUTP (1 « 6)
00214 #define ANA_HP0_HPL_ENA_DLY (1 « 5)
00215 #define ANA_HP0_HPL_ENA (1 « 4)
00216 #define ANA_HP0_HPR_RM_SHORT (1 « 3)
00217 #define ANA_HP0_HPR_ENA_OUTP (1 « 2)
00218 #define ANA_HP0_HPR_ENA_DLY (1 « 1)
00219 #define ANA_HP0_HPR_ENA (1 « 0)
00220
00221 #define ANA_OUT_MUTE (1 « 8)
00222 #define ANA_OUT_VOL_UPDATE (1 « 7)
00223 #define ANA_OUT_ZERO_CROSS (1 « 6)
00224 #define ANA_OUT_VOLUME(x) ((uint16_t)((x & 0x3F) « 0))
00225
00226 #define DGTL_PULLS_MCLK_PU (1 « 7)
00227 #define DGTL_PULLS_MCLK_PD (1 « 6)
00228 #define DGTL_PULLS_DACDAT_PU (1 « 5)
00229 #define DGTL_PULLS_DACDAT_PD (1 « 4)
00230 #define DGTL_PULLS_LRCLK_PU (1 « 3)
00231 #define DGTL_PULLS_LRCLK_PD (1 « 2)
00232 #define DGTL_PULLS_BCLK_PU (1 « 1)
00233 #define DGTL_PULLS_BCLK_PD (1 « 0)
00234
00235 #define FLL_NCO_FRC_NCO_EN (1 « 0)
00236 #define FLL_NCO_FRC_NCO_DIS (0 « 0)
00237
00238 #endif /* ----- #ifndef __WM8904_REG_H  ----- */
```

24.201 PlatformSpecific/SOMRT1061/EFFS_MultiPart/src/fsl_ocotp.h

```
00001 /*
00002  * Copyright 2019 NXP
00003  * All rights reserved.
00004  *
00005  * SPDX-License-Identifier: BSD-3-Clause
00006  */
00007 #ifndef _FSL_OCOTP_H_
00008 #define _FSL_OCOTP_H_
00009
00010 #include <stdint.h>
00011 #include <stddef.h>
00012 #include <stdbool.h>
00013 #include <sim.h>
00014
00015 /*
00016  * Definitions
00017  */
00018
00019 /* OCOTP driver version 2.0.0 */
00020 #define FSL_OCOTP_DRIVER_VERSION (MAKE_VERSION(2, 0, 0))
00021 #define MAKE_STATUS(x, y) y
00022 #define assert(x)
00023
00024 /* Error codes for the OCOTP driver. */
00025 typedef enum _ocotp_status
00026 {
00027     kStatus_Success           = 0,
00028     kStatus_OCOTP_AccessError = 0x80000000,
00029     kStatus_OCOTP_CrcFail     = 0x80000001,
00030 } status_t;
00031
00032 /*  OCOTP timing structure.
00033  *  Note that, these value are used for calcalating the read/write timings.
00034  *  And the values should satisfy below rules:
00035  *
00036  *  Tsp_rd=(WAIT+1)/ipg_clk_freq should be >= 150ns;
00037  *  Tsp_pgm=(RELAX+1)/ipg_clk_freq should be >= 100ns;
00038  *  Trd = ((STROBE_READ+1)- 2*(RELAX_READ+1)) /ipg_clk_freq,
00039  *  The Trd is required to be larger than 40 ns.
00040  *  Tpgm = ((STROBE_PROG+1)- 2*(RELAX_PROG+1)) /ipg_clk_freq;
00041  *  The Tpgm should be configured within the range of 9000 ns < Tpgm < 11000 ns;
00042  */
00043 typedef struct _ocotp_timing
00044 {
00045     uint32_t wait;
00046     uint32_t relax;
00047     uint32_t strobe_prog;
00048     uint32_t strobe_read;
00049 } ocotp_timing_t;
00050
00051 /*
00052  * API
00053  */
00054
00055 #if defined(__cplusplus)
00056 extern "C" {
00057 #endif
00058
00059 /*
00060  * Initializes OCOTP controller.
00061  *
00062  * param base         OCOTP peripheral base address.
00063  * param srcClock_Hz  source clock frequency in unit of Hz.
00064  */
00065 void OCOTP_Init(OCOTP_Type *base, uint32_t srcClock_Hz);
00066
00067 /*
00068  * De-initializes OCOTP controller.
00069  *
00070  * retval kStatus_Success upon successful execution, error status otherwise.
00071  */
00072 void OCOTP_Deinit(OCOTP_Type *base);
00073
00074 /*
00075  * Checking the BUSY bit in CTRL register.
00076  * Checking this BUSY bit will help confirm if the OCOTP controller is ready for access.
00077  *
00078  * param base         OCOTP peripheral base address.
00079  * retval       true for bit set and false for cleared.
00080  */
00081 static inline bool OCOTP_CheckBusyStatus(OCOTP_Type *base)
00082 {
00083     return ((OCOTP_CTRL_BUSY_MASK == (base->CTRL & OCOTP_CTRL_BUSY_MASK)) ? (true) : (false));
00084 }
00085
```

```
00086 /*
00087  * brief Checking the ERROR bit in CTRL register.
00088  *
00089  * param base         OCOTP peripheral base address.
00090  * retval       true for bit set and false for cleared.
00091  */
00092 static inline bool OCOTP_CheckErrorStatus(OCOTP_Type *base)
00093 {
00094     return ((OCOTP_CTRL_ERROR_MASK == (base->CTRL & OCOTP_CTRL_ERROR_MASK)) ? (true) : (false));
00095 }
00096
00097 /*
00098  * brief Clear the error bit if this bit is set.
00099  *
00100  * param base  OCOTP peripheral base address.
00101  */
00102 static inline void OCOTP_ClearErrorStatus(OCOTP_Type *base)
00103 {
00104     base->CTRL_CLR = OCOTP_CTRL_CLR_ERROR_MASK;
00105 }
00106
00107 /*
00108  * brief Reload the shadow register.
00109  * This function will help reload the shadow register without reseting the OCOTP module.
00110  * Please make sure the OCOTP has been initialized before calling this API.
00111  *
00112  * param base OCOTP peripheral base addess.
00113  */
00114 void OCOTP_ReloadShadowRegister(OCOTP_Type *base);
00115
00116 /*
00117  * brief Read the fuse shadow register with the fuse addess.
00118  *
00119  * param base     OCOTP peripheral base address.
00120  * param address  the fuse address to be read from.
00121  */
00122 uint32_t OCOTP_ReadFuseShadowRegister(OCOTP_Type *base, uint32_t address);
00123
00124 /*
00125  * brief Write the fuse shadow register with the fuse addess and data.
00126  * Please make sure the wrtie address is not locked while calling this API.
00127  *
00128  * param base     OCOTP peripheral base address.
00129  * param address  the fuse address to be written.
00130  * param data     the value will be writen to fuse address.
00131  * retval   write status, kStatus_Success for success and kStatus_Fail for failed.
00132  */
00133 status_t OCOTP_WriteFuseShadowRegister(OCOTP_Type *base, uint32_t address, uint32_t data);
00134
00135 /*
00136  * brief Get the OCOTP controller version from the register.
00137  *
00138  * param base    OCOTP peripheral base address.
00139  * retval  return the version value.
00140  */
00141 static inline uint32_t OCOTP_GetVersion(OCOTP_Type *base)
00142 {
00143     return (base->VERSION);
00144 }
00145
00146 #if defined(__cplusplus)
00147 }
00148 #endif
00149
00150 #endif /* _FSL_OCOTP_H_ */
```

24.202 Web/SimpleHtml/src/fsl_ocotp.h

```
00001 /*
00002  * Copyright 2019 NXP
00003  * All rights reserved.
00004  *
00005  * SPDX-License-Identifier: BSD-3-Clause
00006  */
00007 #ifndef _FSL_OCOTP_H_
00008 #define _FSL_OCOTP_H_
00009
00010 #include <stdint.h>
00011 #include <stddef.h>
00012 #include <stdbool.h>
00013 #include <sim.h>
00014
00020 /*******************************************************************************
00021  * Definitions
00022  ******************************************************************************/
```

```
00026 #define FSL_OCOTP_DRIVER_VERSION (MAKE_VERSION(2, 0, 0))
00029 #define MAKE_STATUS(x, y) y
00030 #define assert(x)
00031
00033 typedef enum _ocotp_status
00034 {
00035     kStatus_Success           = 0,
00036     kStatus_OCOTP_AccessError = 0x80000000,
00037     kStatus_OCOTP_CrcFail     = 0x80000001,
00038 } status_t;
00039
00051 typedef struct _ocotp_timing
00052 {
00053     uint32_t wait;
00054     uint32_t relax;
00055     uint32_t strobe_prog;
00056     uint32_t strobe_read;
00057 } ocotp_timing_t;
00058
00059 /*******************************************************************************
00060  * API
00061  ******************************************************************************/
00062
00063 #if defined(__cplusplus)
00064 extern "C" {
00065 #endif
00066
00073 void OCOTP_Init(OCOTP_Type *base, uint32_t srcClock_Hz);
00074
00080 void OCOTP_Deinit(OCOTP_Type *base);
00081
00089 static inline bool OCOTP_CheckBusyStatus(OCOTP_Type *base)
00090 {
00091     return ((OCOTP_CTRL_BUSY_MASK == (base->CTRL & OCOTP_CTRL_BUSY_MASK)) ? (true) : (false));
00092 }
00093
00100 static inline bool OCOTP_CheckErrorStatus(OCOTP_Type *base)
00101 {
00102     return ((OCOTP_CTRL_ERROR_MASK == (base->CTRL & OCOTP_CTRL_ERROR_MASK)) ? (true) : (false));
00103 }
00104
00110 static inline void OCOTP_ClearErrorStatus(OCOTP_Type *base)
00111 {
00112     base->CTRL_CLR = OCOTP_CTRL_CLR_ERROR_MASK;
00113 }
00114
00122 void OCOTP_ReloadShadowRegister(OCOTP_Type *base);
00123
00130 uint32_t OCOTP_ReadFuseShadowRegister(OCOTP_Type *base, uint32_t address);
00131
00141 status_t OCOTP_WriteFuseShadowRegister(OCOTP_Type *base, uint32_t address, uint32_t data);
00142
00149 static inline uint32_t OCOTP_GetVersion(OCOTP_Type *base)
00150 {
00151     return (base->VERSION);
00152 }
00153
00154 #if defined(__cplusplus)
00155 }
00156 #endif
00157
00160 #endif /* _FSL_OCOTP_H_ */
```

24.203 datapump.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DATAPUMP_H_
00006 #define _DATAPUMP_H_
00007
00008 /*
00009  * Since this program may be used for machine to machine connections you
00010  * can configure the messages that are sent.
00011  *
00012  * MTS messages are Message To Serial
00013  * MTN messages are Messages to the Network
00014  */
00015 #define MTS_WHEN_NOTCONNECTED "Waiting for connection \r\n"
00016 #define MTS_WHEN_CONNECTION_OPENED "New Connection Opened\r\n"
00017 #define MTS_WHEN_CONNECTION_CLOSED "Connection Closed by Network \r\n"
00018 #define MTS_WHEN_CONNECTION_TIMEDOUT "Connection Timed out and Closed\r\n"
00019 #define MTS_WHEN_CONNECTION_OVERIDDEN "This Connection is being Overidden.\r\n"
00020
```

```
00021 #define MTN_WHEN_CONNECTION_OVERIDDEN "Your Connection was just Overidden\r\n"
00022 #define MTN_WHEN_CONNECTION_OPENED "Connection Opened \r\n"
00023 #define MTN_WHEN_CONNECTION_TIMEDOUT "Your Connection Timed out and will be Closed\r\n"
00024
00025 #define BUFFER_SIZE (1500)
00026 #define CLIENT_WRITE_BUF_SIZE (256)
00027
00028 int DataPump(int fd1, int fd2, int serverfd);   // file descriptor data pump
00029
00030 #endif
```

24.204 fdtimer.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _FDTIMER_H_
00006 #define _FDTIMER_H_
00007
00008 /*
00009  ******************************************************************************
00010  *
00011  * Global data definitions
00012  *
00013  ******************************************************************************
00014  */
00015 extern volatile int fd_notify[NB_FACTORY_SERIAL_PORTS];
00016 extern volatile int fd_tick_left[NB_FACTORY_SERIAL_PORTS];
00017 extern volatile int FD_shutdown;
00018
00019 /*
00020  ******************************************************************************
00021  *
00022  * Functions
00023  *
00024  ******************************************************************************
00025  */
00026 void SetUpFdtimer(void);
00027 void SetTicks(int n, int ticks);
00028
00029 #endif /* _FDTIMER_H_ */
```

24.205 i2cfuncs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _I2CFUNCS_H_
00006 #define _I2CFUNCS_H_
00007
00008 #ifdef SUPPORTED_I2C_PLATFORM
00009
00010 #if (defined NANO54415 || defined MOD5441X || defined SB800EX)
00011 #include <multichanneli2c.h>
00012 #elif (defined SAME70)
00013 #include <i2c.h>
00014 #else
00015 #warning I2C features not supported for this platform
00016 #endif
00017
00018 #ifdef __cplusplus
00019 extern "C"
00020 {
00021 #endif
00022
00023 #define I2C_MODULE_NUM 0         // use I2C0/TW0
00024 #define I2C_BUS_SPEED (100000)   // bus speed of 100kHz
00025
00026 #define MAX_I2C_BUS_SPEED (400000)  // SAME70 specific value. Max bus speed for the TWI is 400 kHz in
      Fast Mode
00027 #define I2C_LOW_ADDR_LIMIT (0x07)
00028 #define I2C_HIGH_ADDR_LIMIT (0x78)
00029
00030 #define I2C_MAX_BUF_SIZE (64)          // Size allocated to input and output buffers in slave mode I2C
00031 #define I2C_SLAVE_TX_TERM_CHAR (0)     // Terminating char to be sent when Slave TX buffer is empty
00032 #define NB_FACTORY_I2C_ERR_VALUE (0)   // Error value if there is a failure to extract webpage form
      data
00033
00034 /* defined I2C Timeout values will be used if user does not include the 'ticks_to_wait'
00035    parameter when calling I2C functions                                          */
```

```
00036 #define I2C_RX_TX_TIMEOUT (5)     // Ticks allowed before timeout of a single byte transmission
00037 #define I2C_START_TIMEOUT (20)    // Ticks allowed before timeout when attempting start on I2C bus
00038
00039 #define I2C_MAX_WRITE_LEN (255)
00040 #define I2C_MAX_READ_LEN (255)
00041 // *2 to account for inputting values in a 2-digit hex format
00042 #define MAX_BUFFER_LEN ((I2C_MAX_WRITE_LEN > I2C_MAX_READ_LEN) ? I2C_MAX_WRITE_LEN * 2 :
      I2C_MAX_READ_LEN * 2)
00043 #define MAX_I2C_DEVICES (128)
00044
00045 // These are the values the NetBurner I2C functions will return
00046 #define I2C_OK (0)                // Last instruction terminated correctly
00047 #define I2C_NEXT_WRITE_OK (1)     // I2C bus is OK for a write
00048 #define I2C_NEXT_READ_OK (2)      // I2C bus is OK for a read
00049 #define I2C_MASTER_OK (3)         // I2C finished transmission but still owns but (need to stop or
      restart)
00050 #define I2C_TIMEOUT (4)           // A timeout occurred while trying communicate on I2C bus
00051 #define I2C_BUS_NOT_AVAIL (5)     // A timeout occurred while trying gain I2C bus control
00052 #define I2C_NOT_READY (6)         // A read or write was attempted before I2C ready or during a slave
      transmission
00053 #define I2C_LOST_ARB (7)          // Lost arbitration during start
00054 #define I2C_LOST_ARB_ADD (8)      // Lost arbitration and then winner addressed our slave address
00055 #define I2C_NO_LINK_RX_ACK (9)    // We are in Master TX mode and received no ACK from slave device,
      possibly during start
00056
00057
00058     struct ModuleI2CAddress_Struct
00059     {
00060         unsigned char byteAddress;
00061         unsigned char asciiAddress[3];
00062     };
00063
00064     // This structure stores the I2C address retrieved from Non-volatile memory
00065     // typedef struct NVSettings_Struct
00066     //{
00067     //    char slaveI2CAddress[3];
00068     //};
00069
00070     // Stores ascii and byte versions of I2C parameters
00071     struct I2CParam_Struct
00072     {
00073         char asciiAddress[3];
00074         char asciiData[MAX_BUFFER_LEN];
00075         char asciiLength[3];
00076         unsigned char byteAddress;
00077         unsigned char byteData;
00078         unsigned char byteBuffer[MAX_BUFFER_LEN];
00079         int length;
00080     };
00081
00082     // Used for handling the web I2C functions
00083     struct I2CWebFuncs_Struct
00084     {
00085         I2CParam_Struct WriteReadByteParams;
00086         I2CParam_Struct WriteReadBufferParams;
00087         I2CParam_Struct WriteByteParams;
00088         I2CParam_Struct ReadByteParams;
00089         I2CParam_Struct WriteBufferParams;
00090         I2CParam_Struct ReadBufferParams;
00091     };
00092
00093     // Used for handling the serial I2C functions
00094     struct SerialParams_Struct
00095     {
00096         // int strLen;
00097         unsigned char buf[3];
00098         unsigned char byteData;
00099         unsigned char length;
00100         unsigned char addr;
00101         unsigned char byteToWrite;
00102         unsigned char commBuffer[MAX_BUFFER_LEN];
00103     };
00104
00105     struct I2CBusStatus_Struct
00106     {
00107         volatile uint8_t status;
00108         const char *statusMsgs[7];
00109     };
00110
00111 #if (defined SAME70)
00112 void InitI2C(uint32_t busSpeed = 0);
00113 #else
00114 #error Multi_I2CInit declaration missing for defined platform
00115 #endif
00116
00117     unsigned char Ascii2Byte(char *buf);
00118     int I2CScan(bool *discovered, unsigned char slaveAddress);
```

```
00119     void I2CStatus();
00120
00121 #ifdef __cplusplus
00122 }
00123
00124 #endif   // __cplusplus
00125 #endif   // SUPPORTED_I2C_PLATFORM
00126 #endif   // _I2CFUNCS_H_
```

24.206 i2crecord.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _I2C_RECORD_H_
00006 #define _I2C_RECORD_H_
00007 #pragma once
00008
00009 #include <basictypes.h>
00010
00011 /*
00012 ********************************************************************************
00013 *
00014 * Class Definition (struct is class with all members public)
00015 *
00016 ********************************************************************************
00017 */
00018 struct I2CRecord
00019 {
00020     /*
00021      * Data
00022      */
00023     // File descriptor of listening socket
00024     int FD_ListeningSocket;
00025
00026     // File descriptor of connected socket
00027     int FD_ConnectedSocket;
00028
00029     // TRUE if we established the connection
00030     bool bWeInitiatedConnection;
00031
00032     // Time of last network sourced data in seconds since last boot
00033     uint32_t LastNetWorkDataRxed;
00034
00035     // Time of last transmitted data in seconds since last boot
00036     uint32_t LastNetWorkDataTxed;
00037
00038     /*
00039      * Methods
00040      */
00041     void ProcessI2CTimeouts(void);
00042 };
00043
00044 #endif
```

24.207 i2cserver.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _I2CSERVER_H_
00006 #define _I2CSERVER_H_
00007
00008 #define TCP_LISTEN_PORT 23   // Telnet port number
00009 #define FDNET_RX_BUFSIZE 1024
00010 #define MAX_CMD_LEN 1024   // Maximum command buffer length
00011 #define BEGIN_CMD_CHAR '#'
00012 #define END_CMD_CHAR '\n'
00013 #define EOT_CHAR 4
00014 #define MENU_CHAR '?'
00015
00016 #define BEGIN_CMD_CHARS 1   // represented by '#'
00017 #define END_CMD_CHARS 1     // represented by '\n'
00018 #define CMD_CHARS 2         // represented by the two cmd chars, ex: WA, WB, etc
00019 #define NON_PARAM_CHARS BEGIN_CMD_CHARS + END_CMD_CHARS + CMD_CHARS
00020 #define FDNET_READ_CHARS 1
00021 #define MAX_FDNET_READ_CHARS 8
00022
00023 // Character count for each command. Used for verifying the correct input
00024 #define WR_PARAM_CHARS 4   // 2 address chars + 2 data chars
```

```
00025 #define WA_PARAM_CHARS 6   // 2 address chars + 2 data chars + 2 length chars
00026 #define WW_PARAM_CHARS 4   // 2 address chars + 2 length chars, need to add data seperately
00027 #define WB_PARAM_CHARS 4   // 2 address chars + 2 data chars
00028 #define RB_PARAM_CHARS 2   // 2 address chars
00029 #define RR_PARAM_CHARS 4   // 2 address chars + 2 length chars
00030 #define SE_PARAM_CHARS 2   // 2 new address chars
00031 #define SV_PARAM_CHARS 2   // 2 new address chars
00032 #define ST_PARAM_CHARS 0   // no parameters required
00033 #define RE_PARAM_CHARS 0   // no parameters required
00034 #define SC_PARAM_CHARS 0   // no parameters required
00035 #define TM_PARAM_CHARS 0   // no parameters required
00036
00037 extern const char *statusMsgs[];
00038
00039 #ifdef __cplusplus
00040 extern "C"
00041 {
00042 #endif
00043
00044     void checkCmdBuf(char *cmdBuf);
00045     bool checkCmdLength(char *cmdBuf, int cmdParamChars);
00046     bool checkEndCmdChar(char *cmdBuf, int nTotalBytesRead, int nBytesRead);
00047     bool checkEOTChar(int nTotalBytesRead, int &nBytesRead);
00048     bool checkMenuChar(char *cmdBuf, int nTotalBytesRead, volatile int &nBytesRead);
00049     void checkRXBuffer(int nTotalBytesRead, int nBytesRead);
00050     bool checkStartCmdChar(char *cmdBuf, int nTotalBytesRead, int nBytesRead);
00051     void IPtoString(IPADDR ia, char *s);
00052     void MainMenu(int fdnet);
00053     void I2CServerTask(void *pd);
00054     bool parseTelnetInput(char *cmdBuf, int nTotalBytesRead, int &nBytesRead);
00055     void printI2CStatusErrorMsg();
00056     int processCmd(int fdnet, char *cmdBuf);
00057     unsigned char processI2CAddress(char *cmdBuf, unsigned char *asciiBuf);
00058     unsigned char processI2CDataByte(char *cmdBuf, unsigned char *asciiBuf);
00059     bool processI2CReadLength(char *cmdBuf, unsigned char *asciiBuf, unsigned char &length);
00060     bool processI2CWriteLength(char *cmdBuf, unsigned char *asciiBuf, unsigned char &length);
00061     unsigned char processWR(int fdnet, char *cmdBuf);
00062     unsigned char processWA(int fdnet, char *cmdBuf);
00063     unsigned char processWW(int fdnet, char *cmdBuf);
00064     unsigned char processWB(int fdnet, char *cmdBuf);
00065     unsigned char processRB(int fdnet, char *cmdBuf);
00066     unsigned char processRR(int fdnet, char *cmdBuf);
00067     unsigned char processSE(int fdnet, char *cmdBuf);
00068     unsigned char processSV(int fdnet, char *cmdBuf);
00069     unsigned char processST(int fdnet, char *cmdBuf);
00070     unsigned char processRE(int fdnet, char *cmdBuf);
00071     unsigned char processSC(int fdnet, char *cmdBuf);
00072     void storeCmdChars(char *cmdBuf, int nTotalBytesRead, int nBytesRead);
00073
00074 #ifdef __cplusplus
00075 }
00076 #endif
00077
00078 #endif   // _TCPSERVER_H_
```

24.208 SSH/SecureSerToEthFactoryApp/src/nbfactory.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_FACTORY_H_
00006 #define _NB_FACTORY_H_
00007
00008 /*
00009  ******************************************************************************
00010  *
00011  * Factory defaults for this factory application
00012  *
00013  ******************************************************************************
00014  */
00015 /* Configuration verify key (increment if data changed, added, reorganized) */
00016 #define NB_FACTORY_VERIFY_KEY (0x7E545094)
00017 /* STD-EFFS verify key (increment to format file system, required for 1.6-1.9 STD EFFS Upgrade) */
00018 #define STD_EFFS_VERIFY_KEY (0x15A58104)
00019
00020 /* Version  VV.NN.vvvv ( 0xVVNNvvvv ), string must match constant  */
00021 #define NB_FACTORY_DEFAULTS_VERSION (uint32_t)(0x03010004)
00022 #define NB_FACTORY_DEFAULTS_VERSION_STRING "03.01.04"
00023
00024 /* Module Base Name */
00025 #if defined MOD5441X
00026 #define NB_FACTORY_MODULE_BASE_NAME "MOD5441x"
00027 #elif defined SB800EX
00028 #define NB_FACTORY_MODULE_BASE_NAME "SB800EX"
```

```
00029 #elif defined NANO54415
00030 #define NB_FACTORY_MODULE_BASE_NAME "NANO54415"
00031 #elif defined MODM7AE70
00032 #define NB_FACTORY_MODULE_BASE_NAME "MODM7AE70"
00033 #elif defined SBE70LC
00034 #define NB_FACTORY_MODULE_BASE_NAME "SBE70LC"
00035 #elif defined SOMRT1061
00036 #define NB_FACTORY_MODULE_BASE_NAME "SOMRT1061"
00037 #else
00038 #error Module not supported
00039 #endif
00040
00041 /*
00042  * This enables the application to work with the PK70 platform and any one of
00043  * the following quad-UART blade boards:
00044  *
00045  *    - NBPKBU-100/232CR - use PK70_QUAD_232 definition
00046  *    - NBPKBU-485CR - use PK70_QUAD_485 definition
00047  *    - NBPKBU-MMSCR - use PK70_QUAD_MMS definition
00048  *
00049  * Only one definition can be un-commented at a time.
00050  */
00051 //#define PK70_QUAD_232 (1)
00052 //#define PK70_QUAD_485  ( 1 )
00053 //#define PK70_QUAD_MMS  ( 1 )
00054
00055 /* Module Description */
00056 #define NB_FACTORY_BASE_DESC "Network to Serial"
00057
00058 /* Feature Name and Description */
00059 #define NB_FACTORY_FEATURE_NAME "SX"
00060 #define NB_FACTORY_FEATURE_DESC "Securable Network to Serial"
00061
00062 #define WIFI_INSTALLED  ( 1 )    // Applies to general wifi functionality
00063
00064 #if (defined SBE70LC)
00065 #define SUPPORTED_I2C_PLATFORM
00066 #endif
00067
00068 #ifdef SUPPORTED_I2C_PLATFORM
00069 #define NB_FACTORY_I2C_PORTS (1)
00070 #else
00071 #define NB_FACTORY_I2C_PORTS (0)
00072 #endif
00073
00074 /* The GPIO server can interfere with pins needed by other features such as I2C.
00075  * Only use the GPIO server when not using other features that require use of pins
00076  * on the J1 header such as the Ethernet-to-I2C server.  */
00077 #if ((defined SBE70LC) && !(defined SUPPORTED_I2C_PLATFORM))
00078 #define SUPPORTED_GPIO_SERVER_PLATFORM
00079 #endif
00080
00081 /* The GPIO server can take control of pins from other features such as I2C and WiFi.
00082  * Only use the GPIO server when not using other features that require use of pins
00083  * on the P1 header such as WiFi and the Ethernet-to-I2C server.  */
00084 #if ((defined MODM7AE70) && !(defined SUPPORTED_I2C_PLATFORM) && !(defined WIFI_INSTALLED))
00085 #define SUPPORTED_GPIO_SERVER_PLATFORM
00086 #endif
00087
00088 /* Use all 7 serial ports (USARTS+UARTS) on the MODM7AE70 instead of just 2 (USART only).
00089  * Using all 7 serial ports will increase the use of config memory space.
00090  */
00091 // #define USE_E70_UART_SERIAL_PORTS
00092
00093 /* Certain platforms have serial ports that utilize pins that conflict with other system
00094  * hardware, such as ethernet. Defining "DISABLE_SERIAL_PORTS_WITH_PIN_CONFLICTS"
00095  * will disable those serial ports to allow other hardware to claim those pins.
00096  */
00097 #define DISABLE_SERIAL_PORTS_WITH_PIN_CONFLICTS
00098
00099 /* Number of serial ports */
00100 // If you change this to allow other platforms to use more than 2,
00101 // a fair bit of logic will need to change
00102 #if (defined NANO54415)
00103 #define NB_FACTORY_SERIAL_PORTS (5)
00104 #elif (defined MODM7AE70 && defined USE_E70_UART_SERIAL_PORTS)
00105 #define NB_FACTORY_SERIAL_PORTS (7)
00106 #elif (defined SOMRT1061)
00107 #define NB_FACTORY_SERIAL_PORTS (7)
00108 #else
00109 #define NB_FACTORY_SERIAL_PORTS (2)
00110 #endif
00111
00112 /* SSL */
00113 #define NB_FACTORY_INCLUDE_SSL (1)
00114
00115 /* SSH */
```

```
00116 #define NB_FACTORY_INCLUDE_SSH (1)
00117
00118 /*** Network Defaults ***/
00119
00120 /* PROTOCOL_TCP_TO_SERIAL is 2 in serialburnerdata.h */
00121 #define NB_FACTORY_PROTOCOL_DEFAULT "TCP/SSL"
00122 #define NB_FACTORY_LISTEN_PORT_DEFAULT IANA_TELNET_PORT   // Port 23
00123 #define NB_FACTORY_INACTIVITY_TIMEOUT_DEFAULT (60)
00124 #define NB_FACTORY_NEW_CONNECTION_TIMEOUT_DEFAULT (30)
00125
00126 /* I2C to Ethernet default port */
00127 #define NB_FACTORY_I2C_LISTEN_PORT_DEFAULT (26)
00128
00129 /* SERIAL_CONNECT_CONNECT_NEVER is 1 serialburnerdata.h */
00130 #define NB_FACTORY_CONNECT_MODE_DEFAULT SERIAL_CONNECT_CONNECT_NEVER
00131
00132 /* Near the end of well known ports (1023) */
00133 #define NB_FACTORY_CONNECT_PORT_DEFAULT (1000)
00134
00135 /* IP address 0.0.0.0 is zero */
00136 #define NB_FACTORY_CONNECT_ADDRESS (0)
00137
00138 #define NB_CONNECT_IDLE_TIMEOUT_DEFAULT (60)
00139 #define NB_CONNECT_RETRY_TIMEOUT_DEFAULT (360)
00140 #define NB_CONENCT_KEEP_ALIVE_INTERVAL_DEFAULT (0)
00141 #define NB_MAX_TCP_CONNECTIONS (5)
00142
00143 #define NB_FACTORY_NTPSERVERNAME_DEFAULT "pool.ntp.org"
00144
00145 /*** Serial defaults ***/
00146
00147 /* SERIAL_MODE_RS232 is 1 serialburnerdata.h */
00148 #define NB_FACTORY_DEFAULT_SERIAL_MODE SERIAL_MODE_RS232
00149
00150 #if ( defined SBE70LC )
00151 #define NB_FACTORY_DEFAULT_SERIAL_PORT_MODES {SERIAL_MODE_RS232, SERIAL_MODE_RS232, "\0"}
00152 #elif ( defined MODM7AE70 )
00153 #define NB_FACTORY_DEFAULT_SERIAL_PORT_MODES {SERIAL_MODE_RS232, SERIAL_MODE_RS232,
      SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED,
      SERIAL_MODE_DISABLED, "\0"}
00154 #elif ( defined SOMRT1061 )
00155 #define NB_FACTORY_DEFAULT_SERIAL_PORT_MODES {SERIAL_MODE_RS232, SERIAL_MODE_DISABLED,
      SERIAL_MODE_RS232, SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED,
      SERIAL_MODE_DISABLED, "\0"}
00156 #elif ( defined MOD5441X )
00157 #define NB_FACTORY_DEFAULT_SERIAL_PORT_MODES {SERIAL_MODE_RS232, SERIAL_MODE_RS232, "\0"}
00158 #elif ( defined SB800EX )
00159 #define NB_FACTORY_DEFAULT_SERIAL_PORT_MODES {SERIAL_MODE_RS232, SERIAL_MODE_RS232, "\0"}
00160 #elif ( defined NANO54415 )
00161 #define NB_FACTORY_DEFAULT_SERIAL_PORT_MODES {SERIAL_MODE_RS232, SERIAL_MODE_RS232,
      SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED, SERIAL_MODE_DISABLED, "\0"}
00162 #else
00163    #error Module not supported
00164 #endif
00165
00166 /* Serial mode capability sets  */
00167 #define NB_FACTORY_SERIAL_CAPABILITY_FULL SERIAL_MODE_RS232 "," SERIAL_MODE_DEBUG "," SERIAL_MODE_485H
      "," SERIAL_MODE_485F "," SERIAL_MODE_422F "," SERIAL_MODE_DISABLED
00168 #define NB_FACTORY_SERIAL_CAPABILITY_RS232 SERIAL_MODE_RS232 "," SERIAL_MODE_DEBUG ","
      SERIAL_MODE_DISABLED
00169 #define NB_FACTORY_SERIAL_CAPABILITY_DISABLED SERIAL_MODE_DISABLED
00170
00171 /*
00172  * Serial port fully capable RS-232 & RS-485 and debug default
00173  *    Assuming default configurations or development boards
00174  */
00175 #if ( defined SBE70LC )
00176    #define NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00177    #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00178 #elif ( defined MODM7AE70 )
00179    #define NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00180    #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00181    #define NB_FACTORY_SERIAL_PORT_2_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00182    #define NB_FACTORY_SERIAL_PORT_3_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00183    #define NB_FACTORY_SERIAL_PORT_4_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00184    #define NB_FACTORY_SERIAL_PORT_5_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00185    #define NB_FACTORY_SERIAL_PORT_6_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00186 #elif ( defined SOMRT1061 )
00187    #define NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00188    #if (defined DISABLE_SERIAL_PORTS_WITH_PIN_CONFLICTS)
00189       #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_DISABLED
00190    #else
00191    #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00192    #endif  // end DISABLE_SERIAL_PORTS_WITH_PIN_CONFLICTS
00193    #define NB_FACTORY_SERIAL_PORT_2_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00194    #define NB_FACTORY_SERIAL_PORT_3_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00195    #define NB_FACTORY_SERIAL_PORT_4_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
```

```
00196    #if (defined DISABLE_SERIAL_PORTS_WITH_PIN_CONFLICTS)
00197       #define NB_FACTORY_SERIAL_PORT_5_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_DISABLED
00198    #else
00199       #define NB_FACTORY_SERIAL_PORT_5_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00200    #endif // end DISABLE_SERIAL_PORTS_WITH_PIN_CONFLICTS
00201    #define NB_FACTORY_SERIAL_PORT_6_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00202 #elif ( defined MOD5441X )
00203   #define NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY      NB_FACTORY_SERIAL_CAPABILITY_FULL
00204   #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY      NB_FACTORY_SERIAL_CAPABILITY_FULL
00205 #elif ( defined SB800EX )
00206    #define NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00207    #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00208 #elif ( defined NANO54415 )
00209    #define NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00210    #define NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_FULL
00211    #define NB_FACTORY_SERIAL_PORT_2_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00212    #define NB_FACTORY_SERIAL_PORT_3_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00213    #define NB_FACTORY_SERIAL_PORT_4_MODE_CAPABILITY     NB_FACTORY_SERIAL_CAPABILITY_RS232
00214 #else
00215    #error Module not supported
00216 #endif
00217
00218 /*
00219  * Use the system defined default boot UART port as the debug port.
00220  * The value of plat_def_com can be found in the platform's hal.cpp
00221  */
00222 #define NB_FACTORY_DEBUG_SERIAL_PORT_DEFAULT (int)(plat_def_com)
00223
00224 #if (defined SB800EX)
00225 #define WIFI_INSTALLED  ( 1 )    // Applies to general wifi functionality
00226 #define ACTIVITY_LEDS   ( 1 )
00227 #endif
00228
00229 /*
00230  * NetBurner MAC OUI definitions. These are used for confirming the MAC address
00231  * of a NBWIFI module.
00232  */
00233 #define NB_OUI_OCTET_0     (uint8_t)(0x00)
00234 #define NB_OUI_OCTET_1     (uint8_t)(0x03)
00235 #define NB_OUI_OCTET_2     (uint8_t)(0xf4)
00236
00237 /* Serial data settings defaults */
00238 #define NB_FACTORY_SERIAL_DATA_RATE_DEFAULT "115200"
00239 #define NB_FACTORY_SERIAL_CUSTOM_DATA_RATE_DEFAULT (0)
00240 #define NB_FACTORY_SERIAL_DATA_BITS_DEFAULT "8"
00241 #define NB_FACTORY_SERIAL_STOP_BITS_DEFAULT "1"
00242
00243 /* I2C data settings defaults */
00244 #define NB_FACTORY_I2C_BUS_SPEED_DEFAULT (100000)
00245 #define NB_FACTORY_I2C_IC_DEFAULT (0x16)
00246 #define NB_FACTORY_I2C_FREQ_DIVIDER_DEFAULT (768)
00247 #define NB_FACTORY_I2C_CUSTOM_DATA_RATE_DEFAULT (0)
00248 #define NB_FACTORY_I2C_ADDRESS_DEFAULT (0x08)
00249 #define NB_FACTORY_I2C_TCP_AS_SSL_DEFAULT (false)
00250
00251 /*  None is 1 serialburnerdata.h */
00252 #define NB_FACTORY_SERIAL_PARITY_DEFAULT "None"
00253
00254 /* SERIAL_FLOW_MODE_NONE is 1 serialburnerdata.h */
00255 #define NB_FACTORY_SERIAL_FLOW_CONTROL_DEFAULT "None"
00256
00257 #define NB_FACTORY_DO_AT_COMMANDS_DEFAULT (false)
00258
00259 #define NB_FACTORY_DISABLED_SER_PORT_PIN_FUNC_DEFAULT ("High Impedance")
00260
00261 #define NB_FACTORY_ENABLE_GPIO_HI_DRIVE_DEFAULT (false)
00262 #define NB_FACTORY_DO_GPIO_COMMANDS_DEFAULT (true)
00263 #define NB_FACTORY_GPIO_PORT_DEFAULT (1000)
00264 #define NB_FACTORY_GPIO_OVER_SSL_DEFAULT (false)
00265 #define NB_FACTORY_GPIO_SAVE_TO_FLASH_DEFAULT (true)
00266
00267 /* Serial Port Settings jumper page enabled */
00268 #define NB_FACTORY_USE_CONNECT_MSG_DEFAULT (false)
00269 #define NB_FACTORY_USE_CONNECTION_LOST_MSG_DEFAULT (false)
00270 #define NB_FACTORY_BREAK_ON_CONNECT_DEFAULT (false)
00271 #define NB_FACTORY_BREAK_INTERVAL_DEFAULT (20)
00272 #define NB_FACTORY_BREAK_KEY_FLAG_DEFAULT (false)
00273 #define NB_FACTORY_BREAK_KEY_VALUE_DEFAULT "02"
00274
00275 /* SSL */
00276 #define NB_FACTORY_SSL_PERMANENT_DESC_DEFAULT "NetBurner Library Default "
00277 #define NB_FACTORY_SSL_INCLUDED_DESC_DEFAULT "Default "
00278 #define NB_FACTORY_SSL_INSTALLED_DESC_DEFAULT "User Installed "
00279
00280 #define NB_FACTORY_SSL_FILE_NAME_CERT "cert.crt"
00281 #define NB_FACTORY_SSL_FILE_NAME_KEY "cert.key"
00282
```

```
00283 /* SSH */
00284 #define NB_FACTORY_INACTIVITY_TIMEOUT_SSH_DEFAULT (360)
00285 #define NB_FACTORY_NEW_CONNECTION_TIMEOUT_SSH_DEFAULT (180)
00286
00287 #define NB_FACTORY_SSH_PERMANENT_KEY_DESC_DEFAULT "NetBurner Library Default "
00288 #define NB_FACTORY_SSH_INCLUDED_KEY_DESC_DEFAULT "Default "
00289 #define NB_FACTORY_SSH_INSTALLED_KEY_DESC_DEFAULT "User Installed "
00290
00291 #define NB_FACTORY_SSH_FILE_NAME_KEY_RSA   "rsa.key"
00292 #define NB_FACTORY_SSH_FILE_NAME_KEY_ECDSA "ecdsa.key"
00293
00294 /* UDP */
00295 #define NB_FACTORY_ACCUMULATED_CHARS_UDP_DEFAULT (32)
00296 #define NB_FACTORY_WAIT_UDP_IN_TICKS_DEFAULT (100)
00297 #define NB_FACTORY_TRIGGER_CHAR_UDP_DEFAULT ("NA")
00298 #define NB_FACTORY_LEARN_UDP_DEFAULT (false)
00299 #define NB_FACTORY_CHECK_FRAMING_CHAR_UDP_DEFAULT (false)
00300
00301 /* TCP */
00302 #define NB_FACTORY_ACCUMULATED_CHARS_TCP_DEFAULT (32)
00303 #define NB_FACTORY_WAIT_TCP_IN_TICKS_DEFAULT (100)
00304 #define NB_FACTORY_TRIGGER_CHAR_TCP_DEFAULT ("NA")
00305 #define NB_FACTORY_LISTEN_FOR_TCP_CONNECT_DEFAULT (true)
00306 #define NB_FACTORY_CUSTOM_FRAME_TCP_DEFAULT (false)
00307 #define NB_FACTORY_CHECK_FRAMING_CHAR_TCP_DEFAULT (false)
00308
00309 /* DHCP timeout */
00310 #define NB_FACTORY_DHCP_TIMEOUT_IN_TICKS (10 * TICKS_PER_SECOND)
00311
00312 /*
00313  * Maximum size of certificate or key files in bytes
00314  *    Must the maximum of
00315  *       SERIAL_BURNER_CERTIFICATE_SIZE_MAX
00316  *       SERIAL_BURNER_RSA_KEY_SIZE_MAX
00317  *       SERIAL_BURNER_ECC_KEY_SIZE_MAX
00318  */
00319 #define NB_FACTORY_FILE_SIZE_MAXIMUM (4 * 1024)
00320
00321 /*
00322  * On-chip file system EFFS-STD
00323  *    COMPCODEFLAGS end address must be set to file system start
00324  *    (FIRST_ADDR)
00325  */
00326
00327 /* Module unique flash parameters */
00328 #if (defined MOD5441X)
00329 /* Flash */
00330 #define NB_FACTORY_FLASH_32MB_128KB_SECTORS (1)
00331 /* Base address */
00332 #define NB_FACTORY_FS_FLASHBASE (0xC0000000)
00333 /* See makefile for COMPCODEFLAGS */
00334
00335 #elif (defined SB800EX)
00336 /* Flash */
00337 #define NB_FACTORY_FLASH_SPI_8MB_4KB_SECTORS (1)
00338 /* Base address */
00339 // #define NB_FACTORY_FS_FLASHBASE                ( 0xFFC00000 )  // not used for spi flash
00340 /* See makefile for COMPCODEFLAGS */
00341
00342 #elif (defined NANO54415)
00343 /* Flash */
00344 #define NB_FACTORY_FLASH_SPI_8MB_4KB_SECTORS (1)
00345 /* Base address */
00346 //#define NB_FACTORY_FS_FLASHBASE             ( 0x040000 )  // not used for spi flash
00347 /* See makefile for COMPCODEFLAGS */
00348
00349 #elif ((defined MODM7AE70) || (defined SBE70LC))
00350 /* Flash */
00351 #define NB_FACTORY_FLASH_2MB_8KB_SECTORS (1)
00352 /* Base address */
00353 #define NB_FACTORY_FS_FLASHBASE (0x00400000)
00354 #include "flashChip/SAME70Q21.h"
00355 #elif (defined SOMRT1061)
00356    // fall through. Flash definitions are included in SOMRT1061 system source
00357 #else
00358 #error Module not supported
00359 #endif
00360
00361 #if (defined WIFI_INSTALLED)
00362 #define SYSTEM_CONFIG_RECORD_KEY_PSK_MAX_SIZE SYSTEM_CONFIG_RECORD_WPA_PSK_SIZE_MAX
00363 #endif
00364
00365 #endif /* #ifdef _NB_FACTORY_H_ */
```

24.209 SSH/SshServerUserKey/src/nbfactory.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_FACTORY_H_
00006 #define _NB_FACTORY_H_
00007
00008 /*****************************************************************************
00009  *
00010  * Factory defaults for this factory application
00011  *
00012  *****************************************************************************/
00013
00014 // Version  VV.NN.vvvv ( 0xVVNNvvvv ), string must match constant
00015 #define NB_FACTORY_DEFAULTS_VERSION (DWORD)(0x01110000)
00016 #define NB_FACTORY_DEFAULTS_VERSION_STRING "01.00"
00017
00018 // Module Base Name
00019 #if defined MOD5441X
00020 #define NB_FACTORY_MODULE_BASE_NAME "MOD5441X"
00021 #elif defined NANO54415
00022 #define NB_FACTORY_MODULE_BASE_NAME "NANO54415"
00023 #elif defined SB800EX
00024 #define NB_FACTORY_MODULE_BASE_NAME "SB800EX"
00025 #elif defined MODM7AE70
00026 #define NB_FACTORY_MODULE_BASE_NAME "MODM7AE70"
00027 #elif defined SBE70LC
00028 #define NB_FACTORY_MODULE_BASE_NAME "SBE70LC"
00029 #else
00030 #error Module not supported
00031 #endif
00032 // Module Description
00033 #define NB_FACTORY_BASE_DESC "SSH User Key Example"
00034
00035 /* Feature Name and Description */
00036 #define NB_FACTORY_FEATURE_NAME ""
00037 #define NB_FACTORY_FEATURE_DESC "Upload SSH Key Example"
00038
00039 #define NB_FACTORY_INCLUDE_SSH (1)
00040 #define SSH_KEY_SIZE_MAX_PEM ((4 * 1024) - 1)
00041
00042 #define NB_FACTORY_INACTIVITY_TIMEOUT_SSH_DEFAULT (180)
00043 #define NB_FACTORY_NEW_CONNECTION_TIMEOUT_SSH_DEFAULT (360)
00044
00045 #define NB_FACTORY_SSH_PERMANENT_KEY_DESC_DEFAULT "NetBurner Library Default "
00046 #define NB_FACTORY_SSH_INCLUDED_KEY_DESC_DEFAULT "Application Default "
00047 #define NB_FACTORY_SSH_INSTALLED_KEY_DESC_DEFAULT "User Installed "
00048
00049 #define NB_FACTORY_SSH_FILE_NAME_KEY_RSA "rsa.key"
00050 #define NB_FACTORY_SSH_FILE_NAME_KEY_ECC "ecc.key"
00051
00052 // DHCP timeout
00053 #define NB_FACTORY_DHCP_TIMEOUT_IN_TICKS (10 * TICKS_PER_SECOND)
00054
00055 /*
00056  * Maximum size of certificate or key files in bytes
00057  *    Must the maximum of
00058  *       SERIAL_BURNER_CERTIFICATE_SIZE_MAX
00059  *       SERIAL_BURNER_RSA_KEY_SIZE_MAX
00060  *       SERIAL_BURNER_ECC_KEY_SIZE_MAX
00061  */
00062 #define NB_FACTORY_FILE_SIZE_MAXIMUM (4 * 1024)
00063
00064 /*
00065  * On-chip file system EFFS-STD
00066  *    COMPCODEFLAGS end address must be set to file system start
00067  *    (FIRST_ADDR)
00068  */
00069
00070 /* Module unique flash parameters */
00071 #if defined MOD5441X
00072 /* Flash */
00073 #define NB_FACTORY_FLASH_32MB_128KB_SECTORS (1)
00074 /* Base address */
00075 #define NB_FACTORY_FS_FLASHBASE (0xC0000000)
00076 /* See makefile for COMPCODEFLAGS */
00077
00078 #elif defined NANO54415
00079 /* Flash */
00080 #define NB_FACTORY_FLASH_SPI_8MB_4KB_SECTORS (1)
00081 /* Base address */
00082 //#define NB_FACTORY_FS_FLASHBASE              ( 0x040000 )  // not used for spi flash
00083 /* See makefile for COMPCODEFLAGS */
00084
00085 #elif defined SB800EX
```

```
00086 /* Flash */
00087 #define NB_FACTORY_FLASH_SPI_8MB_4KB_SECTORS (1)
00088 /* Base address */
00089 // #define NB_FACTORY_FS_FLASHBASE              ( 0xFFC00000 )  // not used for spi flash
00090 /* See makefile for COMPCODEFLAGS */
00091
00092 #elif ((defined MODM7AE70) || (defined SBE70LC))
00093 /* Flash */
00094 #define NB_FACTORY_FLASH_2MB_8KB_SECTORS (1)
00095 /* Base address */
00096 #define NB_FACTORY_FS_FLASHBASE (0x00400000)
00097
00098 #else
00099 #error Module not supported
00100 #endif
00101
00102 #define debug_iprintf(...)                                  \
00103     {                                                       \
00104         if (bShowDebug == TRUE) { iprintf(__VA_ARGS__); } \
00105     }
00106
00107 #endif /* #ifdef _NB_FACTORY_H_ */
```

24.210 SSL/HttpsUploadCert/src/nbfactory.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_FACTORY_H_
00006 #define _NB_FACTORY_H_
00007
00008 /*
00009  ********************************************************************************
00010  *
00011  * Factory defaults for this factory application
00012  *
00013  ********************************************************************************
00014  */
00015 /* Configuration verify key (increment if data changed, added, reorganized) */
00016 #define NB_FACTORY_VERIFY_KEY (0x7E545085)
00017 /* STD-EFFS verify key (increment to format file system, required for 1.6-1.9 STD EFFS Upgrade) */
00018 #define STD_EFFS_VERIFY_KEY (0x15A58101)
00019
00020 /* Version  VV.NN.vvvv ( 0xVVNNvvvv ), string must match constant  */
00021 #define NB_FACTORY_DEFAULTS_VERSION (uint32_t)(0x01000000)
00022 #define NB_FACTORY_DEFAULTS_VERSION_STRING "01.00.0000"
00023
00024 /* Module Base Name */
00025 #if defined MOD5441X
00026 #define NB_FACTORY_MODULE_BASE_NAME "MOD5441X"
00027 #elif defined NANO54415
00028 #define NB_FACTORY_MODULE_BASE_NAME "NANO54415"
00029 #elif defined SB800EX
00030 #define NB_FACTORY_MODULE_BASE_NAME "SB800EX"
00031 #elif defined MODM7AE70
00032 #define NB_FACTORY_MODULE_BASE_NAME "MODM7AE70"
00033 #elif defined SBE70LC
00034 #define NB_FACTORY_MODULE_BASE_NAME "SBE70LC"
00035 #else
00036 #error Module not supported
00037 #endif
00038
00039 /* Module Description */
00040 #define NB_FACTORY_BASE_DESC "SSL/TLS"
00041
00042 /* Feature Name and Description */
00043 #define NB_FACTORY_FEATURE_NAME "SX"
00044 #define NB_FACTORY_FEATURE_DESC "Certificate Upload Example"
00045
00046 /* SSH */
00047 #define NB_FACTORY_INCLUDE_SSH (1)
00048
00049 /* SSL */
00050 #define NB_FACTORY_SSL_PERMANENT_DESC_DEFAULT "Not Installed"
00051 #define NB_FACTORY_SSL_INCLUDED_DESC_DEFAULT "Default "
00052 #define NB_FACTORY_SSL_INSTALLED_DESC_DEFAULT "User Installed "
00053
00054 #define NB_FACTORY_SSL_FILE_NAME_CERT "cert.crt"
00055 #define NB_FACTORY_SSL_FILE_NAME_KEY "cert.key"
00056
00057 /* DHCP timeout */
00058 #define NB_FACTORY_DHCP_TIMEOUT_IN_TICKS (10 * TICKS_PER_SECOND)
00059
00060 /*
```

```
00061  * Maximum size of certificate or key files in bytes
00062  *    Must the maximum of
00063  *       SERIAL_BURNER_CERTIFICATE_SIZE_MAX
00064  *       SERIAL_BURNER_RSA_KEY_SIZE_MAX
00065  *       SERIAL_BURNER_DSA_KEY_SIZE_MAX
00066  */
00067 #define NB_FACTORY_FILE_SIZE_MAXIMUM (4 * 1024)
00068
00069 /*
00070  * On-chip file system EFFS-STD
00071  *    COMPCODEFLAGS end address must be set to file system start
00072  *    (FIRST_ADDR)
00073  */
00074
00075 /* Module unique flash parameters */
00076 #if defined MOD5441X
00077 /* Flash */
00078 #define NB_FACTORY_FLASH_32MB_128KB_SECTORS (1)
00079 /* Base address */
00080 #define NB_FACTORY_FS_FLASHBASE (0xC0000000)
00081 /* See makefile for COMPCODEFLAGS */
00082
00083 #elif (defined NANO54415) || (defined SB800EX)
00084 /* Flash */
00085 #define NB_FACTORY_FLASH_SPI_8MB_4KB_SECTORS (1)
00086 /* Base address */
00087 //#define NB_FACTORY_FS_FLASHBASE              ( 0x040000 )  // not used for spi flash
00088 /* See makefile for COMPCODEFLAGS */
00089
00090 #elif ((defined MODM7AE70) || (defined SBE70LC))
00091 /* Flash */
00092 #define NB_FACTORY_FLASH_2MB_8KB_SECTORS (1)
00093 /* Base address */
00094 #define NB_FACTORY_FS_FLASHBASE (0x00400000)
00095
00096 #else
00097 #error Module not supported
00098 #endif
00099
00100 #endif /* #ifdef _NB_FACTORY_H_ */
```

24.211 SSL/SslClientVerifyPeerEffs/src/nbfactory.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_FACTORY_H_
00006 #define _NB_FACTORY_H_
00007
00008 /*
00009  ********************************************************************************
00010  *
00011  * Factory defaults for this factory application
00012  *
00013  ********************************************************************************
00014  */
00015 /* Configuration verify key (increment if data changed, added, reorganized) */
00016 #define NB_FACTORY_VERIFY_KEY (0x7E545085)
00017 /* STD-EFFS verify key (increment to format file system, required for 1.6-1.9 STD EFFS Upgrade) */
00018 #define STD_EFFS_VERIFY_KEY (0x15A58101)
00019
00020 /* Version  VV.NN.vvvv ( 0xVVNNvvvv ), string must match constant  */
00021 #define NB_FACTORY_DEFAULTS_VERSION (DWORD)(0x01000000)
00022 #define NB_FACTORY_DEFAULTS_VERSION_STRING "01.00.0000"
00023
00024 /* Module Base Name */
00025 #if defined MOD5441X
00026 #define NB_FACTORY_MODULE_BASE_NAME "MOD5441X"
00027 #elif defined NANO54415
00028 #define NB_FACTORY_MODULE_BASE_NAME "NANO54415"
00029 #elif defined MODM7AE70
00030 #define NB_FACTORY_MODULE_BASE_NAME "MODM7AE70"
00031 #elif defined SB800EX
00032 #define NB_FACTORY_MODULE_BASE_NAME "SB800EX"
00033 #elif defined SBE70LC
00034 #define NB_FACTORY_MODULE_BASE_NAME "SBE70LC"
00035 #else
00036 #error Module not supported
00037 #endif
00038
00039 /* Module Description */
00040 #define NB_FACTORY_BASE_DESC "SSL/TLS"
00041
00042 /* Feature Name and Description */
```

```
00043 #define NB_FACTORY_FEATURE_NAME "SX"
00044 #define NB_FACTORY_FEATURE_DESC "Verify Peer Effs Example"
00045
00046 /* SSL */
00047 #define NB_FACTORY_INCLUDE_SSL (1)
00048
00049 /* SSL */
00050 #define NB_FACTORY_SSL_PERMANENT_DESC_DEFAULT "NetBurner Library Default "
00051 #define NB_FACTORY_SSL_INCLUDED_DESC_DEFAULT "Default "
00052 #define NB_FACTORY_SSL_INSTALLED_DESC_DEFAULT "User Installed "
00053
00054 #define NB_FACTORY_SSL_FILE_NAME_CERT "cert.crt"
00055 #define NB_FACTORY_SSL_FILE_NAME_KEY "cert.key"
00056
00057 /* DHCP timeout */
00058 #define NB_FACTORY_DHCP_TIMEOUT_IN_TICKS (10 * TICKS_PER_SECOND)
00059
00060 /*
00061  * Maximum size of certificate or key files in bytes
00062  *    Must the maximum of
00063  *       SERIAL_BURNER_CERTIFICATE_SIZE_MAX
00064  *       SERIAL_BURNER_RSA_KEY_SIZE_MAX
00065  *       SERIAL_BURNER_DSA_KEY_SIZE_MAX
00066  */
00067 #define NB_FACTORY_FILE_SIZE_MAXIMUM (5 * 1024)
00068
00069 /*
00070  * On-chip file system EFFS-STD
00071  *    COMPCODEFLAGS end address must be set to file system start
00072  *    (FIRST_ADDR)
00073  */
00074
00075 /* Module unique flash parameters */
00076
00077 #if defined MOD5441X
00078 /* Flash */
00079 #define NB_FACTORY_FLASH_32MB_128KB_SECTORS (1)
00080 /* Base address */
00081 #define NB_FACTORY_FS_FLASHBASE (0xC0000000)
00082 /* See makefile for COMPCODEFLAGS */
00083
00084 #elif ((defined NANO54415) || (defined SB800EX))
00085 /* Flash */
00086 #define NB_FACTORY_FLASH_SPI_8MB_4KB_SECTORS (1)
00087 /* Base address */
00088 //#define NB_FACTORY_FS_FLASHBASE              ( 0x040000 )  // not used for spi flash
00089 /* See makefile for COMPCODEFLAGS */
00090
00091 #elif ((defined MODM7AE70) || (defined SBE70LC))
00092 /* Flash */
00093 #define NB_FACTORY_FLASH_2MB_8KB_SECTORS (1)
00094 /* Base address */
00095 #define NB_FACTORY_FS_FLASHBASE (0x00400000)
00096
00097 #else
00098 #error Module not supported
00099 #endif
00100
00101 #endif /* #ifdef _NB_FACTORY_H_ */
```

24.212 permanentcert.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *   Permanent certificate
00007  *      Must be paired with permanent certificate key (permanentcertkey.h)
00008  *
00009  */
00010
00011 #ifndef _PERMANENTCERT_H_
00012 #define _PERMANENTCERT_H_
00013
00014 "-----BEGIN CERTIFICATE-----\r\
00015 MIIC7DCCApagAwIBAgIBADANBgkqhkiG9w0BAQQFADCBiDELMAkGA1UEBhMCVVMx\r\
00016 EzARBgNVBAgTCkNhbGlmb3JuaWExEjAQBgNVBAcTCVNhbiBEaWVnbzEYMBYGA1UE\r\
00017 ChMPTmV0QnVybmVyLCBJbmMuMRIwEAYDVQQDEwlOZXRCdXJuZXIxIjAgBgkqhkiG\r\
00018 9w0BCQEWE3NhbGVzQG5ldGJ1cm5lci5jb20wHhcNMDgwODI3MTcxMDQxWhcNMTgw\r\
00019 ODI1MTcxMDQxWjCBiDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx\r\
00020 EjAQBgNVBAcTCVNhbiBEaWVnbzEYMBYGA1UEChMPTmV0QnVybmVyLCBJbmMuMRIw\r\
00021 EAYDVQQDEwlOZXRCdXJuZXIxIjAgBgkqhkiG9w0BCQEWE3NhbGVzQG5ldGJ1cm5l\r\
00022 ci5jb20wXDANBgkqhkiG9w0BAQEFAANLADBIAkEA7hC9uEH7Bv/Jj2WZVIYqxSht\r\
00023 nb27TszV7o+kMJgBN744Er5llGNlLDpDuuiN3uTwdVnrwj8TCLAQeIjLE6DHUQID\r\
```

```
00024 AQABo4HoMIHlMB0GA1UdDgQWBBRQpzLTD0kF5FyAu8iIBV7uk2jKWzCBtQYDVR0j\r\
00025 BIGtMIGqgBRQpzLTD0kF5FyAu8iIBV7uk2jKW6GBjqSBizCBiDELMAkGA1UEBhMC\r\
00026 VVMxEzARBgNVBAgTCkNhbGlmb3JuaWExEjAQBgNVBAcTCVNhbiBEaWVnbzEYMBYG\r\
00027 A1UEChMPTmV0QnVybmVyLCBJbmMuMRIwEAYDVQQDEwlOZXRCdXJuZXIxIjAgBgkq\r\
00028 hkiG9w0BCQEWE3NhbGVzQG5ldGJ1cm5lci5jb22CAQAwDAYDVR0TBAUwAwEB/zAN\r\
00029 BgkqhkiG9w0BAQQFAANBAC3mhcTjlaRWQZF0fSW5Au9BKuQqxL4x/d84DzeLtH3X\r\
00030 oAzAvYlyCh451FyMokod8RqyWT4j8Nmxya2fPKl+FRk=\r\
00031 -----END CERTIFICATE-----\r"
00032
00033 #endif /*_PERMANENTCERT_H_*/
```

24.213 permanentcertkey.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *   Permanent key accompanying cetificate
00007  *      Must be paired with permanent certificate (permanentcert.h)
00008  *
00009  */
00010
00011 #ifndef _PERMANENTCERTKEY_H_
00012 #define _PERMANENTCERTKEY_H_
00013
00014 "-----BEGIN RSA PRIVATE KEY-----\r\
00015 MIIBOwIBAAJBAO4QvbhB+wb/yY9lmVSGKsUobZ29u07M1e6PpDCYATe+OBK+ZdRj\r\
00016 dSw6Q7rojd7k8HVZ68I/EwiwEHiIyxOgx1ECAwEAAQJBAIwRH36sm2Rn1JBhB5C6\r\
00017 i6mKeswHWJApZmlA7CqHFJhVWSn7VJvgOpV7LudSlF8sbu3y6JLiYxOolS98qDKc\r\
00018 PB0CIQD4DmNuYfnbJL/GPBSQvrH+ZtKQwLp5BtVe8DYdxk7q1wIhAPWwcjaLVkkG\r\
00019 y9L/ffrzI+TShK7d4sI2ux5chn2+bAIXAiEA5VT4HF88jpxFKNL2+HM1QExztBxa\r\
00020 yfXFYsaWYaSr5XUCIEWb4EAj6iZ2jnkRehgHmzvRiVYh94UfDJTlkhU+RkuxAiBY\r\
00021 Y2roCSCyewcHGCAIaJn/RnDovcw479QXrqTD+YXQ1w==\r\
00022 -----END RSA PRIVATE KEY-----\r"
00023
00024 #endif /*_PERMANENTCERTKEY_H_*/
```

24.214 permanentkeyecdsa.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *   Default SSH ECDSA key in openSSL (openSSH) format
00007  *   OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00008  *
00009  */
00010
00011 #ifndef _PERMANENTKEYECDSA_H_
00012 #define _PERMANENTKEYECDSA_H_
00013
00014 "-----BEGIN EC PRIVATE KEY-----\r\
00015 MHcCAQEEIJQCd0y9U8mFVUWqJUNRX2FYkPTBuuYu1MFSm6QaGxA7oAoGCCqGSM49\r\
00016 AwEHoUQDQgAE++54xjrZKCfpmMYqaEdP2w0fTkXC6FEuWp1sg/uuOXiF2cS7GHY/\r\
00017 6PYVqh7SjIH+1QUxaRKlsIhXsIguY0tW3w==\r\
00018 -----END EC PRIVATE KEY-----\r"
00019
00020 #endif /*_PERMANENTKEYECDSA_H_*/
```

24.215 SecureSerToEthFactoryApp/src/permanentkeyrsa.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *   Default SSH RSA key in openSSL (openSSH) format
00007  *   OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00008  *
00009  */
00010
00011 #ifndef _PERMANENTKEYRSA_H_
00012 #define _PERMANENTKEYRSA_H_
00013
00014 "-----BEGIN RSA PRIVATE KEY-----\r\
00015 MIIBOgIBAAJBAOrfRkFnPMI0K41ufL1HLzlpf2yieGLSGE8kL2OQjX0Pp4Qq+91F\r\
00016 DRYD1YuKiPfjxsAkVBqlY7v23ZvzEfNcgDUCAwEAAQJAaFT2KGdrnfj+v7ysvIe6\r\
```

```
00017 eo5ahC9Hut4I3l78jgXQVBSeMhatb+RMyuSshgGq3+2ph6EQQABBstvuWwl5AAkU\r\
00018 oQIhAPtpCjpqiAQtqo1u64T/Pr5fX2IuzmbOhIvW8czDdKF3AiEA7yjxoEGMl+8o\r\
00019 4v8pLFZqR0s4P4G/wgScuqtCPLtjtrMCICrH5QWrux1669rFVS58gKDEeearMFQu\r\
00020 MD/bg6nkWKRhAiBTMuwz8vnFFUclCN069mkMmkdcGHgsN8yKR+/IDuyWbwIhAKZ9\r\
00021 KgZz3UZCnWHDXaelDFJI+Xdstx5XwBdTAlqwOU+L\r\
00022 -----END RSA PRIVATE KEY-----\r"
00023
00024 #endif /*_PERMANENTKEYRSA_H_*/
```

24.216 SshServerUserKey/src/permanentkeyrsa.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *   Default SSH RSA key in openSSL (openSSH) format
00007  *   OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00008  *
00009  */
00010
00011 #ifndef _PERMANENTKEYRSA_H_
00012 #define _PERMANENTKEYRSA_H_
00013
00014 "-----BEGIN RSA PRIVATE KEY-----\r\
00015 MIIBOgIBAAJBAOrfRkFnPMI0K41ufL1HLzlpf2yieGLSGE8kL2OQjX0Pp4Qq+91F\r\
00016 DRYD1YuKiPfjxsAkVBqlY7v23ZvzEfNcgDUCAwEAAQJAaFT2KGdrnfj+v7ysvIe6\r\
00017 eo5ahC9Hut4I3l78jgXQVBSeMhatb+RMyuSshgGq3+2ph6EQQABBstvuWwl5AAkU\r\
00018 oQIhAPtpCjpqiAQtqo1u64T/Pr5fX2IuzmbOhIvW8czDdKF3AiEA7yjxoEGMl+8o\r\
00019 4v8pLFZqR0s4P4G/wgScuqtCPLtjtrMCICrH5QWrux1669rFVS58gKDEeearMFQu\r\
00020 MD/bg6nkWKRhAiBTMuwz8vnFFUclCN069mkMmkdcGHgsN8yKR+/IDuyWbwIhAKZ9\r\
00021 KgZz3UZCnWHDXaelDFJI+Xdstx5XwBdTAlqwOU+L\r\
00022 -----END RSA PRIVATE KEY-----\r"
00023
00024 #endif /*_PERMANENTKEYRSA_H_*/
```

24.217 SSH/SecureSerToEthFactoryApp/src/serialburnerdata.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SERIALBURNERDATA_H_
00006 #define _SERIALBURNERDATA_H_
00007
00008 /*
00009  ****************************************************************************
00010  *
00011  * Include Files
00012  *
00013  ****************************************************************************
00014  */
00015 #include <snmp.h>
00016 #include <netbios.h>
00017 #include <config_obj.h>
00018 #include <netinterface.h>
00019 #include <fdprintf.h>
00020 #include "nbfactory.h"
00021 #include "serialburnerdata.h"
00022
00023 /*
00024  ****************************************************************************
00025  *
00026  * Definitions
00027  *
00028  ****************************************************************************
00029  */
00030 /* Constants that go with Protocol and selection list ProtocolList */
00031 #define PROTOCOL_UDP_TO_SERIAL     "UDP"
00032 #define PROTOCOL_TCP_SSL_TO_SERIAL "TCP/SSL"
00033 #define PROTOCOL_TCP_TO_SERIAL     "TCP"
00034 #define PROTOCOL_SSH "SSH"
00035
00036 /* Constants that go with SerialMode and selection list SerialModeList */
00037 #define SERIAL_MODE_RS232 "RS-232"
00038 #define SERIAL_MODE_DEBUG "DEBUG"
00039 #define SERIAL_MODE_485H "RS-485 Half Duplex"
00040 #define SERIAL_MODE_485F "RS-485 Full Duplex"
00041 #define SERIAL_MODE_422F "RS-422 Full Duplex"
00042 #define SERIAL_MODE_485H_ECHO "RS-485 Half Duplex with Echo"        // Used by PK70 quad UART blade
      board - S/W configurable
```

```
00043 #define SERIAL_MODE_485H_NOECHO "RS-485 Half Duplex without Echo"   // Used by PK70 quad UART blade
      board - S/W configurable
00044 #define SERIAL_MODE_DISABLED "Disabled"
00045
00046 #define SERIAL_MODE_DISABLED_PINS_OUTPUT_HIGH "Output High"
00047 #define SERIAL_MODE_DISABLED_PINS_OUTPUT_LOW "Output Low"
00048 #define SERIAL_MODE_DISABLED_PINS_HIZ "High Impedance"
00049
00050 /* Constants that go with serial FlowMode and selection list FlowModeList */
00051 #define SERIAL_FLOW_MODE_NONE "None"
00052 #define SERIAL_FLOW_MODE_XON_OFF "XON/XOFF"
00053 #define SERIAL_FLOW_MODE_RTS_CTS "RTS/CTS"
00054
00055 /* Constants that go with serial ConnectMode */
00056 #define SERIAL_CONNECT_CONNECT_NEVER "Never"
00057 #define SERIAL_CONNECT_CONNECT_AT_POWERUP "On power-up"
00058 #define SERIAL_CONNECT_CONNECT_WHEN_DATARX "If serial data received"
00059
00060 /* Constants that go with IP_Addr_mode */
00061 /* These strings need to match the string defined in the ip4.mode config_chooser variable defined in
      the InterfaceBlock */
00062 #define IP_ADDR_MODE_DHCP "DHCP"
00063 #define IP_ADDR_MODE_DHCP_W_FALLBACK "DHCP w Fallback"
00064 #define IP_ADDR_MODE_STATIC "Static"
00065 #define IP_ADDR_MODE_DISABLED "Disabled"
00066
00067 /* Constants that go with character strings */
00068 #define CONNECT_NAME_LENGTH (79)
00069 #define CONNECT_MESSAGE_LENGTH (79)
00070 #define CONNECT_LOSS_MESSAGE_LENGTH (79)
00071 #define DEVICE_NAME_LENGTH (15)
00072 #define NTP_NAME_LENGTH (35)
00073 #define USERNAME_LENGTH (39)
00074 #define PASSWORD_LENGTH (39)
00075 #define MAX_NUMERICAL_USER_INPUT_LENTH (32)
00076 #define MAX_STRING_USER_INPUT_LENTH (64)
00077
00078 /* Constants that define buffer sizes */
00079 #define MAX_UDP_RX_BUFFER (1500)
00080 #define BUFFER_SIZE (20000)
00081
00082 /* Constants that define task priorities */
00083 #define LED_TASK_PRIO (MAIN_PRIO - 1)
00084 #define SNMP_PRIO (MAIN_PRIO - 2)
00085 #define AT_COM_PRIO (MAIN_PRIO - 3)
00086 #define I2C_SERVER_TASK_PRIO (MAIN_PRIO + 1)
00087 #define GPIO_SERVER_TASK_PRIO (MAIN_PRIO + 2)
00088 #define START_WIFI_TASK_PRIO (OS_MAX_PRIOS - 2)
00089
00090 #ifdef NB_FACTORY_INCLUDE_SSH
00091
00092 /* Constants that go with SSH character strings */
00093 #define SSH_USERNAME_LENGTH (39)
00094 #define SSH_PASSWORD_LENGTH (39)
00095
00096 #endif /* #ifdef NB_FACTORY_INCLUDE_SSH */
00097
00098 extern MonitorRecord monitor_config;
00099
00100 #ifdef NB_FACTORY_INCLUDE_SSL
00101
00102 /*
00103  ****************************************************************************
00104  * Key or certificate size
00105  *       SSL Certificate size 2200 (empirical)
00106  *       OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00107  *       NULL terminated for conversion
00108  *
00109  ****************************************************************************
00110  */
00111 #define SERIAL_BURNER_CERTIFICATE_SIZE_MAX ((2 * 1024) - 1)
00112 #define SERIAL_BURNER_CERTIFICATE_SIZE_MAX_PEM ((3 * 1024) - 1)
00113
00114 #endif /* #ifdef NB_FACTORY_INCLUDE_SSL    */
00115
00116 #if defined(NB_FACTORY_INCLUDE_SSH) || defined(NB_FACTORY_INCLUDE_SSL)
00117
00118 /* Certificate and key status */
00119 #define SERIAL_BURNER_LIBRARY_DEFAULT ((uint8_t)0x00)
00120 #define SERIAL_BURNER_DEFAULT ((uint8_t)0x01)
00121 #define SERIAL_BURNER_USER_INSTALLED ((uint8_t)0x02)
00122
00123 /*
00124  ****************************************************************************
00125  * Key size
00126  *       SSH MAX_PRIVKEY_SIZE 1700 (options.h)
00127  *       SSH key size (PEM) < 4K (empirical)
```

```
00128  *       NULL terminated for conversion
00129  *
00130  ******************************************************************************
00131  */
00132 #define SERIAL_BURNER_KEY_SIZE_MAX_PEM ((4 * 1024) - 1)
00133
00134 #endif /* #if defined(NB_FACTORY_INCLUDE_SSH) || defined(NB_FACTORY_INCLUDE_SSL) */
00135 /*
00136  ******************************************************************************
00137  * HTML support
00138  ******************************************************************************
00139  */
00140 /* HTML data sizes */
00141 #define SERIAL_BURNER_HTML_INPUT_NUMBER_SIZE (5)
00142
00143 /* HTML data sizes */
00144 #define SERIAL_BURNER_HTML_HEX_NUMBER_LENGTH (2)
00145
00146 /* The following defines the webpage table width in percentage of the column that
00147  * includes text description of each input field. It allows platforms with two
00148  * ports or less to use more of the table width to fit text, improving readability.
00149  */
00150 #if ((NB_FACTORY_SERIAL_PORTS + NB_FACTORY_I2C_PORTS) <= 2)
00151 #define SERIAL_BURNER_DESCRIPTION_COLUMN_WIDTH_PERCENT (40)
00152 #else
00153 #define SERIAL_BURNER_DESCRIPTION_COLUMN_WIDTH_PERCENT (30)
00154 #endif
00155
00156 #define SERIAL_BURNER_PORT_COLUMN_WIDTH_PERCENT ((100 -
      SERIAL_BURNER_DESCRIPTION_COLUMN_WIDTH_PERCENT) / (NB_FACTORY_SERIAL_PORTS + NB_FACTORY_I2C_PORTS))
00157
00158 /* HTML I2C column width percentage */
00159 #define SERIAL_BURNER_HTML_I2C_WIDTH (70)
00160
00161 /* HTML input name maximum size */
00162 #define SERIAL_BURNER_HTML_INPUT_SIZE (39)
00163
00164 /* HTML IP address string size */
00165 #define SERIAL_BURNER_HTML_IP_STRING_SIZE (20)
00166
00167 /* Password protection string (SSH_PASSWORD_LENGTH length ) */
00168 #define SERIAL_BURNER_PASSWORD_STRING "******************"
00169
00170 /*
00171  ******************************************************************************
00172  * Booting support
00173  ******************************************************************************
00174  */
00175 #define boot_iprintf(...)                                          \
00176     {                                                              \
00177         if (monitor_config.Quiet == 0)                             \
00178         {                                                          \
00179             int ifNum = GetFirstInterface();                       \
00180             InterfaceBlock *pIfB = GetInterfaceBlock(ifNum);       \
00181             if (pIfB != nullptr)                                   \
00182             {                                                      \
00183                 iprintf("%s : ", pIfB->device_name.c_str());       \
00184                 iprintf(__VA_ARGS__);                              \
00185                 iprintf("\r\n");                                   \
00186             }                                                      \
00187         }                                                          \
00188     }
00189
00190 /*
00191  ******************************************************************************
00192  * Debug support
00193  ******************************************************************************
00194  */
00195 #define debug_iprintf(...)                                             \
00196     {                                                                  \
00197         if (bShowDebug == true)                                        \
00198         {                                                              \
00199             int ifNum = GetFirstInterface();                           \
00200             InterfaceBlock *pIfB = GetInterfaceBlock(ifNum);           \
00201             if (pIfB != nullptr)                                       \
00202             {                                                          \
00203                 iprintf("%s [debug] : ", pIfB->device_name.c_str());   \
00204                 iprintf(__VA_ARGS__);                                  \
00205                 iprintf("\r\n");                                       \
00206             }                                                          \
00207         }                                                              \
00208     }
00209
00210 /*
00211  ******************************************************************************
00212  *
00213  * Structures
```

```
00214  *
00215  ******************************************************************************
00216  */
00217 // #ifndef SNMP
00218 //    #include <snmp.h>
00219 //    struct SysInfo
00220 //    {
00221 //        char SysContact[ 256 ];
00222 //        char SysName[ 256 ];
00223 //        char SysLocation[ 256 ];
00224 //        unsigned char ReadCommunity[ 40 ];
00225 //        unsigned char WriteCommunity[ 40 ];
00226 //        IPADDR trap_destination;
00227 //        uint32_t trap_enable_flags;
00228 //        uint32_t valid;
00229 //    };
00230 // #endif   /* SNMP */
00231
00232 #ifndef SNMP
00233 #include <snmp.h>
00234 class SysInfo : public config_obj
00235 {
00236    public:
00237     config_string SysContact{"", "Sys Contact"};
00238     config_string SysName{"", "Sys Name"};
00239     config_string SysLocation{"", "Sys Location"};
00240     config_string ReadCommunity{"", "Read Community"};
00241     config_string WriteCommunity{"", "Write Community"};
00242     config_IPADDR4 trap_destination{"", "Trap Destination"};
00243     config_uint trap_enable_flags{0, "Trap Enable Flags"};
00244     config_uint valid{0, "Valid"};
00245     ConfigEndMarker;
00246
00247     SysInfo(const char *name, const char *desc = nullptr) : config_obj(name, desc){};
00248     SysInfo(config_obj &owner, const char *name, const char *desc = nullptr) : config_obj(owner, name,
      desc){};
00249     SysInfo & operator=(const SysInfo & si)
00250     {
00251         SysContact = si.SysContact;
00252         SysName = si.SysName;
00253         SysLocation = si.SysLocation;
00254         ReadCommunity = si.ReadCommunity;
00255         WriteCommunity = si.WriteCommunity;
00256         trap_destination = si.trap_destination;
00257         trap_enable_flags = si.trap_enable_flags;
00258         valid = si.valid;
00259
00260         return *this;
00261     }
00262 };
00263 #endif /* SNMP */
00264
00265 /*
00266       Port Setting (Protocol)
00267
00268          * Listening socket (All) *
00269          ListenPort              - Listen port
00270          InactivityTimeoutInSecs - Inactivity timeout in seconds
00271          New_connection_timeout  - Connection timeout
00272          ConnectMode             - Serial connect mode
00273
00274          * Connect on power-up and received serial data choices (TCP) *
00275          ConnectPort             - Port
00276          ConnectName             - Target name for GetHostByName
00277          ConnectAddress          - Target IP address
00278          ConnectIdleTimeout    - Idle timeout in seconds
00279          Connection_retry_timeout- Retry interval in seconds
00280          keepAliveInterval       - Check and maintain the connection in second intervals
00281
00282          * Primary connection settings - serial port settings (All) *
00283          SerialMode              - Type port
00284          DataBaudRate            - Baud rate
00285          Output_Bits             - Data bits per byte
00286          Output_Parity           - Parity
00287          Output_Stop             - Stop bits
00288          FlowMode                - Flow control
00289
00290          * Primary connection settings - serial data notification settings (TCP, UDP) *
00291          ConnectMessage          - Serial message upon connection
00292          ConnectLossMessage      - Serial message upon connection loss
00293          BreakOnConnect          - Send serial break on connection
00294          BreakInterval           - Break interval
00295          BreakKeyFlag            - Send data then a break
00296          BreakKeyValue           - Data prior to break
00297
00298          * UDP *
00299          UdpAccumulatedChars     - Char. to accumulate before sending packet
```

```
00300          UdpWait                 - Accumulation wait in ticks
00301          UdpTriggerChar          - Flushing character
00302          bLearnUdp               - Learn IP address from connection
00303          bUdpCheckFramingChar    - UDP framing character to check
00304
00305          * Custom Framing (TCP, SSH) *
00306          bTcpCustomFrame          - Enabled?
00307          TcpAccumulatedChars      - Char. to accumulate before sending packet
00308          TcpWait                  - Accumulation wait in ticks
00309          TcpTriggerChar           - Serial flushing character
00310
00311          * Framing (TCP) *
00312          bTcpCheckFramingChar    - TCP framing character to check
00313
00314 */
00315 class NV_OnePortSetting : public config_obj
00316 {
00317    public:
00318     config_uint ListenPort{NB_FACTORY_LISTEN_PORT_DEFAULT, "Listen Port"};
00319     config_uint InactivityTimeoutInSecs{NB_FACTORY_INACTIVITY_TIMEOUT_DEFAULT, "Inactivity
      Timeout(seconds)"};
00320     config_uint New_connection_timeout{NB_FACTORY_NEW_CONNECTION_TIMEOUT_DEFAULT, "New Connection
      Timeout(seconds)"};
00321     config_chooser ConnectMode{"Connect Mode", "Never", "Never,On power-up,If serial data received"};
00322     config_uint ConnectPort{NB_FACTORY_CONNECT_PORT_DEFAULT, "Connect Port"};
00323     config_string ConnectName1{"", "Connect Name 1"};
00324     config_string ConnectName2{"", "Connect Name 2"};
00325     config_IPADDR4 ConnectAddress1{"0.0.0.0", "Connect Address 1"};
00326     config_IPADDR4 ConnectAddress2{"0.0.0.0", "Connect Address 2"};
00327     config_uint ConnectIdleTimeout{NB_CONNECT_IDLE_TIMEOUT_DEFAULT, "Connect Idle Timeout(seconds)"};
00328     config_uint Connection_retry_timeout{NB_CONNECT_RETRY_TIMEOUT_DEFAULT, "Connection Retry
      Timeout(seconds)"};
00329     config_uint keepAliveInterval{NB_CONENCT_KEEP_ALIVE_INTERVAL_DEFAULT, "Keep Alive
      Interval(seconds)"};
00330     // Port-TODO: Hook in system serial port configurations so that any change made in the Factory App
      also change the system's serial port
00331     // config variable.
00332     config_chooser SerialMode{"Serial Mode", NB_FACTORY_DEFAULT_SERIAL_MODE,
      NB_FACTORY_SERIAL_CAPABILITY_RS232};
00333     config_chooser DataBaudRate{
00334         "Data Baud Rate", NB_FACTORY_SERIAL_DATA_RATE_DEFAULT,
00335
      "921600,500000,460800,345600,256000,230400,128000,115200,57600,38400,32000,19200,9600,4800,2400,1200,Custom"};
00336     config_uint CustomBaudRate{NB_FACTORY_SERIAL_CUSTOM_DATA_RATE_DEFAULT, "Custom Baudrate"};
00337     config_chooser Output_Bits{"Output Bits", NB_FACTORY_SERIAL_DATA_BITS_DEFAULT, "8,7,6,5"};
00338     config_chooser Output_Parity{"Output Parity", NB_FACTORY_SERIAL_PARITY_DEFAULT, "None,Odd,Even"};
00339     config_chooser Output_Stop{"Output Stop", NB_FACTORY_SERIAL_STOP_BITS_DEFAULT, "1,2"};
00340     config_chooser FlowMode{"Flow Control Mode", NB_FACTORY_SERIAL_FLOW_CONTROL_DEFAULT,
      "None,XON/XOFF,RTS/CTS"};
00341     config_bool bDoAtCommand{NB_FACTORY_DO_AT_COMMANDS_DEFAULT, "Do AT Commands"};
00342     config_chooser DisabledPortPinFunction{"Disabled Port Pin Function",
      NB_FACTORY_DISABLED_SER_PORT_PIN_FUNC_DEFAULT, "Output High,Output Low,High Impedance"};
00343     config_string ConnectMessage{"", "Connect Message"};
00344     config_bool bUseConnectMessage{NB_FACTORY_USE_CONNECT_MSG_DEFAULT, "Use the Configured Connect
      Message"};
00345     config_string ConnectLossMessage{"", "Lost Connection Message"};
00346     config_bool bUseConnectLossMessage{NB_FACTORY_USE_CONNECTION_LOST_MSG_DEFAULT, "Use the Configured
      Lost Connection Message"};
00347     config_bool BreakOnConnect{NB_FACTORY_BREAK_ON_CONNECT_DEFAULT, "Break On Connect"};
00348     config_uint BreakInterval{NB_FACTORY_BREAK_INTERVAL_DEFAULT, "Break Interval"};
00349     config_bool BreakKeyFlag{NB_FACTORY_BREAK_KEY_FLAG_DEFAULT, "Break Key Flag"};
00350     config_string BreakKeyValue{NB_FACTORY_BREAK_KEY_VALUE_DEFAULT, "Break Key Value"};
00351     config_uint UdpAccumulatedChars{NB_FACTORY_ACCUMULATED_CHARS_UDP_DEFAULT, "UDP Accumulated
      Chars"};
00352     config_uint UdpWait{NB_FACTORY_WAIT_UDP_IN_TICKS_DEFAULT, "UDP Wait(ticks)"};
00353     config_uint TcpAccumulatedChars{NB_FACTORY_ACCUMULATED_CHARS_TCP_DEFAULT, "TCP Accumulated
      Chars"};
00354     config_uint TcpWait{NB_FACTORY_WAIT_TCP_IN_TICKS_DEFAULT, "TCP Wait(ticks)"};
00355     config_string UdpTriggerChar{NB_FACTORY_TRIGGER_CHAR_UDP_DEFAULT,
00356                                  "UDP Trigger Character"};
00357     config_string TcpTriggerChar{NB_FACTORY_TRIGGER_CHAR_TCP_DEFAULT,
00358                                  "TCP Trigger Character"};   // Port-TODO: need to test char to
      config_uint comparison when parsing
00359     config_bool bListenForTCPConnections{NB_FACTORY_LISTEN_FOR_TCP_CONNECT_DEFAULT, "Listen for
      incoming network connections"};
00360     config_bool bTcpCustomFrame{NB_FACTORY_CUSTOM_FRAME_TCP_DEFAULT, "TCP Custom Frame"};
00361     config_bool bLearnUdp{NB_FACTORY_LEARN_UDP_DEFAULT, "Learn UDP"};
00362     config_bool bUdpCheckFramingChar{NB_FACTORY_CHECK_FRAMING_CHAR_UDP_DEFAULT, "Check UDP Framing
      Character"};
00363     config_bool bTcpCheckFramingChar{NB_FACTORY_CHECK_FRAMING_CHAR_TCP_DEFAULT, "Check TCP Framing
      Character"};
00364     config_bool bAlwaysStoreSerialChars{false, "Always Store Serial Characters"};
00365 #ifdef SB800EX
00366     config_bool bDTRReflectsConnection{false, "DTR Reflects Connection"};   // Port-TODO: requires
      logic based on port number to initialize. See setDefaultsForUDP()
00367 #endif /* SB800EX */
00368     config_bool bTCPasSSL{false, "Allow SSL on TCP Socket"};
```

```
00369     ConfigEndMarker;
00370
00371     NV_OnePortSetting(const char *name,
00372                       const char *desc = nullptr,
00373                       const char *serialMode = nullptr,
00374                       const char *serialModeChoices = nullptr)
00375         : config_obj(name, desc)
00376     {
00377         if (serialModeChoices != nullptr) { SerialMode.SetChoices(serialModeChoices); }
00378         if (serialMode != nullptr)
00379         {
00380             SerialMode = serialMode;
00381
00382             if (strncmp(serialMode, SERIAL_MODE_DEBUG, sizeof(SERIAL_MODE_DEBUG)) == 0)
00383             {
00384                 // Disable AT command parser on a debug serial port
00385                 bDoAtCommand = false;
00386             }
00387         }
00388     };
00389
00390     NV_OnePortSetting(config_obj &owner,
00391                       const char *name,
00392                       const char *desc = nullptr,
00393                       const char *serialMode = nullptr,
00394                       const char *serialModeChoices = nullptr)
00395         : config_obj(owner, name, desc)
00396     {
00397         if (serialModeChoices != nullptr) { SerialMode.SetChoices(serialModeChoices); }
00398         if (serialMode != nullptr)
00399         {
00400             SerialMode = serialMode;
00401
00402             if (strncmp(serialMode, SERIAL_MODE_DEBUG, sizeof(SERIAL_MODE_DEBUG)) == 0)
00403             {
00404                 // Disable AT command parser on a debug serial port
00405                 bDoAtCommand = false;
00406             }
00407         }
00408     };
00409 };
00410
00411 /* ic  -  The byte value written to the I2FDR register of the processor
00412  *        to reach a desired frequency divider. Used with InitI2C() to
00413  *        initialize the bus to a particular I2C bus speed
00414  * FreqDivider - The frequency divider that corresponds to the byte value in ic.
00415  *               For a table of ic-frequency divider pairs, refer to the reference manual for the
00416  *               processor - specifically, refer to the I2FDR section of the
00417  *               I2C chapter.
00418  */
00419 class NV_OneI2CPortSetting : public config_obj
00420 {
00421    public:
00422     // I2C parameters
00423     config_uint ic{NB_FACTORY_I2C_IC_DEFAULT, "ic"};
00424     config_uint FreqDivider{NB_FACTORY_I2C_FREQ_DIVIDER_DEFAULT, "Frequency Divider"};
00425     config_uint CustomBaudRate{0, "Custom Baud Rate"};
00426     config_uint BusSpeed{100000, "I2C Bus Speed"};
00427     config_uint SetMasterAddress{NB_FACTORY_I2C_ADDRESS_DEFAULT, "Set Master Address"};
00428     config_uint SaveMasterAddress{NB_FACTORY_I2C_ADDRESS_DEFAULT, "Save Master Address"};
00429
00430     // TCP parameters
00431     config_uint ListenPort{NB_FACTORY_I2C_LISTEN_PORT_DEFAULT, "Listen Port"};
00432     config_bool bListenCheck{NB_FACTORY_LISTEN_FOR_TCP_CONNECT_DEFAULT, "Listen for incoming network
      connections"};
00433     config_uint InactivityTimeoutInSecs{NB_FACTORY_INACTIVITY_TIMEOUT_DEFAULT, "Inactivity
      Timeout(seconds)"};
00434     config_uint New_connection_timeout{NB_FACTORY_NEW_CONNECTION_TIMEOUT_DEFAULT, "New Connection
      Timeout(seconds)"};
00435     config_uint ConnectIdleTimeout{NB_CONNECT_IDLE_TIMEOUT_DEFAULT, "Connect Idle Timeout(seconds)"};
00436     config_bool bTCPasSSL{false, "Allow TCP as SSL"};
00437     ConfigEndMarker;
00438
00439     NV_OneI2CPortSetting(const char *name, const char *desc = nullptr) : config_obj(name, desc){};
00440     NV_OneI2CPortSetting(config_obj &owner, const char *name, const char *desc = nullptr) :
      config_obj(owner, name, desc){};
00441 };
00442
00443 /* Refer to the I2FDR section of the reference manual for the processor in use
00444  * for a table of dividers and the respective ICs's */
00445 struct BaudRate
00446 {
00447     unsigned char ic;
00448     int divider;
00449 };
00450
00451
```

```
00452 extern const char* defaultSerialPortModes[];
00453
00454 class NV_SettingsStruct : public config_obj
00455 {
00456    public:
00457     /* NetBurner protocol */
00458     config_chooser Protocol{"Protocol", "TCP/SSL", "UDP,TCP/SSL,SSH,TCP"};
00459     config_chooser ConnectMode{"Connect Mode", "Never", "Never,On power-up,If serial data received"};
00460
00461     /* NetBurner address configuration */
00462     config_string NetBIOSName{"", "NetBIOS Name"};
00463     config_string NTPName{NB_FACTORY_NTPSERVERNAME_DEFAULT, "NTP Name"};   // NTPName[
      NTP_NAME_LENGTH+ 1 ];
00464     config_IPADDR4 NTP_Addr{"0.0.0.0", "NTP IPv4 Address"};
00465
00466     NV_OnePortSetting port0{"Serial Port 0", nullptr, defaultSerialPortModes[0],
      NB_FACTORY_SERIAL_PORT_0_MODE_CAPABILITY};
00467 #if !(defined SOMRT1061)
00468     NV_OnePortSetting port1{"Serial Port 1", nullptr, defaultSerialPortModes[1],
      NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY};
00469 #endif
00470
00471 #if (defined NANO54415)
00472     NV_OnePortSetting port2{"Serial Port 2", nullptr, defaultSerialPortModes[2],
      NB_FACTORY_SERIAL_PORT_2_MODE_CAPABILITY};
00473     NV_OnePortSetting port3{"Serial Port 3", nullptr, defaultSerialPortModes[3],
      NB_FACTORY_SERIAL_PORT_3_MODE_CAPABILITY};
00474     NV_OnePortSetting port4{"Serial Port 4", nullptr, defaultSerialPortModes[4],
      NB_FACTORY_SERIAL_PORT_4_MODE_CAPABILITY};
00475 #elif ((defined MODM7AE70 && defined USE_E70_UART_SERIAL_PORTS))
00476     NV_OnePortSetting port2{"Serial Port 2", nullptr, defaultSerialPortModes[2],
      NB_FACTORY_SERIAL_PORT_2_MODE_CAPABILITY};
00477     NV_OnePortSetting port3{"Serial Port 3", nullptr, defaultSerialPortModes[3],
      NB_FACTORY_SERIAL_PORT_3_MODE_CAPABILITY};
00478     NV_OnePortSetting port4{"Serial Port 4", nullptr, defaultSerialPortModes[4],
      NB_FACTORY_SERIAL_PORT_4_MODE_CAPABILITY};
00479     NV_OnePortSetting port5{"Serial Port 5", nullptr, defaultSerialPortModes[5],
      NB_FACTORY_SERIAL_PORT_5_MODE_CAPABILITY};
00480     NV_OnePortSetting port6{"Serial Port 6", nullptr, defaultSerialPortModes[6],
      NB_FACTORY_SERIAL_PORT_6_MODE_CAPABILITY};
00481 #elif ( defined SOMRT1061 )
00482     NV_OnePortSetting port1{"Serial Port 1", nullptr, defaultSerialPortModes[1],
      NB_FACTORY_SERIAL_PORT_1_MODE_CAPABILITY};
00483     NV_OnePortSetting port2{"Serial Port 2", nullptr, defaultSerialPortModes[2],
      NB_FACTORY_SERIAL_PORT_2_MODE_CAPABILITY};
00484     NV_OnePortSetting port3{"Serial Port 3", nullptr, defaultSerialPortModes[3],
      NB_FACTORY_SERIAL_PORT_3_MODE_CAPABILITY};
00485     NV_OnePortSetting port4{"Serial Port 4", nullptr, defaultSerialPortModes[4],
      NB_FACTORY_SERIAL_PORT_4_MODE_CAPABILITY};
00486     NV_OnePortSetting port5{"Serial Port 5", nullptr, defaultSerialPortModes[5],
      NB_FACTORY_SERIAL_PORT_5_MODE_CAPABILITY};
00487     NV_OnePortSetting port6{"Serial Port 6", nullptr, defaultSerialPortModes[6],
      NB_FACTORY_SERIAL_PORT_6_MODE_CAPABILITY};
00488 #endif
00489
00490 #ifdef SUPPORTED_GPIO_SERVER_PLATFORM
00491     config_bool bEnableGpioProcess{NB_FACTORY_DO_GPIO_COMMANDS_DEFAULT, "bEnableGpioProcess"};
00492     config_bool bEnableHighDrive{NB_FACTORY_ENABLE_GPIO_HI_DRIVE_DEFAULT, "bEnableHighDrive"};
00493     config_uint GpioProcessPort{NB_FACTORY_GPIO_PORT_DEFAULT, "GpioProcessPort"};
00494     config_uint GpioInactivityTimeoutInSecs{NB_CONNECT_IDLE_TIMEOUT_DEFAULT, "Inactivity timeout in
      seconds"};
00495     config_uint GpioMaxConnections{NB_MAX_TCP_CONNECTIONS, "Max simultaneous connections"};
00496     config_bool bSaveGpioToFlash{
00497         NB_FACTORY_GPIO_SAVE_TO_FLASH_DEFAULT,
00498         "Save every change to a pin's setting/state to flash. This will use more flash write cycles if
      a pin is modified often."};
00499 #endif
00500
00501     SysInfo SysInfoData{"SNMP_SysInfo"};
00502
00503     /* I2C address */
00504     config_string slaveI2CAddress{"", "I2C Slave Address"};   // char slaveI2CAddress[3];
00505
00506 #if (defined SUPPORTED_I2C_PLATFORM)
00507     NV_OneI2CPortSetting i2c_port{"I2C Configuration"};
00508 #endif
00509
00510 #ifdef NB_FACTORY_INCLUDE_SSL
00511
00512     /* SSL certificate and keys file lengths */
00513     config_uint SslCertificateSource{SERIAL_BURNER_LIBRARY_DEFAULT, "SSL Certificate Source"};
00514     config_uint SslCertificateLength{0, "SSL Certificate Length"};
00515     config_uint SslKeyLength{0, "SSL Key Length"};
00516
00517 #endif /* #ifdef NB_FACTORY_INCLUDE_SSL    */
00518
00519 #ifdef NB_FACTORY_INCLUDE_SSH
```

```
00520
00521     /* SSH user name and password */
00522     config_pass SshUserName{"", "SSH User Name"};
00523     config_pass SshPassword{"", "SSH Password"};
00524
00525     /* SSH key source and lengths (default and user installed) */
00526     config_uint SshKeyRsaSource{SERIAL_BURNER_LIBRARY_DEFAULT, "SSH Key RSA Source"};
00527     config_uint SshKeyRsaLength{0, "SSH Key RSA Length"};
00528     config_uint SshKeyEcdsaSource{SERIAL_BURNER_LIBRARY_DEFAULT, "SSH Key ECDSA Source"};
00529     config_uint SshKeyEcdsaLength{0, "SSH Key ECDSA Length"};
00530
00531 #endif /* #ifdef NB_FACTORY_INCLUDE_SSH */
00532
00533     config_bool bAllowAtToAccessAPassword{true, "Allow AT to Access A Password"};
00534     config_bool bAllowAtToAccessSPassword{true, "Allow AT to Access S Password"};
00535
00536     /* Version verification key */
00537     config_uint VerifyKey{NB_FACTORY_VERIFY_KEY, "App Version Verification Key"};
00538
00539     /* Flash File System Version verification key */
00540     config_uint STDEFFSVerifyKey{STD_EFFS_VERIFY_KEY, "File System Version Verification Key"};
00541
00542     config_bool bFirstRun{true, "First time loading the SS2E Factory app on this device"};
00543     ConfigEndMarker;
00544
00545     NV_OnePortSetting *ports[NB_FACTORY_SERIAL_PORTS];
00546
00547     NV_SettingsStruct(const char *name, const char *desc = nullptr, uint32_t configFlag = 0) :
      config_obj(name, desc)
00548     {
00549         if (configFlag > 0) { SetFlag(configFlag); }
00550         // SetFlag(fConfigHidden);     // hide from the config server
00551
00552         ports[0] = &port0;
00553         ports[1] = &port1;
00554
00555 #if (defined NANO54415)
00556         ports[2] = &port2;
00557         ports[3] = &port3;
00558         ports[4] = &port4;
00559 #elif ((defined MODM7AE70 && defined USE_E70_UART_SERIAL_PORTS) || (defined SOMRT1061) )
00560         ports[2] = &port2;
00561         ports[3] = &port3;
00562         ports[4] = &port4;
00563         ports[5] = &port5;
00564         ports[6] = &port6;
00565 #endif
00566
00567         #ifdef SB800EX
00568         // SB800EX serial ports are numbered 1 and 2, not 0 and 1
00569         ports[(int)(monitor_config.Uart) - 1]->SerialMode = SERIAL_MODE_DEBUG;
00570         #else
00571         ports[(int)(monitor_config.Uart)]->SerialMode = SERIAL_MODE_DEBUG;
00572         #endif
00573
00574         for (uint8_t i = 1; i < NB_FACTORY_SERIAL_PORTS; i++)
00575         {
00576             // intentionally skip port 0 since it's initialized with a default value
00577             ports[i]->ListenPort = ports[i]->ListenPort + i;
00578             ports[i]->ConnectPort = ports[i]->ConnectPort + i;
00579         }
00580     };
00581
00582     NV_SettingsStruct(config_obj &owner, const char *name, const char *desc = nullptr, uint32_t
      configFlag = 0)
00583         : config_obj(owner, name, desc)
00584     {
00585         if (configFlag > 0) { SetFlag(configFlag); }
00586         // SetFlag(fConfigHidden);
00587
00588         ports[0] = &port0;
00589         ports[1] = &port1;
00590
00591 #if (defined NANO54415)
00592         ports[2] = &port2;
00593         ports[3] = &port3;
00594         ports[4] = &port4;
00595 #elif ((defined MODM7AE70 && defined USE_E70_UART_SERIAL_PORTS) || (defined SOMRT1061) )
00596         ports[2] = &port2;
00597         ports[3] = &port3;
00598         ports[4] = &port4;
00599         ports[5] = &port5;
00600         ports[6] = &port6;
00601 #endif
00602
00603         #ifdef SB800EX
00604         // SB800EX serial ports are numbered 1 and 2, not 0 and 1
```

```
00605         ports[(int)(monitor_config.Uart) - 1]->SerialMode = SERIAL_MODE_DEBUG;
00606         #else
00607         ports[(int)(monitor_config.Uart)]->SerialMode = SERIAL_MODE_DEBUG;
00608         #endif
00609
00610         for (uint8_t i = 1; i < NB_FACTORY_SERIAL_PORTS; i++)
00611         {
00612             // intentionally skip port 0 since it's initialized with a default value
00613             ports[i]->ListenPort = ports[i]->ListenPort + i;
00614             ports[i]->ConnectPort = ports[i]->ConnectPort + i;
00615         }
00616     };
00617 };
00618
00619 #ifdef SUPPORTED_GPIO_SERVER_PLATFORM
00620 #include <cpu_pins.h>
00621 #include <pins.h>
00622 #include <htmlfiles.h>
00623 class NV_PinIOConfig : public config_obj
00624 {
00625    public:
00626     config_chooser cc_pinFunction{"Function", "Not Configured", "Not Configured,GPIO Input,GPIO Output
      0,GPIO Output 1"};
00627     config_string cs_currentState{"Not Configured", "Current State"};
00628
00629     ConfigEndMarker;
00630
00631     typedef enum
00632     {
00633         PIN_FN_NO_CONFIG = 0,
00634         PIN_FN_IN,
00635         PIN_FN_OUT_0,
00636         PIN_FN_OUT_1,
00637         PIN_FN_A,
00638         PIN_FN_B,
00639         PIN_FN_C,
00640         PIN_FN_D
00641     } pin_fn_t;
00642
00643     uint8_t m_headerNum;
00644     uint8_t m_pinNum;
00645     uint32_t m_configFlags;
00646     PinIO m_pinIO;
00647
00648     NV_PinIOConfig(const char *name,
00649                    PinIO pin,
00650                    const char *desc = nullptr,
00651                    uint8_t headerNumber = 0,
00652                    uint8_t pinNumber = 0,
00653                    const char *pinFuncSelection = nullptr,
00654                    const char *pinFunctionChoices = nullptr,
00655                    uint32_t configFlag = 0)
00656         : m_pinIO(pin), config_obj(name, desc)
00657     {
00658         if (configFlag) { SetFlag(configFlag); }
00659         if (pinFunctionChoices != nullptr) { cc_pinFunction.SetChoices(pinFunctionChoices); }
00660         if (pinFuncSelection != nullptr) { cc_pinFunction = pinFuncSelection; }
00661
00662         m_headerNum = headerNumber;
00663         m_pinNum = pinNumber;
00664         m_configFlags = configFlag;
00665
00666         if (m_pinIO.mask == 0)   // pin is not modifiable. Ex: GND, VCC, or not a valid header pin
00667         {
00668             SetFlag(fConfigReadOnly);
00669             cc_pinFunction.SetChoices("Fixed");
00670             cc_pinFunction = "Fixed";
00671         }
00672     }
00673
00674     NV_PinIOConfig(config_obj &owner,
00675                    const char *name,
00676                    PinIO pin,
00677                    const char *desc = nullptr,
00678                    uint8_t headerNumber = 0,
00679                    uint8_t pinNumber = 0,
00680                    const char *pinFuncSelection = nullptr,
00681                    const char *pinFunctionChoices = nullptr,
00682                    uint32_t configFlag = 0)
00683         : m_pinIO(pin), config_obj(owner, name, desc)
00684     {
00685         if (configFlag > 0) { SetFlag(configFlag); }
00686         if (pinFunctionChoices != nullptr) { cc_pinFunction.SetChoices(pinFunctionChoices); }
00687         if (pinFuncSelection != nullptr) { cc_pinFunction = pinFuncSelection; }
00688
00689         m_headerNum = headerNumber;
00690         m_pinNum = pinNumber;
```

```
00691         m_configFlags = configFlag;
00692
00693         if (m_pinIO.mask == 0)   // pin is not modifiable. Ex: GND, VCC, or not a valid header pin
00694         {
00695             SetFlag(fConfigReadOnly);
00696             cc_pinFunction.SetChoices("Fixed");
00697             cc_pinFunction = "Fixed";
00698         }
00699     }
00700
00701     /* It is assumed that a valid NV_PinIOConfig will have a name. An invalid
00702      * NV_PinIOConfig, like an end marker, will have a nullptr name value */
00703     inline bool valid() { return pName != nullptr; }
00704
00705     bool ConfigureFromPinIOFunction()
00706     {
00707         int8_t pin_fn = m_pinIO.getFn();
00708         if (pin_fn < 0) { return false; }
00709
00710         // check if the pin has already been configured for a function other than GPIO
00711         // if the PinIO is not configured for a peripheral function, allow it to be configured
00712
00713         // if the PinIO is already configured for a peripheral function, what should we do?
00714         // option 1: don't allow it to be configured via the gpio webpage
00715         // option 2: display which peripheral function (A,B,C,D) and allow it to be reconfigured for
    GPIO
00716     }
00717
00718     void DisplayPinFunctionToWeb(int sock)
00719     {
00720         fdprintf(sock, "\r\n\t\t<td><font size=\"2\">");
00721         WriteHtmlVariable(sock, ConfigRenderFunc(2, cc_pinFunction));
00722         fdprintf(sock, "\r\n\t\t</font></td>\r\n");
00723     }
00724
00725     void DisplayPinDriveToWeb(int sock)
00726     {
00727         if (!valid() || m_pinIO.mask == 0)
00728         {
00729             // The pin has not been initialized
00730             cs_currentState = "N/A";
00731         }
00732         else
00733         {
00734             if (cc_pinFunction.IsSelected("GPIO Input"))   // if the pis an input, read and display
    the input
00735             {
00736                 if (m_pinIO.readBack()) { cs_currentState = "GPIO Input = 1"; }
00737                 else
00738                 {
00739                     cs_currentState = "GPIO Input = 0";
00740                 }
00741             }
00742             else if (cc_pinFunction.IsSelected("GPIO Output 0"))   // if the pin is an output, display
    the configured output
00743             {
00744                 cs_currentState = "GPIO Output = 0";
00745             }
00746             else if (cc_pinFunction.IsSelected("GPIO Output 1"))
00747             {
00748                 cs_currentState = "GPIO Output = 1";
00749             }
00750             else if (cc_pinFunction.IsSelected("Pin State"))
00751             {
00752                 cs_currentState = "Not Configured";
00753             }
00754             else
00755             {
00756                 cs_currentState = "Not Configured";
00757             }
00758         }
00759
00760         fdprintf(sock, "\r\n\t\t<td><font size=\"2\">");
00761         WriteHtmlVariable(sock, ConfigRenderFunc(1, cs_currentState));
00762         fdprintf(sock, "\r\n\t\t</font></td>\r\n");
00763     }
00764
00765     void DisplayFunction(int sock)
00766     {
00767         static char rowLine[sizeof("\r\n\t\t<td><font size=\"2\">")];
00768         writestring(sock, rowLine);
00769         WriteHtmlVariable(sock, ConfigRenderFunc(2, cc_pinFunction));
00770         writestring(sock, "</td>");
00771     }
00772
00773     // void DisplayCurrentState(int sock)
00774     // {
```

```
00775     //    if( !valid() )
00776     //    {
00777     //       // The pin has not been initialized
00778     //       cs_currentState = "Not Configured";
00779     //    }
00780     //    else
00781     //    {
00782     //       if( m_pinIO.readBack() )  { cs_currentState = "1"; }
00783     //       else                      { cs_currentState = "0"; }
00784     //    }
00785
00786     //    static char rowLine[ sizeof( "\r\n\t\t<td><font size=\"2\">") ];
00787     //    writestring( sock, rowLine  );
00788     //    WriteHtmlVariable(sock,ConfigRenderFunc(1, cs_currentState));
00789     //    writestring( sock, "</td>"  );
00790     // }
00791
00792     uint8_t GetPinFunctionIndex()
00793     {
00794         NBString pinFuncs(cc_pinFunction.GetChoices());
00795
00796         // find the position of the cc_pinFunction value within the choices to find the index
00797         size_t position = pinFuncs.find(((NBString)cc_pinFunction).c_str());
00798
00799         uint8_t commaCount = 0;
00800         for (size_t start = position; start > 0; start--)
00801         {
00802             if (pinFuncs[start] == ',') { commaCount++; }
00803         }
00804
00805         return commaCount;
00806     }
00807
00808     // returns < 0 if failed or the NV_PinIOConfig::pin_fn_t configuration that the pin was
      initialized to.
00809     uint8_t InitPinFunction()
00810     {
00811         if (!valid() || m_pinIO.mask == 0) { return -1; }
00812         // Don't initialize pins that are marked as Read Only
00813         if (GetFlags() & fConfigReadOnly) { return -2; }
00814
00815         uint8_t cc_pinFunctionVal = GetPinFunctionIndex();
00816
00817         switch (cc_pinFunctionVal)
00818         {
00819             case NV_PinIOConfig::PIN_FN_NO_CONFIG: m_pinIO.function(PinIO::PIN_FN_IN); return
      NV_PinIOConfig::PIN_FN_IN;
00820
00821             case NV_PinIOConfig::PIN_FN_IN: m_pinIO.function(PinIO::PIN_FN_IN); return
      NV_PinIOConfig::PIN_FN_IN;
00822
00823             case NV_PinIOConfig::NV_PinIOConfig::PIN_FN_OUT_0:
00824                 m_pinIO.function(PinIO::PIN_FN_OUT);
00825                 m_pinIO.clr();
00826                 return NV_PinIOConfig::NV_PinIOConfig::PIN_FN_OUT_0;
00827
00828             case NV_PinIOConfig::PIN_FN_OUT_1:
00829                 m_pinIO.function(PinIO::PIN_FN_OUT);
00830                 m_pinIO.set();
00831                 return NV_PinIOConfig::PIN_FN_OUT_1;
00832
00833             case NV_PinIOConfig::PIN_FN_A: m_pinIO.function(PinIO::PIN_FN_A); return
      NV_PinIOConfig::PIN_FN_A;
00834
00835             case NV_PinIOConfig::PIN_FN_B: m_pinIO.function(PinIO::PIN_FN_B); return
      NV_PinIOConfig::PIN_FN_B;
00836
00837             case NV_PinIOConfig::PIN_FN_C: m_pinIO.function(PinIO::PIN_FN_C); return
      NV_PinIOConfig::PIN_FN_C;
00838
00839             case NV_PinIOConfig::PIN_FN_D: m_pinIO.function(PinIO::PIN_FN_D); return
      NV_PinIOConfig::PIN_FN_D;
00840
00841             default: return -3;
00842         }
00843     }
00844 };
00845
00846 #ifdef SBE70LC
00847 #define PIN_HEADER_SIZE 20   // J1 Header
00848 #else
00849 #define PIN_HEADER_SIZE 50
00850 #endif
00851
00852 extern NV_PinIOConfig pinHeaderArray[];
00853 bool InitPinArrayFunctions();
00854 bool SetGpioConfigToReadOnly(bool bSetToReadOnly);
```

```
00855 bool SetGpioHighCurrentDrive(bool bEnableHighDrive);
00856 #endif   // end SUPPORTED_GPIO_SERVER_PLATFORM
00857
00858 /*
00859  ****************************************************************************
00860  *
00861  * Global Data Declarations
00862  *
00863  ****************************************************************************
00864  */
00865 /* DHCP object */
00866 // extern DhcpObject* gDHCPObjPtr;
00867
00868 /* Settings changed flag */
00869 extern volatile bool Settings_Changed;
00870 extern volatile int old_config_ver;
00871
00872 /* User parameters */
00873 extern NV_SettingsStruct NV_Settings;
00874
00875 /* User parameters change flag */
00876 extern volatile bool gChangedUserParameters;
00877
00878 /* Debugging flag */
00879 extern bool bShowDebug;
00880
00881 /*
00882  * Selection lists for the web page configuration
00883  */
00884 extern const char *IPModeList[];
00885 extern const char *IPValueList[];
00886 extern const char *IPOnClickList[];
00887 extern const char *BaudList[];
00888 extern const char *BitList[];
00889 extern const char *ParityList[];
00890 extern const char *StopList[];
00891 extern const char *FlowModeList[];
00892 extern const char *SerialModeListFull[];
00893 extern const char *SerialModeListRS232[];
00894 extern const char *SerialModeListRS485[];
00895 extern const char *SerialModeListHybrid[];
00896 extern const char *SerialModeListQuad485[];
00897 extern const char *SerialModeListQuadMMS[];
00898 extern const char *ConnectMode[];
00899 extern const char *ProtocolList[];
00900 /*
00901  ****************************************************************************
00902  *
00903  * Routines
00904  *
00905  ****************************************************************************
00906  */
00907 /* Save data to file */
00908 extern bool UserSaveData(char *dataPtr, int dataSize, const char *fileName);
00909
00910 /* Get saved data */
00911 extern bool UserGetData(char *dataPtr, char *fileName, int dataSize);
00912
00913 extern void RegisterPost(void);
00914
00915 extern void CheckNVSettings(bool returnToFactory = FALSE);
00916
00917 extern void SetAndSaveDefaults(void);
00918
00919 /* For processing hexadecimal break key value */
00920 extern uint8_t GetHexByte(const char *cp);
00921
00922 #endif /* _SERIALBURNERDATA_H_ */
```

24.218 SSL/HttpsUploadCert/src/serialburnerdata.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SERIALBURNERDATA_H_
00006 #define _SERIALBURNERDATA_H_
00007
00008 #include <config_obj.h>
00009 extern MonitorRecord monitor_config;
00010
00011 #define DEVICE_NAME_LENGTH (15)
00012
00013 /*
00014  ****************************************************************************
```

```
00015  * Key or certificate size
00016  *       SSL Certificate size 2200 (empirical)
00017  *       OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00018  *       NULL terminated for conversion
00019  *
00020  ******************************************************************************
00021  */
00022 #define SERIAL_BURNER_CERTIFICATE_SIZE_MAX ((2 * 1024) - 1)
00023 #define SERIAL_BURNER_CERTIFICATE_SIZE_MAX_PEM ((3 * 1024) - 1)
00024
00025 /* Certificate and key status */
00026 #define SERIAL_BURNER_LIBRARY_DEFAULT ((uint8_t)0x00)
00027 #define SERIAL_BURNER_DEFAULT ((uint8_t)0x01)
00028 #define SERIAL_BURNER_USER_INSTALLED ((uint8_t)0x02)
00029
00030 /*
00031  ******************************************************************************
00032  * Key size
00033  *       SSL key size (PEM) < 4K (empirical)
00034  *       NULL terminated for conversion
00035  *
00036  ******************************************************************************
00037  */
00038 #define SERIAL_BURNER_KEY_SIZE_MAX_PEM ((4 * 1024) - 1)
00039
00040 /*
00041  ******************************************************************************
00042  * Booting support
00043  ******************************************************************************
00044  */
00045 #define boot_iprintf(...)                               \
00046     {                                                   \
00047         if (monitor_config.Quiet == 0)                  \
00048         {                                               \
00049             iprintf("%s : ", NV_Settings.DeviceName); \
00050             iprintf(__VA_ARGS__);                       \
00051             iprintf("\r\n");                            \
00052         }                                               \
00053     }
00054
00055 /*
00056  ******************************************************************************
00057  * Debug support
00058  ******************************************************************************
00059  */
00060 #define debug_iprintf(...)                              \
00061     {                                                   \
00062         if (bShowDebug == TRUE)                         \
00063         {                                               \
00064             iprintf("%s : ", NV_Settings.DeviceName); \
00065             iprintf(__VA_ARGS__);                       \
00066             iprintf("\r\n");                            \
00067         }                                               \
00068     }
00069
00070 /*
00071  ******************************************************************************
00072  *
00073  * Structures
00074  *
00075  ******************************************************************************
00076  */
00077
00078 /*
00079       Configuration Settings
00080
00081          DeviceName              - Device name for DHCP
00082          NetBIOSName             - NetBIOS name
00083
00084          * SSL *
00085          CertificateRsaLength    - Certificate length
00086          CertificateData         - Certificate
00087          KeyHttpsRsaLength       - RSA key for HTTPS length, 0 is none
00088          KeyHttpsRsaData         - RSA key for HTTPS
00089          KeyRsaLength            - RSA key length, 0 is none
00090          KeyRsaData              - RSA key
00091          KeyDsaLength            - DSA key length, 0 is none
00092          KeyDsaData              - DSA key
00093
00094          * Version change key *
00095          VerifyKey               - Version change key
00096
00097 */
00098 struct NV_SettingsStruct
00099 {
00100     /* NetBurner address configuration */
00101     char DeviceName[(DEVICE_NAME_LENGTH + 1)];
```

```
00102     char NetBIOSName[(NETBIOS_NAME_SIZE_IN_CHARS + 1)];
00103
00104     /* SSL certificate and keys file lengths */
00105     uint8_t SslCertificateSource;
00106     uint16_t SslCertificateLength;
00107     uint16_t SslKeyLength;
00108
00109     /* Version verification key */
00110     uint32_t VerifyKey;
00111     /* Flash File System Version verification key */
00112     uint32_t STDEFFSVerifyKey;
00113 };
00114
00115 /*
00116  ******************************************************************************
00117  *
00118  * Global Data Declarations
00119  *
00120  ******************************************************************************
00121  */
00122
00123 /* User parameters */
00124 extern NV_SettingsStruct NV_Settings;
00125
00126 /* User parameters change candidate */
00127 extern NV_SettingsStruct gNV_SettingsChangeCopy;
00128
00129 /* User parameters change flag */
00130 extern volatile BOOL gChangedUserParameters;
00131
00132 /* Debugging flag */
00133 extern BOOL bShowDebug;
00134
00135 /*
00136  ******************************************************************************
00137  *
00138  * Routines
00139  *
00140  ******************************************************************************
00141  */
00142
00143 extern void RegisterPost(void);
00144
00145 /* Save data to file */
00146 extern BOOL UserSaveData(char *dataPtr, int dataSize, const char *fileName);
00147
00148 /* Get saved data */
00149 extern BOOL UserGetData(char *dataPtr, char *fileName, int dataSize);
00150
00151 extern void CheckNVSettings(BOOL returnToFactory = FALSE);
00152
00153 extern void SetAndSaveDefaults(void);
00154
00155 /* For processing hexadecimal break key value */
00156 extern char GetHexByte(const char *cp);
00157
00158 #endif /* _SERIALBURNERDATA_H_ */
```

24.219 SSL/SslClientVerifyPeerEffs/src/serialburnerdata.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SERIALBURNERDATA_H_
00006 #define _SERIALBURNERDATA_H_
00007
00008 #include <config_obj.h>
00009 #include <netbios.h>
00010
00011 extern MonitorRecord monitor_config;
00012
00013 #define DEVICE_NAME_LENGTH (15)
00014
00021 #define SERIAL_BURNER_CERTIFICATE_SIZE_MAX ((5 * 1024) - 1)
00022 #define SERIAL_BURNER_CERTIFICATE_SIZE_MAX_PEM ((5 * 1024) - 1)
00023
00024 /* Certificate and key status */
00025 #define SERIAL_BURNER_LIBRARY_DEFAULT ((uint8_t)0x00)
00026 #define SERIAL_BURNER_DEFAULT ((uint8_t)0x01)
00027 #define SERIAL_BURNER_USER_INSTALLED ((uint8_t)0x02)
00028
00035 #define SERIAL_BURNER_KEY_SIZE_MAX_PEM ((4 * 1024) - 1)
00036
00040 #define boot_iprintf(...)                              \
```

```
00041     {                                                   \
00042         if (monitor_config.Quiet == 0)                  \
00043         {                                               \
00044             iprintf("%s : ", NV_Settings.DeviceName); \
00045             iprintf(__VA_ARGS__);                       \
00046             iprintf("\r\n");                            \
00047         }                                               \
00048     }
00049
00053 #define debug_iprintf(...)                              \
00054     {                                                   \
00055         if (bShowDebug == true)                         \
00056         {                                               \
00057             iprintf("%s : ", NV_Settings.DeviceName); \
00058             iprintf(__VA_ARGS__);                       \
00059             iprintf("\r\n");                            \
00060         }                                               \
00061     }
00062
00082 struct NV_SettingsStruct
00083 {
00084     /* NetBurner address configuration */
00085     char DeviceName[(DEVICE_NAME_LENGTH + 1)];
00086     char NetBIOSName[(NETBIOS_NAME_SIZE_IN_CHARS + 1)];
00087
00088     /* SSL certificate and keys file lengths */
00089     uint8_t SslCertificateSource;
00090     uint16_t SslCertificateLength;
00091     uint16_t SslKeyLength;
00092
00093     /* Version verification key */
00094     uint32_t VerifyKey;
00095     /* Flash File System Version verification key */
00096     uint32_t STDEFFSVerifyKey;
00097 };
00098
00099 /* User parameters */
00100 extern NV_SettingsStruct NV_Settings;
00101
00102 /* User parameters change candidate */
00103 extern NV_SettingsStruct gNV_SettingsChangeCopy;
00104
00105 /* User parameters change flag */
00106 extern volatile bool gChangedUserParameters;
00107
00108 /* Debugging flag */
00109 extern bool bShowDebug;
00110
00111 /* Register a post */
00112 extern void RegisterPost(void);
00113
00114 extern void CheckNVSettings(bool returnToFactory = false);
00115
00116 extern void SetAndSaveDefaults(void);
00117
00118 /* For processing hexadecimal break key value */
00119 extern char GetHexByte(const char *cp);
00120
00121 #endif /* _SERIALBURNERDATA_H_ */
```

24.220 serialportinfo.h

```
00001 #ifndef SERIAL_PORT_INFO
00002 #define SERIAL_PORT_INFO
00003
00004 #ifdef MOD5441X
00005 const char *serial_port_pin_info = R"(
00006     <table border="1" cellspacing="0" cellpadding="4">
00007         <tr>
00008           <td width="127" valign="top"><b>Serial Port</b></td>
00009           <td width="127" valign="top"><b>Rx Pin</b></td>
00010           <td width="127" valign="top"><b>Tx Pin</b></td>
00011           <td width="127" valign="top"><b>CTS Pin</b></td>
00012           <td width="127" valign="top"><b>RTS Pin</b></td>
00013           <td width="127" valign="top"><b>Comment</b></td>
00014         </tr>
00015         <tr>
00016           <td width="127" valign="top">0</td>
00017           <td width="127" valign="top">J2[3]</td>
00018           <td width="127" valign="top">J2[4]</td>
00019           <td width="127" valign="top">J2[29]</td>
00020           <td width="127" valign="top">J2[38]</td>
00021           <td width="127" valign="top"></td>
00022         </tr>
00023         <tr>
```

```
00024           <td width="127" valign="top">1</td>
00025           <td width="127" valign="top">J2[21]</td>
00026           <td width="127" valign="top">J2[22]</td>
00027           <td width="127" valign="top">J2[24]</td>
00028           <td width="127" valign="top">J2[23]</td>
00029           <td width="127" valign="top"></td>
00030         </tr>
00031         <tr>
00032           <td width="127" valign="top">2</td>
00033           <td width="127" valign="top">J2[31]</td>
00034           <td width="127" valign="top">J2[19]</td>
00035           <td width="127" valign="top">-</td>
00036           <td width="127" valign="top">-</td>
00037           <td width="127" valign="top">Hardware flow control not supported</td>
00038         </tr>
00039         <tr>
00040           <td width="127" valign="top">3</td>
00041           <td width="127" valign="top">-</td>
00042           <td width="127" valign="top">-</td>
00043           <td width="127" valign="top">-</td>
00044           <td width="127" valign="top">-</td>
00045           <td width="127" valign="top">UART not available through the header pins</td>
00046         </tr>
00047         <tr>
00048           <td width="127" valign="top">4</td>
00049           <td width="127" valign="top">J2[38]</td>
00050           <td width="127" valign="top">J2[29]</td>
00051           <td width="127" valign="top">-</td>
00052           <td width="127" valign="top">-</td>
00053           <td width="127" valign="top">Hardware flow control not supported</td>
00054         </tr>
00055         <tr>
00056           <td width="127" valign="top">5</td>
00057           <td width="127" valign="top">J2[23]</td>
00058           <td width="127" valign="top">J2[24]</td>
00059           <td width="127" valign="top">-</td>
00060           <td width="127" valign="top">-</td>
00061           <td width="127" valign="top">Hardware flow control not supported</td>
00062         </tr>
00063         <tr>
00064           <td width="127" valign="top">6</td>
00065           <td width="127" valign="top">-</td>
00066           <td width="127" valign="top">-</td>
00067           <td width="127" valign="top">-</td>
00068           <td width="127" valign="top">-</td>
00069           <td width="127" valign="top">UART not available through the header pins</td>
00070         </tr>
00071         <tr>
00072           <td width="127" valign="top">7</td>
00073           <td width="127" valign="top">J2[16]</td>
00074           <td width="127" valign="top">J2[20]</td>
00075           <td width="127" valign="top">-</td>
00076           <td width="127" valign="top">-</td>
00077           <td width="127" valign="top">Hardware flow control not supported</td>
00078         </tr>
00079         <tr>
00080           <td width="127" valign="top">8</td>
00081           <td width="127" valign="top">J2[39]</td>
00082           <td width="127" valign="top">J2[42]</td>
00083           <td width="127" valign="top">-</td>
00084           <td width="127" valign="top">-</td>
00085           <td width="127" valign="top">Hardware flow control not supported</td>
00086         </tr>
00087         <tr>
00088           <td width="127" valign="top">9</td>
00089           <td width="127" valign="top">J2[41]</td>
00090           <td width="127" valign="top">J2[44]</td>
00091           <td width="127" valign="top">-</td>
00092           <td width="127" valign="top">-</td>
00093           <td width="127" valign="top">Hardware flow control not supported</td>
00094         </tr>
00095     </table>
00096 )";
00097 #elif (defined NANO54415)
00098 const char *serial_port_pin_info = R"(
00099     <table border="1" cellspacing="0" cellpadding="4">
00100         <tr>
00101           <td width="127" valign="top"><b>Serial Port</b></td>
00102           <td width="127" valign="top"><b>Rx Pin</b></td>
00103           <td width="127" valign="top"><b>Tx Pin</b></td>
00104           <td width="127" valign="top"><b>CTS Pin</b></td>
00105           <td width="127" valign="top"><b>RTS Pin</b></td>
00106           <td width="127" valign="top"><b>Comment</b></td>
00107         </tr>
00108         <tr>
00109           <td width="127" valign="top">0</td>
00110           <td width="127" valign="top">P1[24]</td>
```

```
00111           <td width="127" valign="top">P1[26]</td>
00112           <td width="127" valign="top">P1[30]</td>
00113           <td width="127" valign="top">P1[28]</td>
00114           <td width="127" valign="top"></td>
00115         </tr>
00116         <tr>
00117           <td width="127" valign="top">1</td>
00118           <td width="127" valign="top">P1[32]</td>
00119           <td width="127" valign="top">P1[34]</td>
00120           <td width="127" valign="top">P1[38]</td>
00121           <td width="127" valign="top">P1[36]</td>
00122           <td width="127" valign="top"></td>
00123         </tr>
00124         <tr>
00125           <td width="127" valign="top">2</td>
00126           <td width="127" valign="top">P1[13]</td>
00127           <td width="127" valign="top">P1[16]</td>
00128           <td width="127" valign="top">P1[10]</td>
00129           <td width="127" valign="top">P1[14]</td>
00130           <td width="127" valign="top"></td>
00131         </tr>
00132         <tr>
00133           <td width="127" valign="top">3</td>
00134           <td width="127" valign="top">P1[29]</td>
00135           <td width="127" valign="top">P1[27]</td>
00136           <td width="127" valign="top">-</td>
00137           <td width="127" valign="top">-</td>
00138           <td width="127" valign="top">Hardware flow control not supported</td>
00139         </tr>
00140         <tr>
00141           <td width="127" valign="top">4</td>
00142           <td width="127" valign="top">P1[20]</td>
00143           <td width="127" valign="top">P1[22]</td>
00144           <td width="127" valign="top">-</td>
00145           <td width="127" valign="top">-</td>
00146           <td width="127" valign="top">Hardware flow control not supported</td>
00147         </tr>
00148     </table>
00149 )";
00150 #elif (defined MODM7AE70)
00151 const char *serial_port_pin_info = R"(
00152     <table border="1" cellspacing="0" cellpadding="4">
00153         <tr>
00154           <td width="127" valign="top"><b>Serial Port</b></td>
00155           <td width="127" valign="top"><b>Rx Pin</b></td>
00156           <td width="127" valign="top"><b>Tx Pin</b></td>
00157           <td width="127" valign="top"><b>CTS Pin</b></td>
00158           <td width="127" valign="top"><b>RTS Pin</b></td>
00159           <td width="127" valign="top"><b>Comment</b></td>
00160         </tr>
00161         <tr>
00162           <td width="127" valign="top">0</td>
00163           <td width="127" valign="top">P2[3]</td>
00164           <td width="127" valign="top">P2[4]</td>
00165           <td width="127" valign="top">P2[29]</td>
00166           <td width="127" valign="top">P2[38]</td>
00167           <td width="127" valign="top"></td>
00168         </tr>
00169         <tr>
00170           <td width="127" valign="top">1</td>
00171           <td width="127" valign="top">P2[21]</td>
00172           <td width="127" valign="top">P2[22]</td>
00173           <td width="127" valign="top">P2[33]</td>
00174           <td width="127" valign="top">P2[32]</td>
00175           <td width="127" valign="top"></td>
00176         </tr>
00177         <tr>
00178           <td width="127" valign="top">2</td>
00179           <td width="127" valign="top">P2[34]</td>
00180           <td width="127" valign="top">P2[35]</td>
00181           <td width="127" valign="top">-</td>
00182           <td width="127" valign="top">-</td>
00183           <td width="127" valign="top">Hardware flow control not supported</td>
00184         </tr>
00185         <tr>
00186           <td width="127" valign="top">3</td>
00187           <td width="127" valign="top">P2[12]</td>
00188           <td width="127" valign="top">P1[31]</td>
00189           <td width="127" valign="top">-</td>
00190           <td width="127" valign="top">-</td>
00191           <td width="127" valign="top">Hardware flow control not supported</td>
00192         </tr>
00193         <tr>
00194           <td width="127" valign="top">4</td>
00195           <td width="127" valign="top">P2[41]</td>
00196           <td width="127" valign="top">P2[44]</td>
00197           <td width="127" valign="top">-</td>
```

```
00198           <td width="127" valign="top">-</td>
00199           <td width="127" valign="top">Hardware flow control not supported</td>
00200         </tr>
00201         <tr>
00202           <td width="127" valign="top">5</td>
00203           <td width="127" valign="top">P2[23]</td>
00204           <td width="127" valign="top">P2[7]</td>
00205           <td width="127" valign="top">-</td>
00206           <td width="127" valign="top">-</td>
00207           <td width="127" valign="top">Hardware flow control not supported</td>
00208         </tr>
00209         <tr>
00210           <td width="127" valign="top">6</td>
00211           <td width="127" valign="top">P2[10]</td>
00212           <td width="127" valign="top">P1[7]</td>
00213           <td width="127" valign="top">-</td>
00214           <td width="127" valign="top">-</td>
00215           <td width="127" valign="top">Hardware flow control not supported</td>
00216         </tr>
00217     </table>
00218 )";
00219 #elif (defined SBE70LC)
00220 const char *serial_port_pin_info = R"(
00221     <table border="1" cellspacing="0" cellpadding="4">
00222         <tr>
00223           <td width="127" valign="top"><b>Serial Port</b></td>
00224           <td width="127" valign="top"><b>Rx Pin</b></td>
00225           <td width="127" valign="top"><b>Tx Pin</b></td>
00226           <td width="127" valign="top"><b>CTS Pin</b></td>
00227           <td width="127" valign="top"><b>RTS Pin</b></td>
00228           <td width="127" valign="top"><b>Comment</b></td>
00229         </tr>
00230         <tr>
00231           <td width="127" valign="top">0</td>
00232           <td width="127" valign="top">J1[11]</td>
00233           <td width="127" valign="top">J1[10]</td>
00234           <td width="127" valign="top">J1[14]</td>
00235           <td width="127" valign="top">J1[5]</td>
00236           <td width="127" valign="top"></td>
00237         </tr>
00238         <tr>
00239           <td width="127" valign="top">1</td>
00240           <td width="127" valign="top">J1[13]</td>
00241           <td width="127" valign="top">J1[12]</td>
00242           <td width="127" valign="top">J1[8]</td>
00243           <td width="127" valign="top">J1[9]</td>
00244           <td width="127" valign="top"></td>
00245         </tr>
00246     </table>
00247 )";
00248 #elif (defined SOMRT1061)
00249 const char *serial_port_pin_info = R"(
00250     <table border="1" cellspacing="0" cellpadding="4">
00251         <tr>
00252           <td width="127" valign="top"><b>Serial Port</b></td>
00253           <td width="127" valign="top"><b>Rx Pin</b></td>
00254           <td width="127" valign="top"><b>Tx Pin</b></td>
00255           <td width="127" valign="top"><b>CTS Pin</b></td>
00256           <td width="127" valign="top"><b>RTS Pin</b></td>
00257           <td width="127" valign="top"><b>Comment</b></td>
00258         </tr>
00259         <tr>
00260           <td width="127" valign="top">0</td>
00261           <td width="127" valign="top">P1[7]</td>
00262           <td width="127" valign="top">P1[8]</td>
00263           <td width="127" valign="top">P1[6]</td>
00264           <td width="127" valign="top">P1[4]</td>
00265           <td width="127" valign="top"></td>
00266         </tr>
00267         <tr>
00268           <td width="127" valign="top">1</td>
00269           <td width="127" valign="top">P1[43]</td>
00270           <td width="127" valign="top">P1[42]</td>
00271           <td width="127" valign="top">P1[40]</td>
00272           <td width="127" valign="top">P1[41]</td>
00273           <td width="127" valign="top"></td>
00274         </tr>
00275         <tr>
00276           <td width="127" valign="top">2</td>
00277           <td width="127" valign="top">P1[88]</td>
00278           <td width="127" valign="top">P1[87]</td>
00279           <td width="127" valign="top">P1[3]</td>
00280           <td width="127" valign="top">P1[2]</td>
00281           <td width="127" valign="top"></td>
00282         </tr>
00283         <tr>
00284           <td width="127" valign="top">3</td>
```

```
00285           <td width="127" valign="top">P1[85]</td>
00286           <td width="127" valign="top">P1[86]</td>
00287           <td width="127" valign="top">P1[81]</td>
00288           <td width="127" valign="top">P1[82]</td>
00289           <td width="127" valign="top"></td>
00290         </tr>
00291         <tr>
00292           <td width="127" valign="top">4</td>
00293           <td width="127" valign="top">P1[83]</td>
00294           <td width="127" valign="top">P1[84]</td>
00295           <td width="127" valign="top">P1[78]</td>
00296           <td width="127" valign="top">P1[80]</td>
00297           <td width="127" valign="top"></td>
00298         </tr>
00299         <tr>
00300           <td width="127" valign="top">5</td>
00301           <td width="127" valign="top">P1[76]</td>
00302           <td width="127" valign="top">P1[77]</td>
00303           <td width="127" valign="top">-</td>
00304           <td width="127" valign="top">-</td>
00305           <td width="127" valign="top">Hardware flow control not supported</td>
00306         </tr>
00307         <tr>
00308           <td width="127" valign="top">6</td>
00309           <td width="127" valign="top">P1[28]</td>
00310           <td width="127" valign="top">P1[29]</td>
00311           <td width="127" valign="top">P1[25]</td>
00312           <td width="127" valign="top">P1[24]</td>
00313           <td width="127" valign="top"></td>
00314         </tr>
00315     </table>
00316 )";
00317 #endif
00318
00319 #endif // SERIAL_PORT_INFO
```

24.221 serialrecord.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SERIAL_RECORD_H_
00006 #define _SERIAL_RECORD_H_
00007 #pragma once
00008
00009 #include <basictypes.h>
00010 #include <buffers.h>
00011 #include <iosys.h>
00012 #include <nettypes.h>
00013 #include <system.h>
00014
00015 #include "nbfactory.h"
00016 #include "serialburnerdata.h"
00017
00022 struct SerialRecord
00023 {
00024     /*
00025      * Data
00026      */
00027     // File descriptor of serial port
00028     int FD_SerialPort;
00029
00030     // File descriptor of listening socket
00031     int FD_ListeningSocket;
00032
00033     // File descriptor of connected socket
00034     int FD_ConnectedSocket;
00035
00036     // TRUE if we established the connection
00037     bool bWeInitiatedConnection;
00038
00039     // Time of last network sourced data in seconds since last boot
00040     uint32_t LastNetWorkDataRxed;
00041
00042     // Time of last transmitted data in seconds since last boot
00043     uint32_t LastNetWorkDataTxed;
00044
00045     // Time of last attempted outgoing connection in seconds since last boot
00046     uint32_t LastConnectTry;
00047
00048     bool bSerialBoundDataBlocked;
00049     bool bNetWorkBoundDataBlocked;
00050
00051     // Circular buffer from serial port to network
```

```
00052     char Buffer_From_S2N[BUFFER_SIZE];
00053     int Buf_S2N_Start;
00054     int Buf_S2N_End;
00055
00056     // Circular buffer from network to serial port
00057     char Buffer_From_N2S[BUFFER_SIZE];
00058     int Buf_N2S_Start;
00059     int Buf_N2S_End;
00060
00061     IPADDR LearnedUdp;
00062     uint32_t RxBufferCount;
00063     int port_index;    // A global PortRecord array will be declared
00064     int uart_num;      // Physical UART number
00065     int conn_try;
00066
00067     // Flag verification and tick time-stamping for keep-alive implementation
00068     bool tcpKeepAliveSent;
00069     uint32_t tcpLastRxTicks;
00070     uint32_t tcpkeepAliveTicks;
00071
00075     void AssignUartNumber(void);
00076
00077     void ProcessUdpSerialRead(void);
00078     void ProcessUdpRead(void);
00079     void ProcessUdpTxTo(void);
00080     void SetUdpReadFD(fd_set &fd_rd);
00081
00082     void SetTcpFDs(fd_set &fd_rd, fd_set &fd_wr, fd_set &fd_err);
00083     void ProcessTcpFDs(fd_set &fd_rd, fd_set &fd_wr, fd_set &fd_err);
00084     void ProcessTCPReadSerialData(void);
00085     void ProcessSpecialFrameTCPReadSerialData(void);
00086
00087     void ProcessSpecialFrameWriteNetworkData(void);
00088     void ProcessSpecialFrameWriteTimeout(void);
00089     void MakeTcpConnection(void);
00090
00091 #ifdef SB800EX
00092     void TestDSR(void);
00093     void TestCD(void);
00094 #endif   // SB800EX
00095
00096     void SendSerialMessage(const char *msg);
00097     int SerialBreakWrite(char *start, int len);
00098     void CloseListenPort(void);
00099     void MakeUdpConnection(void);
00100     bool OkToListen(void);
00101     void ProcessTimeouts(void);
00102     void ProcessAccept(void);
00103     void OpenSerialPort(void);
00104     void DisableSerialPort(void);
00105     void ProcessReadNetworkData(void);
00106     void OpenListenPort(void);
00107     void ProcessWriteNetworkData(void);
00108     void ProcessWriteSerialData(void);
00109
00110     void ProcessSerialError(void);
00111     void ProcessListenError(void);
00112     void ProcessNetworkError(void);
00113     void EnableATCommands();
00114     void DisableATCommands();
00115     void GetCurrentChannelStatus(char *buffer);
00116 };
00117
00118 #endif
```

24.222 SecureSerToEthFactoryApp/src/sshuser.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SSHUSER_H_
00006 #define _SSHUSER_H_
00007
00008 #include "serialburnerdata.h"
00009 /*
00010  ****************************************************************************
00011  ****************************************************************************
00012  *
00013  * OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00014  *
00015  ****************************************************************************
00016  ****************************************************************************
00017  */
00018
```

```
00019 /*
00020  ***************************************************************************
00021  *
00022  * Global data definitions (declared in sshuser.cpp)
00023  *
00024  ***************************************************************************
00025  */
00026
00027 /* SSH keys PEM encoded (sshuser.cpp) */
00028 extern char *gSshRsaKeyPemEncoded[(SERIAL_BURNER_KEY_SIZE_MAX_PEM + 1)];
00029 extern char *gSshEcdsaKeyPemEncoded[(SERIAL_BURNER_KEY_SIZE_MAX_PEM + 1)];
00030
00031 /*
00032  ***************************************************************************
00033  *
00034  * Functions
00035  *
00036  ***************************************************************************
00037  */
00038 /*
00039  ***************************************************************************
00040  *
00041  * "C" Routines
00042  *
00043  ***************************************************************************
00044  */
00045 #ifdef __cplusplus
00046 extern "C"
00047 {
00048 #endif
00049
00050     /*
00051      ***************************************************************************
00052
00053        User provided SSH username and password authenticate routine.
00054
00055        Parameters:
00056           usernamePtr          - Username in plain text
00057           passwordPtr          - Password in plain text
00058
00059        Return:
00060           1 - Authenticated, all else error
00061
00062        Notes:
00063           None
00064
00065      ***************************************************************************
00066      */
00067     int SshUserAuthenticate(const char *usernamePtr, const char *passwordPtr, AuthType authType);
00068
00069     /*
00070      ***************************************************************************
00071
00072        User provided SSH key retrieval
00073
00074        Parameters:
00075           keyRequested         - Type key requested
00076                                      SSH_KEY_RSA
00077                                      SSH_KEY_ECC
00078           keyBufferPtr         - Key from user storage
00079           keyLengthPtr         - Size of key in 8 bit bytes
00080
00081        Return:
00082           0 - key and length is valid, -1 - key requested not available
00083
00084        Notes:
00085           openSS(L|H) key pair, PEM encoded, no encrypted or with passphrase.
00086           Key must be valid. Each type asked for once at at startup.
00087           The buffer containing the key will NOT be deallocated.
00088           Server will disable task scheduling calling OSLock, copy contents, then
00089           call OSUnlock
00090
00091      ***************************************************************************
00092      */
00093     int SshUserGetKey(int keyRequested, const unsigned char **keyBufferPtr, int *keyLengthPtr);
00094
00095     /*
00096      ***************************************************************************
00097
00098        Verifies SSH key
00099
00100        Parameters:
00101           pemKeyPtr            - PEM encoded key data
00102           pemKeySize           - PEM encoded key size in bytes
00103           keyTypePtr           - Pointer for key type
00104
00105        Return:
```

```
00106           TRUE - OK, FALSE invalid.
00107
00108        Notes:
00109           None
00110
00111      ************************************************************************
00112      */
00113     bool SshUserVerifyKey(char *pemKeyPtr, int pemKeySize, int *keyTypePtr);
00114
00115     /*
00116      ************************************************************************
00117
00118        Checks and installs SSH keys permanent defaults
00119
00120        Parameters:
00121           None
00122
00123        Return:
00124           None
00125
00126        Notes:
00127           Sets NV_Settings elements:
00128              SshKeyRsaSource;
00129              SshKeyRsaLength;
00130              SshKeyEcdsaSource;
00131              SshKeyEcdsaLength;
00132
00133      ************************************************************************
00134      */
00135     void SshUserSetDefault(void);
00136
00137     /*
00138      ************************************************************************
00139
00140        Retrieves and set keys
00141
00142        Parameters:
00143           None
00144
00145        Return:
00146           None
00147
00148        Notes:
00149           Clears SSH settings for CertificateNKeysDataStatus element of
00150                 struct NV_SettingsStruct if retrieval error occurs
00151
00152      ************************************************************************
00153      */
00154     void SshUserRetrieveKeys(void);
00155
00156 #ifdef __cplusplus
00157 };
00158 #endif
00159
00160     /*
00161      ************************************************************************
00162      *
00163      * "C++" Routines
00164      *
00165      ************************************************************************
00166      */
00167
00168 #endif /* _SSHUSER_H_ */
```

24.223 SshServerUserKey/src/sshuser.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SSHUSER_H_
00006 #define _SSHUSER_H_
00007
00008 #include <ssh/NetBurner/UserAuthManager.h>
00009
00010 /* Certificate and key status */
00011 #define SSH_KEY_LIBRARY_DEFAULT ((uint8_t)0x00)
00012 #define SSH_KEY_DEFAULT ((uint8_t)0x01)
00013 #define SSH_KEY_USER_INSTALLED ((uint8_t)0x02)
00014
00015 /* HTML Certificate and Key file status */
00016 #define SSH_KEY_VALID (0)
00017 #define SSH_KEY_NOT_FOUND (1)
00018 #define SSH_KEY_FILE_INVALID (2)
00019 #define SSH_KEY_CERTIFICATE_INVALID (3)
```

```
00020 #define SSH_KEY_INVALID (4)
00021
00022 /*
00023  ****************************************************************************
00024  * Key size
00025  *       SSH MAX_PRIVKEY_SIZE 1700 (options.h)
00026  *       SSH key size (PEM) < 4K (empirical)
00027  *       NULL terminated for conversion
00028  *
00029  ****************************************************************************
00030  */
00031 #define SSH_KEY_SIZE_MAX_PEM ((4 * 1024) - 1)
00032
00033 /*
00034  ****************************************************************************
00035  *
00036  * Global data definitions (declared in sshuser.cpp)
00037  *
00038  ****************************************************************************
00039  */
00040
00041 /* SSH keys PEM encoded (sshuser.cpp) */
00042 extern char *gSshRsaKeyPemEncoded[(SSH_KEY_SIZE_MAX_PEM + 1)];
00043 extern char *gSshEccKeyPemEncoded[(SSH_KEY_SIZE_MAX_PEM + 1)];
00044
00045 /*
00046  ****************************************************************************
00047  *
00048  * "C" Routines
00049  *
00050  ****************************************************************************
00051  */
00052 #ifdef __cplusplus
00053 extern "C"
00054 {
00055 #endif
00056
00057     /*
00058      ***************************************************************************
00059
00060        User provided SSH username and passuint16_t authenticate routine.
00061
00062        Parameters:
00063           usernamePtr          - Username in plain text
00064           authValPtr           - Password or Key in plain text
00065           authType             - Specify if the authorization value is a password or key
00066
00067        Return:
00068           1 - Authenticated, all else error
00069
00070        Notes:
00071           None
00072
00073      ***************************************************************************
00074      */
00075     int SshUserAuthenticate(const char *usernamePtr, const char *authValPtr, AuthType authType);
00076
00077     /*
00078      ***************************************************************************
00079
00080        User provided SSH key retrieval
00081
00082        Parameters:
00083           keyRequested         - Type key requested
00084                                        SSH_KEY_ECC
00085                                        SSH_KEY_RSA
00086           keyBufferPtr         - Key from user storage
00087           keyLengthPtr         - Size of key in 8 bit uint8_ts
00088
00089        Return:
00090           0 - key and length is valid, -1 - key requested not available
00091
00092        Notes:
00093           openSS(L|H) key pair, PEM encoded, no encrypted or with passphrase.
00094           Key must be valid. Each type asked for once at at startup.
00095           The buffer containing the key will NOT be deallocated.
00096           Server will disable task scheduling calling OSLock, copy contents, then
00097           call OSUnlock
00098
00099      ***************************************************************************
00100      */
00101     int SshUserGetKey(int keyRequested, const unsigned char **keyBufferPtr, int *keyLengthPtr);
00102
00103     /*
00104      ***************************************************************************
00105
00106        Verifies SSH key
```

```
00107
00108       Parameters:
00109          pemKeyPtr            - PEM encoded key data
00110          pemKeySize           - PEM encoded key size in uint8_ts
00111          keyTypePtr           - Pointer for key type
00112
00113       Return:
00114          TRUE - OK, FALSE invalid.
00115
00116       Notes:
00117          None
00118
00119     ***************************************************************************
00120     */
00121    BOOL SshUserVerifyKey(char *pemKeyPtr, int pemKeySize, int *keyTypePtr);
00122
00123    /*
00124     ***************************************************************************
00125
00126       Checks and installs SSH keys permanent defaults
00127
00128       Parameters:
00129          None
00130
00131       Return:
00132          None
00133
00134       Notes:
00135          Sets NV_Settings elements:
00136             SshKeyEccSource;
00137             SshKeyEccLength;
00138             SshKeyRsaSource;
00139             SshKeyRsaLength;
00140
00141     ***************************************************************************
00142     */
00143    void SshUserSetDefault(void);
00144
00145    /*
00146     ***************************************************************************
00147
00148       Retrieves and set keys
00149
00150       Parameters:
00151          None
00152
00153       Return:
00154          None
00155
00156       Notes:
00157          Clears SSH settings for CertificateNKeysDataStatus element of
00158                struct NV_SettingsStruct if retrieval error occurs
00159
00160     ***************************************************************************
00161     */
00162    void SshUserRetrieveKeys(void);
00163
00164 #ifdef __cplusplus
00165 };
00166 #endif
00167
00168    /*
00169     ***************************************************************************
00170     *
00171     * "C++" Routines
00172     *
00173     ***************************************************************************
00174     */
00175
00176 #endif /* _SSHUSER_H_ */
```

24.224 SSH/SecureSerToEthFactoryApp/src/ssluser.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SSLUSER_H_
00006 #define _SSLUSER_H_
00007
00008 /* NB Secure Sockets Layer (SSL)  */
00009 #include <crypto/ssl.h>
00010
00011 /*
00012  ***************************************************************************
```

```
00013  ********************************************************************************
00014  *
00015  * OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00016  *
00017  ********************************************************************************
00018  ********************************************************************************
00019  */
00020
00021 /*
00022  ********************************************************************************
00023  *
00024  * Global data definitions
00025  *
00026  ********************************************************************************
00027  */
00028 /* Default or installed user PEM encoded certificate */
00029 extern char gSslCert[(SERIAL_BURNER_CERTIFICATE_SIZE_MAX_PEM + 1)];
00030 extern char gSslKey[(SERIAL_BURNER_KEY_SIZE_MAX_PEM + 1)];
00031
00032 /*
00033  ********************************************************************************
00034  *
00035  * Functions
00036  *
00037  ********************************************************************************
00038  */
00039 /*
00040  ********************************************************************************
00041  *
00042  * "C" Routines
00043  *
00044  ********************************************************************************
00045  */
00046 #ifdef __cplusplus
00047 extern "C"
00048 {
00049 #endif
00050     /*
00051      ********************************************************************************
00052
00053        Checks and installs SSL default certificate and key
00054
00055        Parameters:
00056           None
00057
00058        Return:
00059           None
00060
00061        Notes:
00062           Sets NV_Settings elements:
00063              SslCertificateSource;
00064              SslCertificateLength;
00065              SslKeyLength;
00066
00067
00068      ********************************************************************************
00069      */
00070     void SslUserSetDefault(void);
00071
00072     /*
00073      ********************************************************************************
00074
00075        Retrieves and set certificate and key
00076
00077        Parameters:
00078           None
00079
00080        Return:
00081           None
00082
00083        Notes:
00084           Clears SSL settings for CertificateNKeysDataStatus element of
00085                 struct NV_SettingsStruct if retrieval error occurs
00086
00087      ********************************************************************************
00088      */
00089     void SslUserRetrieveCertificateNKey(void);
00090
00091 #ifdef __cplusplus
00092 };
00093 #endif
00094
00095     /*
00096      ********************************************************************************
00097      *
00098      * "C++" Routines
00099      *
```

```
00100       ***************************************************************************
00101       */
00102
00103 #endif /* _SSLUSER_H_ */
```

24.225 SSL/HttpsUploadCert/src/ssluser.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SSLUSER_H_
00006 #define _SSLUSER_H_
00007
00008 /* NB Secure Sockets Layer (SSL)  */
00009 #include <crypto/ssl.h>
00010
00011 /* Default or installed user PEM encoded certificate */
00012 extern char gSslCert[(SERIAL_BURNER_CERTIFICATE_SIZE_MAX_PEM + 1)];
00013 extern char gSslKey[(SERIAL_BURNER_KEY_SIZE_MAX_PEM + 1)];
00014
00015 #ifdef __cplusplus
00016 extern "C"
00017 {
00018 #endif
00019
00028     void SslUserSetDefault(void);
00029
00036     void SslUserRetrieveCertificateNKey(void);
00037
00038 #ifdef __cplusplus
00039 };
00040 #endif
00041
00042 #endif /* _SSLUSER_H_ */
```

24.226 UserAuth.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _USER_AUTH_H_
00006 #define _USER_AUTH_H_
00007
00008 #include <predef.h>
00009
00010 #include <ssh/NetBurner/UserAuthManager.h>
00011
00012 int SaveAuthRecordsUserParam(const UserAuthRecord *authRec);
00013 int LoadAuthRecordsUserParam(UserAuthRecord *authRec);
00014 int AuthenticateUser(const char *usernamePtr, const char *authPtr, AuthType authType);
00015
00016 AuthResponse ProcessAddNewUser();
00017 AuthResponse ProcessDeleteUser();
00018
00019 #endif /* _USER_AUTH_H_ */
```

24.227 permanentkeyecc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *   Default SSH ECC key in openSSL (openSSH) format
00007  *   OpenSSL format is Privacy-enhanced Electronic Mail (PEM) encoded
00008  *
00009  */
00010
00011 #ifndef _PERMANENTKEYECC_H_
00012 #define _PERMANENTKEYECC_H_
00013
00014 "-----BEGIN EC PRIVATE KEY-----\r\
00015 MHcCAQEEIJQCd0y9U8mFVUWqJUNRX2FYkPTBuuYu1MFSm6QaGxA7oAoGCCqGSM49\r\
00016 AwEHoUQDQgAE++54xjrZKCfpmMYqaEdP2w0fTkXC6FEuWp1sg/uuOXiF2cS7GHY/\r\
00017 6PYVqh7SjIH+1QUxaRKlsIhXsIguY0tW3w==\r\
00018 -----END EC PRIVATE KEY-----\r"
00019
```

```
00020 #endif /*_PERMANENTKEYECC_H_*/
```

24.228 acmeRFC8555.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_ACME_H
00006 #define _NB_ACME_H
00007
00008 #include <predef.h>
00009 #include <ip.h>
00010 #include <nettypes.h>
00011
00012
00013 #include <base64.h>
00014 #include <json_lexer.h>
00015 #include <nbstring.h>
00016 #include <diagnostics.h>
00017 #include <webclient/http_funcs.h>
00018
00019 #include <crypto/wolfssl/wolfcrypt/ecc.h>
00020 #include <crypto/wolfssl/wolfcrypt/random.h>
00021 #include <crypto/wolfssl/wolfcrypt/sha256.h>
00022 #include <crypto/wolfssl/wolfcrypt/hash.h>
00023 #include <crypto/wolfssl/wolfcrypt/asn_public.h>
00024 #include <crypto/wolfssl/wolfcrypt/wolfmath.h>
00025 #include <crypto/certgen.h>
00026
00027
00028 extern const char * DefAcmeUrl;
00029
00030 const int ACME_ERR_NONE=0;
00031 const int ACME_ERR_HAL_SAVE_FAIL=-10;
00032 const int ACME_ERR_CSR_ERR=-11;
00033 const int ACME_ERR_FAIL_READ_EXPIRY=-12;
00034 const int ACME_ERR_STORED_CERT_INVALID=-14;
00035 const int ACME_FAILED_GET_DIR=-15;
00036 const int ACME_FAILED_GET_NONCE=-16;
00037 const int ACME_FAILED_GET_NONCE_DIR=-17;
00038 const int ACME_FAILED_GET_ACCOUNT=-18;
00039 const int ACME_ERR_GET_CSR_FAIL=-19;
00040 const int ACME_ERR_ORDER_FAIL=-20;
00041 const int ACME_ERR_MALLOC_FAIL=-21;
00042 const int ACME_ERR_SELF_FAIL=-22;
00043
00044
00045
00046
00047 //Class to handel header processing as ACME needs both headers and body processed.
00048 class AcmeObject;
00049 class AcmeJsonBuffer : public ParsedJsonDataSet
00050
00051 {
00052 AcmeObject * pAcme;
00053     virtual void ProcessHeader(const char * hdr);
00054 public:
00055     AcmeJsonBuffer(AcmeObject * pOwner) {pAcme=pOwner;}
00056
00057 };
00058
00059
00060
00061
00062 class AcmeObject : public config_obj , public HtmlPageHandler ,public DiagItemClass
00063 {
00064 config_string m_DnsName;
00065 config_string m_AcmeServerDir;
00066     ConfigEndMarker;   // No new data members below this line
00067
00068
00069 AcmeJsonBuffer DirListing;  //Holds the rood directoory listing of services.
00070 AcmeJsonBuffer OrderResult; //Holds the order in process.
00071 AcmeJsonBuffer TransactionResult; //Temp holds the transactions...
00072
00073
00074 NBString nonce;
00075 NBString jwk;
00076 NBString kid;
00077 NBString AuthItem;
00078 NBString FinalizeUrl;
00079 NBString AuthToken;
00080 NBString AuthUrl;
00081 NBString AuthPath;
```

```
00082 NBString RetryAfter;
00083 NBString Location;
00084
00085 ecc_key  AccountKey;
00086 ecc_key  ServerKey;
00087 uint32_t keysize;
00088
00089 //Used by SSL to get keys etc...
00090 puint8_t m_pServerKey;
00091 puint8_t m_pServerCert;
00092
00093
00094 //State variables
00095 bool m_bSelfSigned;  //Is cert selfsigned?
00096 bool m_bKeyInit;     //Have keys been initialized
00097 time_t m_expiry;     //When does cert expire
00098 time_t m_issued;     //When was cert issued?
00099 time_t m_retrydate;  //When shoudl we renew?
00100 NBString m_CertIssuer; //Name of entity who signed cert.
00101
00102
00103
00104 WC_RNG rng;
00105
00106 //Detect when response is queried
00107 //When the verification happens this SEM is posted to.
00108 OS_SEM HitSem;
00109
00110
00111 //The erorr state of the world.
00112 NBString err_str;
00113 int err_code;
00114
00115 //Active status of the world
00116 NBString status_str;
00117
00118
00119 private:
00120
00121 //Access the things we need to generate responses...
00122     NBString GetJwk(); //Java web key
00123     NBString GetThumb(); //Thumb print of account key
00124     bool GetCSR(NBString & s); //Generate a Certificate signing request
00125
00126 //Global pointer so we can get the one and only object for SSL key handeling
00127 static AcmeObject * TheOneAndOnly;
00128
00129
00130
00131
00132 //Helper function Used to scan incomming headers for things of interest.
00133 bool ScanHeaderAndSet(const char * pTarget, NBString & setv,const char * hdr);
00134
00135 //Called for every header in incomming transactions.
00136 void ProcessHeader(const char * hdr);
00137
00138
00139 //Will be replaced.
00140 PoolPtr PrepTransaction(const NBString & url, const char* payload="", bool bJwk=false);
00141 bool DoTransaction(AcmeJsonBuffer & buf,const char * dir_entry, const char* payload="", bool
      bJwk=false);
00142 bool DoTransactionUrl(AcmeJsonBuffer & buf,const NBString & url, const char* payload="", bool
      bJwk=false);
00143
00144
00145 //Save current active keys to storage
00146 bool SaveKeysToStorage();
00147
00148 //Make and save to storage a selfsigned cert...
00149 bool MakeSaveSelfSignedCert();
00150
00151 //Read the server cert and set the issue and expiry date
00152 void GetCertTimes();
00153
00154 //HtmlPageHandler callback used to serve up the response page
00155 //Possible this might change from is one to has one...
00156 virtual int ProcessRaw(int sock, HTTP_Request &pd);
00157
00158
00159 //DigItemClass virtual for showing state of the world on diag item.
00160 virtual void ServeContent(int fd);
00161
00162
00163
00164
00165 void RefreshNonce();
00166 bool NewAccount();
```

```
00167 bool NewOrder();
00168 void InitKeys();
00169 void CheckCert();
00170
00171 public:
00172
00173
00174 //Constructor builds the object...
00175 AcmeObject(const char * dns_name, const char * pUrlDir=DefAcmeUrl)
00176 :config_obj(appdata,"AcmeService", "The ACME RFC8555 service"),
00177 HtmlPageHandler("/.well-known/acme-challenge/*"),
00178 DiagItemClass("ACMEClient"),
00179 m_DnsName(dns_name, "DNSName","The DNS Name you want the acme server to use getting a certificate"),
00180 m_AcmeServerDir(pUrlDir,"AcmeServerUrl", "The URL for the acme service directory"),
00181 DirListing(this),
00182 OrderResult(this),
00183 TransactionResult(this)
00184 {TheOneAndOnly=this;};
00185
00186
00187
00188 void StartAcme(bool ForceNewCert=false);
00189 void ServiceAcme();
00190
00191 //Access the one and only needed to serve SSL keys.
00192 static AcmeObject * GetAcme();
00193
00194
00195 friend CertGenData *GetDataForCertGen();
00196 friend class AcmeJsonBuffer;
00197 friend int GetPrivateKeyLen();
00198 friend int GetCertificateLen();
00199 friend const char * GetPrivateKeyPEM();
00200 friend const char * GetCertificatePEM();
00201
00202
00203
00204
00205
00206
00207 };
00208 #endif
```

24.229 caList.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 const uint32_t GithubCertLen = 1345;
00006 const unsigned char GithubCert[GithubCertLen] = {
00007     "-----BEGIN CERTIFICATE-----"
00008     "MIIDxTCCAq2gAwIBAgIQAqxcJmoLQJuPC3nyrkYldzANBgkqhkiG9w0BAQUFADBsMQswCQYDVQQG"
00009     "EwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3d3cuZGlnaWNlcnQuY29tMSsw"
00010     "KQYDVQQDEyJEaWdpQ2VydCBIaWdoIEFzc3VyYW5jZSBFViBSb290IENBMB4XDTA2MTExMDAwMDAw"
00011     "MFoXDTMxMTExMDAwMDAwMFowbDELMAkGA1UEBhMCVVMxFTATBgNVBAoTDERpZ2lDZXJ0IEluYzEZ"
00012     "MBcGA1UECxMQd3d3LmRpZ2ljZXJ0LmNvbTErMCkGA1UEAxMiRGlnaUNlcnQgSGlnaCBBc3N1cmFu"
00013     "Y2UgRVYgUm9vdCBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMbM5XPm+9S75S0t"
00014     "Mqbf5YE/yc0lSbZxKsPVlDRnogocsF9ppkCxxLeyj9CYpKlBWTrT3JTWPNt0OKRKzE0lgvdKpVMS"
00015     "OO7zSW1xkX5jtqumX8OkhPhPYlG++MXs2ziS4wblCJEMxChBVfvLWokVfnHoNb9Ncgk9vjo4UFt3"
00016     "MRuNs8ckRZqnrG0AFFoEt7oT61EKmEFBIk5lYYeBQVCmeVyJ3hlKV9Uu5l0cUyx+mM0aBhakaHPQ"
00017     "NAQTXKFx01p8VdteZOE3hzBWBOURtCmAEvF5OYiiAhF8J2a3iLd48soKqDirCmTCv2ZdlYTBoSUe"
00018     "h10aUAsgEsxBu24LUTi4S8sCAwEAAaNjMGEwDgYDVR0PAQH/BAQDAgGGMA8GA1UdEwEB/wQFMAMB"
00019     "Af8wHQYDVR0OBBYEFLE+w2kD+L9HAdSYJhoIAu9jZCvDMB8GA1UdIwQYMBaAFLE+w2kD+L9HAdSY"
00020     "JhoIAu9jZCvDMA0GCSqGSIb3DQEBBQUAA4IBAQAcGgaX3NecnzyIZgYIVyHbIUf4Kmeqvxgydk AQ"
00021     "V8GK83rZEWWONfqe/EW1ntlMMUu4kehDLI6zeM7b41N5cdblIZQB2lWHmiRk9opmzN6cN82oNLFp"
00022     "myPInngiK3BD41VHMWEZ71jFhS9OMPagMRYjyOfiZRYzy78aG6A9+MpeizGLYAiJLQwGXFK3xPkK"
00023     "mNEVX58Svnw2Yzi9RKR/5CYrCsSXaQ3pjOLAEFe4yHYSkVXySGnYvCoCWw9E1CAx2/S6cCZdkGCe"
00024     "vEsXCS+0yx5DaMkHJ8HSXPfqIbloEpw8nL+e/IBcm2PN7EeqJSdnoDfzAIJ9VNep+OkuE6N36B9K"
00025     "-----END CERTIFICATE-----"};
```

24.230 Advanced/src/TimeUtil.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 void setSystemTime(char *timeZoneName);
00009 void displaySystemTime();
```

```
00010
00011 #endif
```

24.231 CompiledCa/src/TimeUtil.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 void setSystemTime(char *timeZoneName);
00009 void displaySystemTime();
00010
00011 #endif
```

24.232 Simple/src/TimeUtil.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _EFFS_TIME_H
00006 #define _EFFS_TIME_H
00007
00008 void setSystemTime(char *timeZoneName);
00009 void displaySystemTime();
00010
00011 #endif
```

24.233 TcpClientSimple/src/clientweb.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_CLIENTWEB_H
00006 #define _NB_CLIENTWEB_H
00007
00008 //----- Function Prototypes -----
00009 void RegisterPost();
00010
00011 #endif
```

24.234 TcpMultiInterfaceTest/src/clientweb.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_CLIENTWEB_H
00006 #define _NB_CLIENTWEB_H
00007
00008 //----- Function Prototypes -----
00009 void RegisterPost();
00010
00011 #endif
```

24.235 webif.cpp File Reference

This module handles the web page interface to the UDP to Serial program example.

```
#include <basictypes.h>
#include <http.h>
#include <iosys.h>
#include <stdlib.h>
#include <string.h>
#include <system.h>
#include <httppost.h>
```

```
#include "webif.h"
```

Functions

- void WebLocalPort (int sock, PCSTR url)
- void WebDestPort (int sock, PCSTR url)
- void WebDestIp (int sock, PCSTR url)
- void CheckNVSettings ()

24.235.1 Detailed Description

This module handles the web page interface to the UDP to Serial program example.

24.235.2 Function Documentation

24.235.2.1 CheckNVSettings()

```
void CheckNVSettings ( )
```

Check NV Settings. This function will check the flash memory user parameter storage area for valid stored values. If the values are invalid (VerifyKey), it will assign default values.

24.235.2.2 WebDestIp()

```
void WebDestIp (
            int sock,
            PCSTR url )
```

Function to display current destination IP address

24.235.2.3 WebDestPort()

```
void WebDestPort (
            int sock,
            PCSTR url )
```

Function to display destination port number

24.235.2.4 WebLocalPort()

```
void WebLocalPort (
            int sock,
            PCSTR url )
```

Function to display destination port number

24.236 webif.h File Reference

Classes

- struct NV_SettingsStruct

  Configuration Settings.

24.237 webif.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
```

```
00004
00009 #define APP_VERSION "Version 1.1 11-Jan-2018"
00010 #define LOCAL_PORT 3333
00011 #define DEST_PORT 3334
00012 #define DEST_IPADDR "10.1.1.104"
00013
00014 #define POST_BUFSIZE (4096)
00015
00016 #define VERIFY_KEY (0x18256052)   // NV Settings key code
00017 #define MAX_IPADDR_LEN (20)
00018
00019 struct NV_SettingsStruct   // Non-volatile storage structure
00020 {
00021     uint32_t VerifyKey = 0;              // Flash memory key for initialization
00022     int nLocalPort = 0;                  // Local UDP port to listen on
00023     int nDestPort = 0;                   // Destination UDP port to send serial data
00024     char szDestIpAddr[MAX_IPADDR_LEN];   // Destination UDP ip address
00025 };
```

24.238 datagenerator.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __DATAGENERATOR_H
00006 #define __DATAGENERATOR_H
00007
00008 #include "datalog.h"
00009
00010 #define LOG_SIZE 1000
00011 #define SENSOR_1_PERIOD 13
00012 #define SENSOR_2_PERIOD 11
00013 #define SENSOR_1_INIT 44.23
00014 #define SENSOR_2_INIT -10.0
00015
00016 extern RingLog dataLog;
00017
00018 int32_t SeedLog();
00019 void LogSensor(int id, int32_t tick);
00020
00021 #endif /* __DATAGENERATOR_H */
```

24.239 datalog.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __DATALOG_H
00006 #define __DATALOG_H
00007
00008 #include <stdint.h>
00009
00010 struct dataStruct
00011 {
00012     int sensorID = 0;
00013     uint32_t timeStamp = 0;
00014     float value = 0.0;
00015 };
00016
00017 typedef void (*RingLogSerializer)(dataStruct *item, void *args);
00018
00019 class RingLog
00020 {
00021    private:
00022     dataStruct *pHead = nullptr;
00023     dataStruct *pTail = nullptr;
00024     int32_t count = 0;
00025
00026     const dataStruct *pStart = nullptr;
00027     const dataStruct *pEnd = nullptr;
00028     const int32_t maxSize = 0;
00029
00030     void IncrementTail();
00031     void IncrementHead();
00032     void IncrementPtr(dataStruct *&ptr);
00033
00034    public:
00035     RingLog(dataStruct *buffer, uint32_t bufferSize);
00036     ~RingLog();
00037
```

```
00038     void Add(dataStruct *item);
00039     void Remove(dataStruct *item);
00040     void Clear();
00041     int32_t GetCount();
00042     void Dump();
00043     void Serialize(RingLogSerializer serializer, void *args);
00044 };
00045
00046 #endif /* #ifndef __DATALOG_H  */
```

24.240 webFormValues.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _WEB_FORM_VALUES_H_
00006 #define _WEB_FORM_VALUES_H_
00007 #pragma once
00008
00009 #define DEF_WIDTH (250)
00010 #define DEF_HT (250)
00011 #define DEF_FRED (150)
00012 #define DEF_FGREEN (0)
00013 #define DEF_FBLUE (250)
00014 #define DEF_BRED (250)
00015 #define DEF_BGREEN (150)
00016 #define DEF_BBLUE (0)
00017 #define DEF_TRED (255)
00018 #define DEF_TGREEN (255)
00019 #define DEF_TBLUE (255)
00020
00021 /* This class defines a BAR size width etc... */
00022
00023 class WebFormValues
00024 {
00025    public:
00026     uint16_t m_ht = DEF_HT;
00027     uint16_t m_wid = DEF_WIDTH;
00028     uint8_t m_fillRed = DEF_FRED;
00029     uint8_t m_fillGreen = DEF_FGREEN;
00030     uint8_t m_fillBlue = DEF_FBLUE;
00031     uint8_t m_borderRed = DEF_BRED;
00032     uint8_t m_borderGreen = DEF_BGREEN;
00033     uint8_t m_borderBlue = DEF_BBLUE;
00034     uint8_t m_textRed = DEF_TRED;
00035     uint8_t m_textGreen = DEF_TGREEN;
00036     uint8_t m_textBlue = DEF_TBLUE;
00037
00038     WebFormValues();
00039     WebFormValues(PCSTR url);
00040     void WriteUrl(int sock, PCSTR Prefix);
00041 };
00042
00043 #endif /* _BAR_DEFINITION_H_ */
```

24.241 WebFunctions.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef WEBFUNCTIONS_H_
00006 #define WEBFUNCTIONS_H_
00007
00008 extern void RegisterWebHandler();
00009 extern bool WifiInitComplete;
00010
00011 #endif /*WEBFUNCTIONS_H_*/
```

24.242 CryptoServer.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _CRYPTO_SERVER_H_
00006 #define _CRYPTO_SERVER_H_
00007
```

```
00008 #include <predef.h>
00009 #if defined(NB_SSL_SUPPORTED) || defined(NB_SSH_SUPPORTED)
00010
00011 enum CryptoServerErrors : int16_t
00012 {
00013     eCryptoServerErrorInvalidSocketType = -1
00014 };
00015
00016 enum CryptoQueueAction : unsigned char
00017 {
00018     eCryptoQueueActionError = 1,
00019     eCryptoQueueActionTcpRead = 2,
00020     eCryptoQueueActionTcpClose = 3,
00021     eCryptoQueueActionExtraFdClose = 4
00022 };
00023
00024 int ioReturnCode(int n, int tcpFd, SocketType_t sockType);
00025
00026 void InitCryptoServer();
00027 void PostCryptoQueueData(CryptoSocket *sck, CryptoQueueAction action);
00028
00029 void CryptoReadNotify(int fd);
00030 void CryptoWriteNotify(int fd);
00031
00032 int WolfCryptoCbIoRecv(int tcp, char *buf, int len, void *ctx, SocketType_t sckType);
00033 int WolfCryptoCbIoSend(int tcp, char *buf, int len, void *ctx, SocketType_t sckType);
00034
00035 int WolfCryptoExtraFdRead(int fd, char *buf, int nbytes);
00036 int WolfCryptoExtraFdWrite(int fd, const char *buf, int nbytes);
00037 int WolfCryptoExtraFdClose(int extra_fd);
00038
00039 #endif   // NB_SSL_SUPPORTED || NB_SSH_SUPPORTED
00040 #endif   //_CRYPTO_SERVER_H_
```

24.243 CryptoSocket.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _CRYPTO_SOCKET_H_
00006 #define _CRYPTO_SOCKET_H_
00007
00008 #include <basictypes.h>
00009 #include <buffers.h>
00010 #include <netinterface.h>
00011 #include <iointernal.h>
00012
00013 #define NUM_WOLF_CRYPTO_SOCKETS (16)
00014 #define NUM_WOLF_CRYPTO_SESSIONS (16)
00015 #define CRYPTO_BUFFER_SEGMENTS TCP_BUFFER_SEGMENTS
00016
00017 #define CRYPTO_OBJ_FLAG_WNOTIFY_IGNORE (0x0001)
00018 #define CRYPTO_OBJ_FLAG_USER_CLOSED (0x0002)   // User program called close on the SSL/ExtraFd.
00019 #define CRYPTO_OBJ_FLAG_DOINGSHUTDOWN (0x0004)
00020 #define CRYPTO_OBJ_FLAG_DOINGCONNECT (0x0008)
00021 #define CRYPTO_OBJ_FLAG_DOINGACCEPT (0x0010)
00022 #define CRYPTO_OBJ_FLAG_HAVEERROR (0x0020)
00023 #define CRYPTO_OBJ_FLAG_OTHER_SIDE_CLOSED (0x0040)
00024 #define CRYPTO_OBJ_FLAG_RESET_SOCKET \
00025     (0x0080)   // If we fail during negotiation, we need to go ahead and reset the full socket as the
      user won't get
00026                // a valid FD to close, which will result in a socket leak.
00027 #define CRYPTO_OBJ_FLAG_ERROR_RETURNED \
00028     (0x0100)   // If we are resetting the socket in the asynch task, we want to make sure we don't do
      so until the
00029                // connection has returned any errors.
00030 #define CRYPTO_OBJ_FLAG_TCP_RESET (0x0200)   // We recieved a reset or abort on the tcp conneciton.
00031
00032 // Time stuff
00033 #define SHUTDOWN_RETRY_INTERVAL_IMMEDIATE \
00034     (1)   // Because of our asynchronous connections, we will generally have to try at least twice to
      completely
00035           // shutdown with WolfSSL. However, the time between these calls can be extremely short.
00036 #define SHUTDOWN_RETRY_INTERVAL (TICKS_PER_SECOND)
00037 #define CONNECT_RETRY_INTERVAL (TICKS_PER_SECOND)
00038
00039 enum SocketType_t : unsigned char
00040 {
00041     SocketType_Default = 0,
00042     SocketType_Tls = 1,
00043     SocketType_Ssh = 2,
00044 };
00045
00046 enum SocketHasData_t : int
```

```
00047 {
00048     SocketHasData_NoData = 0,
00049     SocketHasData_HasData = 1,
00050     SocketHasData_PendingData = 2
00051 };
00052
00053 class CryptoSocket
00054 {
00055 public:
00056     // Member variables
00057     SocketType_t m_socketType = SocketType_Default;
00058
00059     OS_CRIT m_Crit;      // Critical seciton to keep wolf multithread safe
00060     int m_error = 0;     // The error value
00061     int m_options = 0;   // Options like push etc ...
00062
00063     PoolPtr m_NoPushbuffer;
00064     fifo_buffer_storage m_dataTcpOverflow;   // Used to hold overflow on outbound data to tcp
00065
00066     // Will either be a WOLFSSL* or WOLFSSH*
00067     void *m_wolfCtx = nullptr;
00068
00069     // Volatile becasue they can be touched from multiple places
00070     volatile int m_tcpFd = 0;                       // The underlyign tcp socket
00071     volatile int m_cryptoExtraFd = 0;               // The ExtraFd the user code uses to interact with
      thei socket
00072     OS_FLAGS m_OS_Flags; //State flags
00073
00074     volatile uint32_t m_TimeTickOfNextAction = 0;   // Time tick to do some time releated thing
00075
00076     // This is an index that gets incrmented every time a socket is reset.
00077     // This is used to detect race conditions on the worker queue to keep
00078     // messages from sockets already closed from getting reused.
00079     volatile uint8_t m_IndexOfReuse;
00080     volatile uint8_t m_RetryCount;
00081
00082     CryptoSocket();
00083     virtual ~CryptoSocket();
00084
00085     SocketType_t GetSocketType() { return m_socketType; }
00086     bool InUse();
00087
00088     // Virtual Fuctions
00089     // when push gets turned off or the socket gets closed with data to be written we cleanup here
00090     void CleanupUnWritten();
00091
00092     // Called from wolf loop mostly to handle notifications and
00093     // time out stuff for all async process except read. Needs to be
00094     // defined for each socket type.
00095     virtual void ProcessAsyncStuff() = 0;
00096
00097     virtual int CheckSocketRecv() = 0;
00098
00099     // Final clean up when we are done usign this socket.
00100     virtual void ResetSocket();
00101
00102     // Called when the underlying TCP socket got closed
00103     virtual void CleanupTcpClose();
00104
00105     // Flag Functions
00106     inline void SetFlag(uint32_t flag) { m_OS_Flags.Set(flag); }
00107     inline void ClrFlag(uint32_t flag) { m_OS_Flags.Clear(flag); }
00108     inline bool GetFlag(uint32_t flag) { return (m_OS_Flags.State()&flag)!=0;}
00109     inline uint32_t GetFlags() { return m_OS_Flags.State();}
00110     inline void ClrAllFlags() { m_OS_Flags.Clear(0xFFFFFFFF); }
00111     inline bool PendFlagUntil(uint32_t flag, uint32_t Time) {return
      (m_OS_Flags.PendAnyUntil(flag,Time)!=OS_TIMEOUT);}
00112     inline bool PendFlagDly(uint32_t flag, uint32_t Delay){return
      (m_OS_Flags.PendAny(flag,Delay)!=OS_TIMEOUT);};
00113
00114     // Options functions
00115     inline int GetOptions() { return m_options; }
00116     inline void SetOptions(int option) { m_options |= option; }
00117     inline void ClrOptions(int option) { m_options &= ~option; }
00118
00119     virtual uint32_t GetCheckableIndexFromSocket() = 0;
00120     virtual uint32_t SocketRead(char *buf, uint32_t len) = 0;
00121     virtual uint32_t SocketWrite(const char *buf, uint32_t len) = 0;
00122     virtual SocketHasData_t SocketHasData() = 0;
00123
00124     inline int GetTcpFd() { return m_tcpFd; }
00125     inline int GetCryptoExtraFd() { return m_cryptoExtraFd; }
00126     inline int GetError() { return m_error; }
00127     inline void SetError(int err) { m_error = err; }
00128     inline uint32_t GetTimeOfNextAction() { return m_TimeTickOfNextAction; }
00129     inline OS_CRIT &GetSockCrit() { return m_Crit; }
00130
```

```
00131     // Buffer Handling
00132     inline fifo_buffer_storage &GetTcpOverflow() { return m_dataTcpOverflow; }
00133     inline PoolPtr GetNoPushBuffer() { return m_NoPushbuffer; }
00134     inline void SetNewNoPushBuffer() { m_NoPushbuffer = GetBuffer(); }
00135     inline bool IsNoPushBufferNull() { return (m_NoPushbuffer == 0); }
00136
00137     void SetCryptoExtraFd(int cpExFd) { m_cryptoExtraFd = cpExFd; }
00138
00139     static CryptoSocket *GetSocketFromTcpFd(int tcpFd);
00140     static CryptoSocket *GetSocketFromIndex(uint32_t index);
00141     static CryptoSocket *GetSocketFromCryptoFd(int cryptoFd);
00142
00143    protected:
00144     void InitSocket(int tcpFd, IoExpandStruct *ioExpStr);
00145
00146     bool CloseTcpFd(IPADDR &remoteAddr, int &remotePort);
00147
00148     // Static Member Functions
00149     // Need to add a GetNewSocket() for each derived type
00150     static CryptoSocket *GetSocketFromIndex(CryptoSocket *cryptoSockets, uint32_t index);
00151     static CryptoSocket *GetSocketFromTcpFd(CryptoSocket *cryptoSockets, int tcpFd);
00152     static CryptoSocket *GetSocketFromCryptoFd(CryptoSocket *cryptoSockets, int cryptoFd);
00153
00154     uint32_t GetCheckableIndexFromSocket(CryptoSocket *cryptoSockets);
00155
00156     // The following should be defined for each class type
00157     virtual void WriteUnwrittenData() = 0;
00158     virtual bool PendOnHandshake() { return false; }
00159
00160     void ClearNextActionTimeTick() { m_TimeTickOfNextAction = 0; };
00161     void SetNextActionTimeTick(uint32_t time);
00162     void SetSocketError(int error, bool setTcpClose, bool resetSocket);
00163     void ResetSequence(bool success);
00164
00165     static CryptoSocket *FindNextEmptySocket(CryptoSocket *cryptoSockets);
00166 };
00167
00168 #endif   // _CRYPTO_SOCKET_H_
```

24.244 memoryAllocator.h

```
00001 #ifndef _MEMORY_ALLOCATOR_H_
00002 #define _MEMORY_ALLOCATOR_H_
00003
00004 #include <stdint.h>
00005 #include <predef.h>
00006
00007 /* C Standard Library */
00008 #include <ctype.h>
00009 #include <malloc.h>
00010 #include <stdio.h>
00011
00012 /* Portability & uCos Definitions */
00013 #include <basictypes.h>
00014 #include <nbrtos.h>
00015 #include <nbrtoscpu.h>
00016
00017 /* NB Runtime Libraries */
00018 #include <constants.h>
00019
00020 #ifdef __cplusplus
00021 class MemoryAllocator
00022 {
00023    public:
00024     virtual void *allocate(size_t size) = 0;
00025     virtual void deallocate(void *ptr) = 0;
00026     virtual void *reallocate(void *ptr, size_t size) = 0;
00027     virtual ~MemoryAllocator() {}
00028 };
00029
00030 class BlockMemoryAllocator : public MemoryAllocator
00031 {
00032    public:
00033     BlockMemoryAllocator(void *pMemory, size_t memorySize, size_t blockSize);
00034     ~BlockMemoryAllocator() override;
00035     void *allocate(size_t size) override;
00036     void deallocate(void *ptr) override;
00037     void *reallocate(void *ptr, size_t size) override;
00038
00039     void printStats() const;
00040    private:
00041     struct Block
00042     {
00043         Block *prev;
00044         Block *next;
```

```
00045         bool free;
00046     };
00047     void *_pMemory;
00048     size_t _memorySize;
00049     OS_CRIT m_critSec;
00050     Block *_pBlockList;
00051     size_t _blockSizeMultiplier;   // All blocks are a multiple of this size
00052 
00053     const size_t largestSizeToAllocate = 10560;
00054 
00055     // Profiling data
00056     size_t _numAllocations;
00057     size_t _numDeallocations;
00058     size_t _numOOMAllocations;
00059     size_t _numToLargeAllocations;
00060     size_t _allocatedMemory;
00061 
00062     size_t getBlockSize(Block *pBlock) const;
00063     size_t getAlignedBlockSize(size_t size) const;
00064     Block *getBlockFromPointer(void *pBlock) const;
00065     void splitBlock(Block *pBlock, size_t size);
00066     void coalesceBlocks(Block *pBlock);
00067 
00068 };
00069 
00070 #define CREATE_MEMORY_ALLOCATOR_SRAM(name, memorySize)                          \
00071     alignas(4) static uint8_t memoryBuffer##name[memorySize] FAST_USER_VAR; \
00072     BlockMemoryAllocator name(memoryBuffer##name, memorySize, 32)
00073 
00074 #define CREATE_MEMORY_ALLOCATOR_TCM(name, memorySize)                           \
00075     alignas(4) static uint8_t memoryBuffer##name[memorySize] FAST_SYS_VAR; \
00076     BlockMemoryAllocator name(memoryBuffer##name, memorySize, 32)
00077 
00078 #define CREATE_MEMORY_ALLOCATOR_SDRAM(name, memorySize)         \
00079     alignas(4) static uint8_t memoryBuffer##name[memorySize]; \
00080     BlockMemoryAllocator name(memoryBuffer##name, memorySize, 32)
00081 
00082 #endif
00083 #endif   // _MEMORY_ALLOCATOR_H_
```

24.245 NbSslCtx.h

```
00001 /*NB_REVISION*/
00002 
00003 /*NB_COPYRIGHT*/
00004 
00005 #ifndef _NB_SSL_CTX_H_
00006 #define _NB_SSL_CTX_H_
00007 
00008 #include <crypto/wolfssl/internal.h>
00009 
00010 enum SslCtxVerifyPeer
00011 {
00012     SslCtxVerifyPeer_default    = 0,
00013     SslCtxVerifyPeer_on,
00014     SslCtxVerifyPeer_off,
00015 };
00016 
00017 class NbSslCtx
00018 {
00019     public:
00020         bool Create( bool isServer );
00021 
00022         WOLFSSL_CTX* GetCtx(){ return m_wolfCtx; }
00023 
00024         void SetInit(bool init){ m_ctxInit = init; }
00025         bool GetInit(){ return m_ctxInit; }
00026 
00027         bool GetCaSet(){ return m_ctxCaSet; }
00028         void SetCaSet(bool set){ m_ctxCaSet = set; }
00029 
00030         SslCtxVerifyPeer GetVerifyPeer(){ return m_verifyPeer; }
00031         void SetVerifyPeer( SslCtxVerifyPeer verifYPeer ){ m_verifyPeer = verifYPeer; }
00032 
00033     private:
00034         WOLFSSL_CTX*        m_wolfCtx       = nullptr;
00035         bool                m_ctxInit       = false;
00036         bool                m_ctxCaSet      = false;
00037         SslCtxVerifyPeer    m_verifyPeer    = SslCtxVerifyPeer_default;
00038 };
00039 
00040 #endif /* _NB_SSL_CTX_H_ */
```

24.246 NbWolfSsl.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_WOLF_SSL_H_
00006 #define _NB_WOLF_SSL_H_
00007
00008 #include <predef.h>
00009 #ifdef NB_SSL_SUPPORTED
00010
00011 #include <crypto/wolfssl/ssl.h>
00012 #include <crypto/wolfssl/callbacks.h>
00013
00014 void CheckGenerateNewCert();
00015
00016 #endif /* NB_SSL_SUPPORTED */
00017 #endif
```

24.247 SslClientSession.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SSL_SESSION_ID_H_
00006 #define _SSL_SESSION_ID_H_
00007
00008 #include <predef.h>
00009 #ifdef NB_SSL_SUPPORTED
00010
00011 #include <buffers.h>
00012 #include <tcp.h>
00013 #include <netinterface.h>
00014
00015 #include <crypto/wolfssl/ssl.h>
00016 #include <crypto/NetBurner/SslSocket.h>
00017
00018 // Used for session resumption with session ID's
00019 class SslClientSession
00020 {
00021    public:
00022     ~SslClientSession();
00023
00024     inline void SetSession(IPADDR &ip, int port, WOLFSSL_SESSION *session);
00025     void ClearSession();
00026
00027     IPADDR m_ipAddr = IPADDR::NullIP();
00028     int m_port = 0;
00029     OS_CRIT m_Crit;   // Critical seciton to keep wolf multithread safe
00030     WOLFSSL_SESSION *m_session = nullptr;
00031 };
00032
00033 WOLFSSL_SESSION *FindSession(IPADDR &ipAddr, int port);
00034 bool SaveSession(IPADDR &ipAddr, int port, WOLFSSL_SESSION *session);
00035
00036 #endif   /* NB_SSL_SUPPORTED */
00037 #endif   // _SSL_SESSION_ID_H_
```

24.248 SslSocket.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SSL_SOCKET_H_
00006 #define _SSL_SOCKET_H_
00007
00008 #ifdef NB_SSL_SUPPORTED
00009
00010 #include <buffers.h>
00011 #include <netinterface.h>
00012
00013 #include <crypto/wolfssl/ssl.h>
00014 #include <crypto/NetBurner/CryptoSocket.h>
00015 #include <crypto/ssl.h>
00016
00017 // Time stuff
00018 #define SHUTDOWN_RETRY_INTERVAL_IMMEDIATE \
00019     (1)   // Because of our asynchronous connections, we will generally have to try at least twice to
      completely
```

```
00020           // shutdown with WolfSSL. However, the time between these calls can be extremely short.
00021 #define SHUTDOWN_RETRY_INTERVAL (TICKS_PER_SECOND)
00022 #define CONNECT_RETRY_INTERVAL (TICKS_PER_SECOND)
00023
00024 class SslSocket : public CryptoSocket
00025 {
00026    public:
00027     SslSocket();
00028     ~SslSocket();
00029
00030     uint16_t InitSocket(int tcpFd, WOLFSSL_CTX *ctx, const char *commonName, uint32_t sockFlags = 0);
00031
00032     void CleanupTcpClose();
00033
00034     // Called from wolf ssl loop mostly to handle notifications and
00035     // time out stuff for all async process except read
00036     void ProcessAsyncStuff() override;
00037     int CheckSocketRecv() override;
00038     uint32_t GetCheckableIndexFromSocket() override;
00039     uint32_t SocketRead(char *buf, uint32_t len) override;
00040     uint32_t SocketWrite(const char *buf, uint32_t len) override;
00041     SocketHasData_t SocketHasData() override;
00042
00043     // Need to add a GetNewSocket() for each derived type
00044     static SslSocket *GetSocketFromIndex(uint32_t index);
00045     static SslSocket *GetSocketFromTcpFd(int tcpFd);
00046     static SslSocket *GetSocketFromCryptoFd(int cryptoFd);
00047     static SslSocket *GetNewSocket(int tcpFd, WOLFSSL_CTX *ctx, const char *commonName, uint32_t
      sockFlags = 0);
00048
00049     inline WOLFSSL *GetWolfSsl() { return (WOLFSSL *)m_wolfCtx; }
00050
00051    private:
00052     void WriteUnwrittenData() override;
00053     virtual bool PendOnHandshake() { return (m_error == SSL_ERROR_HANDSHAKE_INCOMPLETE); }
00054 };
00055
00056 extern SslSocket gSslSockets[];
00057
00058 #endif /* NB_SSL_SUPPORTED */
00059 #endif
```

24.249 E70_RAM/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
```

```
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
```

```
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
```

```
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* ---------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* ---------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
```

```
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
```

```
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* --------------------------------------------------------------------------- */
00469 /* Custom Standard Lib */
00470 /* --------------------------------------------------------------------------- */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
```

```
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
```

```
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.250 IC_D20/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
```

```
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
```

```
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
```

```
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
```

```
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
```

```
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* --------------------------------------------------------------------------- */
00469 /* Custom Standard Lib */
00470 /* --------------------------------------------------------------------------- */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
```

```
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
```

```
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA   // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128      // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256      // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH    // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.251 MOD5441X/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030 #include <endian.h>
00031
00032 #define TARGET_EMBEDDED
00033
00034 /* ------------------------------------------------------------------------ */
00035 /* Platform */
```

```
00036 /* ------------------------------------------------------------------------ */
00037 #define BIG_ENDIAN_ORDER
00038 #define WOLFSSL_GENERAL_ALIGNMENT 4
00039 #define SIZEOF_LONG_LONG 8
00040 #if 0
00041     #define NO_64BIT /* disable use of 64-bit variables */
00042 #endif
00043
00044 #ifdef TARGET_EMBEDDED
00045     /* disable mutex locking */
00046     // #define SINGLE_THREADED
00047
00048     /* reduce stack use. For variables over 100 bytes allocate from heap */
00049     #define WOLFSSL_SMALL_STACK
00050
00051     /* disable the built-in socket support and use the IO callbacks.
00052      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00053      */
00054     #define WOLFSSL_USER_IO
00055 #endif
00056
00057 #define WOLFSSL_32BIT_MILLI_TIME
00058
00059 /* ------------------------------------------------------------------------ */
00060 /* Math Configuration */
00061 /* ------------------------------------------------------------------------ */
00062 #define ULLONG_MAX 18446744073709551615ULL
00063 #define SP_WORD_SIZE 32
00064
00065 #undef USE_FAST_MATH
00066 #if 0
00067     /* fast math (tfmc.) (stack based and timing resistant) */
00068     #define USE_FAST_MATH
00069     #define TFM_TIMING_RESISTANT
00070 #else
00071     /* normal heap based integer.c (not timing resistant) */
00072 #endif
00073
00074 /* Wolf Single Precision Math */
00075 #undef WOLFSSL_SP
00076 #if 1
00077     #define WOLFSSL_SP
00078     #define WOLFSSL_HAVE_SP_RSA
00079     #define WOLFSSL_HAVE_SP_DH
00080     #define WOLFSSL_HAVE_SP_ECC
00081     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00082     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00083
00084     #define WOLFSSL_SP_CACHE_RESISTANT
00085     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00086     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00087
00088     //#define WOLFSSL_SP_NO_MALLOC
00089     // #define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00090
00091     // #define WOLFSSL_SP_SMALL
00092     // #define WOLFSSL_SP_ASM
00093
00094     // #define WOLFSSL_SP_LARGE_CODE
00095
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     //#define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
```

```
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
```

```
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
```

```
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
```

```
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     // #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* --------------------------------------------------------------------------- */
00469 /* Custom Standard Lib */
00470 /* --------------------------------------------------------------------------- */
```

```
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
```

```
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.252 MODM7AE70/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
```

```
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
```

```
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
```

```
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
```

```
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
```

```
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 1
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
```

```
00468 /* ------------------------------------------------------------------------ */
00469 /* Custom Standard Lib */
00470 /* ------------------------------------------------------------------------ */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
```

```
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA   // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128      // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256      // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH    // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.253 MODRT1171/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
```

```
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
```

```
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
```

```
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
```

```
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
```

```
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
```

```
00465 #endif
00466
00467
00468 /* ------------------------------------------------------------------------ */
00469 /* Custom Standard Lib */
00470 /* ------------------------------------------------------------------------ */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
```

```
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE    // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE    // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.254 MON_RT10xx/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
```

```
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
```

```
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
```

```
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
```

```
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 /* Override Memory API's */
00364 #ifdef SSL_CUSTOM_MALLOC
00365     #define XMALLOC_OVERRIDE
00366
00367     /* prototypes for user heap override functions */
00368     /* Note: Realloc only required for normal math */
00369     #include <stddef.h>  /* for size_t */
00370
00371     extern void* NBMalloc(size_t n);
00372     extern void NBFree(void *p);
00373     extern void* NBRealloc(void *p, size_t n);
00374
```

```
00375     #define XMALLOC(n, h, t)     NBMalloc(n)
00376     #define XFREE(p, h, t)       NBFree(p)
00377     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00378
00379     // Platform specific fastest memory location
00380     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00381     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00382     #elif SSL_CUSTOM_MALLOC == 2
00383     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00384     #elif SSL_CUSTOM_MALLOC == 3
00385     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00386     #elif SSL_CUSTOM_MALLOC == 4
00387     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00388     #endif
00389 #endif
00390
00391 #if 0
00392     /* Static memory requires fast math */
00393     #define WOLFSSL_STATIC_MEMORY
00394
00395     /* Disable fallback malloc/free */
00396     #define WOLFSSL_NO_MALLOC
00397     #if 1
00398         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00399     #endif
00400 #endif
00401
00402 /* Memory callbacks */
00403 #if 0
00404     #undef  USE_WOLFSSL_MEMORY
00405     #define USE_WOLFSSL_MEMORY
00406
00407     /* Use this to measure / print heap usage */
00408     #if 1
00409         #define WOLFSSL_TRACK_MEMORY
00410         #define WOLFSSL_DEBUG_MEMORY
00411     #endif
00412 #else
00413     #ifndef WOLFSSL_STATIC_MEMORY
00414         #define NO_WOLFSSL_MEMORY
00415         /* Otherwise we will use stdlib malloc, free and realloc */
00416     #endif
00417 #endif
00418
00419
00420 /* --------------------------------------------------------------------------- */
00421 /* Port */
00422 /* --------------------------------------------------------------------------- */
00423
00424 /* Override Current Time */
00425 #if 1
00426     /* Allows custom "custom_time()" function to be used for benchmark */
00427     #define WOLFSSL_USER_CURRTIME
00428     // #define WOLFSSL_GMTIME
00429     #define USER_TICKS
00430     #include <time.h>
00431     extern unsigned long my_time(time_t *timer);
00432     #define XTIME my_time
00433 #endif
00434
00435
00436 /* --------------------------------------------------------------------------- */
00437 /* RNG */
00438 /* --------------------------------------------------------------------------- */
00439
00440 /* Choose RNG method */
00441 #if 1
00442     /* Custom Seed Source */
00443     #if 1
00444         /* Size of returned HW RNG value */
00445         #define CUSTOM_RAND_TYPE      unsigned int
00446         extern unsigned int my_rng_seed_gen(void);
00447         #undef  CUSTOM_RAND_GENERATE
00448         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00449     #endif
00450
00451     // NetBurner specific define for enabling hardware random number generation for M7
00452     #define GATHER_RANDOM_USE_HW
00453
00454     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00455     /* P-RNG + HW RNG (P-RNG is ~8K) */
00456     #undef  HAVE_HASHDRBG
00457     #define HAVE_HASHDRBG
00458 #else
00459     #undef  WC_NO_HASHDRBG
00460     #define WC_NO_HASHDRBG
00461
```

```
00462     /* Bypass P-RNG and use only HW RNG */
00463     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00464     #undef  CUSTOM_RAND_GENERATE_BLOCK
00465     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00466 #endif
00467
00468
00469 /* ---------------------------------------------------------------------------- */
00470 /* Custom Standard Lib */
00471 /* ---------------------------------------------------------------------------- */
00472 /* Allows override of all standard library functions */
00473 #undef STRING_USER
00474 #if 0
00475     #define STRING_USER
00476
00477     #include <string.h>
00478
00479     #define USE_WOLF_STRSEP
00480     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00481
00482     #define USE_WOLF_STRTOK
00483     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00484
00485     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00486
00487     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00488     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00489     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00490     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00491
00492     #define XSTRLEN(s1)       strlen((s1))
00493     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00494     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00495
00496     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00497     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00498     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00499
00500     #define XSNPRINTF snprintf
00501 #endif
00502
00503
00504
00505 /* ---------------------------------------------------------------------------- */
00506 /* Enable Features */
00507 /* ---------------------------------------------------------------------------- */
00508
00509 #define WOLFSSL_TLS13
00510 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00511 #define HAVE_TLS_EXTENSIONS
00512 #define HAVE_SUPPORTED_CURVES
00513 #define WOLFSSL_BASE64_ENCODE
00514
00515
00516 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00517 #define KEEP_PEER_CERT
00518 //#define HAVE_COMP_KEY
00519
00520 /* TLS Session Cache */
00521 #if 1
00522     #define SMALL_SESSION_CACHE
00523 #else
00524     #define NO_SESSION_CACHE
00525 #endif
00526
00527 #define HAVE_ONE_TIME_AUTH
00528 #define HAVE_SNI
00529 #define HAVE_SESSION_TICKET
00530
00531 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00532 #define WOLFSSL_TICKET_NONCE_MALLOC
00533
00534 /* ---------------------------------------------------------------------------- */
00535 /* Disable Features */
00536 /* ---------------------------------------------------------------------------- */
00537 //#define NO_WOLFSSL_SERVER
00538 //#define NO_WOLFSSL_CLIENT
00539 //#define NO_CRYPT_TEST
00540 //#define NO_CRYPT_BENCHMARK
00541 //#define WOLFCRYPT_ONLY
00542
00543 /* In-lining of misc.c functions */
00544 /* If defined, must include wolfcrypt/src/misc.c in build */
00545 /* Slower, but about 1k smaller */
00546 //#define NO_INLINE
00547
```

```
00548 #define WOLFSSL_NO_SOCK
00549 #define NO_WOLFSSL_DIR
00550
00551 #ifdef TARGET_EMBEDDED
00552     #define NO_FILESYSTEM
00553     #define NO_WRITEV
00554     #define NO_MAIN_DRIVER
00555     #define NO_DEV_RANDOM
00556 #endif
00557
00558 #define NO_OLD_TLS
00559 #define NO_PSK
00560
00561 #define NO_DSA
00562 // #define NO_RC4
00563 #define NO_MD4
00564 #define NO_PWDBASED
00565 //#define NO_CODING
00566 //#define NO_ASN_TIME
00567 //#define NO_CERTS
00568 //#define NO_SIG_WRAPPER
00569
00570 #define NO_HC128
00571 #define NO_RABBIT
00572
00573 #define WOLFSSL_IGNORE_FILE_WARN
00574
00575 #undef NO_TLS
00576
00577 // Settings made for compatibility
00578 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00579 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00580 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00581 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00582
00583 #define WOLFSSL_CERT_REQ
00584 #define WOLFSSL_CERT_GEN
00585 #define WOLFSSL_ALT_NAMES
00586 #define WOLFSSL_DER_TO_PEM
00587 #define WOLFSSL_KEY_GEN
00588
00589 #define ENABLE_ECCKEY_CREATE    // Custom define, maybe should move to predef?
00590 #define ENABLE_RSAKEY_CREATE    // Custom define, maybe should move to predef?
00591
00592     // For wolfSSH
00593     //#undef WOLFSSH_SFTP
00594     //#define WOLFSSH_SFTP
00595
00596     //#undef WOLFSSH_SCP
00597     //#define WOLFSSH_SCP
00598
00599 #undef WOLFSSH_USER_IO
00600 #define WOLFSSH_USER_IO
00601
00602 #ifdef __cplusplus
00603 }
00604 #endif
00605
00606 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.255 MON_RT11xx/user_settings.h

```
00001
00002 #ifndef USER_SETTINGS_H
00003 #define USER_SETTINGS_H
00004
00005 #include <predef.h>
00006
00007 #ifdef __cplusplus
00008 extern "C" {
00009 #endif
00010
00011 //#define DEBUG_WOLFSSL
00012 #include <endian.h>
00013 #define XHTONS(a) __htons(a)
00014
00015 //#define WOLFSSL_CALLBACKS
00016
00017 /* ------------------------------------------------------------------------ */
00018 /* Platform */
00019 /* ------------------------------------------------------------------------ */
00020 #undef  WOLFSSL_GENERAL_ALIGNMENT
00021 #define WOLFSSL_GENERAL_ALIGNMENT   4
00022
```

```
00023 #undef  SINGLE_THREADED
00024 //#define SINGLE_THREADED
00025
00026 #undef  WOLFSSL_SMALL_STACK
00027 #define WOLFSSL_SMALL_STACK
00028
00029 #undef  WOLFSSL_USER_IO
00030 #define WOLFSSL_USER_IO
00031
00032 /* ------------------------------------------------------------------------ */
00033 /* Math Configuration */
00034 /* ------------------------------------------------------------------------ */
00035 #undef  SIZEOF_LONG_LONG
00036 #define SIZEOF_LONG_LONG 8
00037
00038 #undef USE_FAST_MATH
00039 #if 1
00040     #define USE_FAST_MATH
00041
00042     #undef  TFM_TIMING_RESISTANT
00043     #define TFM_TIMING_RESISTANT
00044
00045     /* Optimizations */
00046     //#define TFM_ARM
00047 #endif
00048
00049 /* Wolf Single Precision Math */
00050 #undef WOLFSSL_SP
00051 #if 0
00052     #define WOLFSSL_SP
00053     #define WOLFSSL_SP_SMALL      /* use smaller version of code */
00054     #define WOLFSSL_HAVE_SP_RSA
00055     #define WOLFSSL_HAVE_SP_DH
00056     #define WOLFSSL_HAVE_SP_ECC
00057     #define WOLFSSL_SP_CACHE_RESISTANT
00058     //#define WOLFSSL_SP_MATH     /* only SP math - eliminates fast math code */
00059
00060     /* 64 or 32 bit version */
00061     //#define WOLFSSL_SP_ASM      /* required if using the ASM versions */
00062     //#define WOLFSSL_SP_ARM32_ASM
00063     //#define WOLFSSL_SP_ARM64_ASM
00064 #endif
00065
00066 /* ------------------------------------------------------------------------ */
00067 /* FIPS - Requires eval or license from wolfSSL */
00068 /* ------------------------------------------------------------------------ */
00069 #undef  HAVE_FIPS
00070 #if 0
00071     #define HAVE_FIPS
00072
00073     #undef  HAVE_FIPS_VERSION
00074     #define HAVE_FIPS_VERSION 2
00075
00076     #ifdef SINGLE_THREADED
00077         #undef  NO_THREAD_LS
00078         #define NO_THREAD_LS
00079     #endif
00080 #endif
00081
00082
00083 /* ------------------------------------------------------------------------ */
00084 /* Crypto */
00085 /* ------------------------------------------------------------------------ */
00086 /* RSA */
00087 #undef NO_RSA
00088 #if 1
00089     #ifdef USE_FAST_MATH
00090         /* Maximum math bits (Max RSA key bits * 2) */
00091         #undef  FP_MAX_BITS
00092         #define FP_MAX_BITS     8192
00093     #endif
00094
00095     /* half as much memory but twice as slow */
00096     #undef  RSA_LOW_MEM
00097     //#define RSA_LOW_MEM
00098
00099     /* Enables blinding mode, to prevent timing attacks */
00100     #if 1
00101         #undef  WC_RSA_BLINDING
00102         #define WC_RSA_BLINDING
00103     #else
00104         #undef  WC_NO_HARDEN
00105         #define WC_NO_HARDEN
00106     #endif
00107
00108     /* RSA PSS Support */
00109     #if 1
```

```
00110         #define WC_RSA_PSS
00111     #endif
00112
00113     #if 1
00114         #define WC_RSA_NO_PADDING
00115     #endif
00116 #else
00117     #define NO_RSA
00118 #endif
00119
00120 /* ECC */
00121 #undef HAVE_ECC
00122 #if 1
00123     #define HAVE_ECC
00124
00125     // Manually add support for curves.
00126     #undef  ECC_USER_CURVES
00127     #define ECC_USER_CURVES
00128
00129     #ifdef ECC_USER_CURVES
00130         /* Manual Curve Selection */
00131         #define HAVE_ECC192 // For WolfSSL
00132         #define HAVE_ECC224 // For WolfSSL
00133         #undef NO_ECC256
00134
00135         // Added for specific curve support for WolfSSH
00136         // To enable other curves, look at ecc_sets in wolfcrypt/src/ecc.c and add required defines.
00137         // You will also need to make adjustments to wolfSSH_ProcessBuffer() where the curveId is
      checked.
00138         #define ECC192
00139         #define ECC224
00140         #undef NO_ECC_SECP
00141         #define HAVE_ECC_SECPR2
00142         #define HAVE_ECC_SECPR3
00143
00144         #ifdef ENABLE_ECC384 // predef.h
00145             #define HAVE_ECC384 // - Disabled until we can get better performance
00146         #endif
00147
00148         #ifdef ENABLE_ECC521 // predef.h
00149             #define HAVE_ECC521 // - Disabled until we can get better performance
00150         #endif
00151
00152         #ifdef ENABLE_ED25519 // predef.h
00153             #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00154             #define HAVE_CURVE25519
00155
00156             /* Optionally use small math (less flash usage, but much slower) */
00157             //#define CURVED25519_SMALL
00158         #endif
00159     #endif
00160
00161     /* Fixed point cache (speeds repeated operations against same private key) */
00162     #undef  FP_ECC
00163     #define FP_ECC
00164     #ifdef FP_ECC
00165         /* Bits / Entries */
00166         #undef  FP_ENTRIES
00167         #define FP_ENTRIES  15
00168         #undef  FP_LUT
00169         #define FP_LUT      4
00170     #endif
00171
00172     /* Optional ECC calculation method */
00173     /* Note: doubles heap usage, but slightly faster */
00174     #undef  ECC_SHAMIR
00175     #define ECC_SHAMIR
00176
00177     /* Reduces heap usage, but slower */
00178     #undef  ECC_TIMING_RESISTANT
00179     //#define ECC_TIMING_RESISTANT // - Disabled for performance
00180
00181     /* Enable cofactor support */
00182     #ifdef HAVE_FIPS
00183         #undef  HAVE_ECC_CDH
00184         #define HAVE_ECC_CDH
00185     #endif
00186
00187     /* Validate import */
00188     #ifdef HAVE_FIPS
00189         #undef  WOLFSSL_VALIDATE_ECC_IMPORT
00190         #define WOLFSSL_VALIDATE_ECC_IMPORT
00191     #endif
00192
00193     /* Compressed Key Support */
00194     #undef  HAVE_COMP_KEY
00195     //#define HAVE_COMP_KEY
```

```
00196
00197     /* Use alternate ECC size for ECC math */
00198     #ifdef USE_FAST_MATH
00199         #ifdef NO_RSA
00200             /* Custom fastmath size if not using RSA */
00201             /* MAX = ROUND32(ECC BITS 256) + SIZE_OF_MP_DIGIT(32) */
00202             #undef  FP_MAX_BITS
00203             #define FP_MAX_BITS     (256 + 32)
00204         #else
00205             #undef  ALT_ECC_SIZE
00206             #define ALT_ECC_SIZE
00207         #endif
00208
00209         /* Speedups specific to curve */
00210         #ifndef NO_ECC256
00211             #undef  TFM_ECC256
00212             #define TFM_ECC256
00213         #endif
00214     #endif
00215 #endif
00216
00217 /* DH */
00218 #undef  NO_DH
00219 #if 1
00220     /* Use table for DH instead of -lm (math) lib dependency */
00221     #if 1
00222         #define WOLFSSL_DH_CONST
00223         #define HAVE_FFDHE_2048
00224         #define HAVE_FFDHE_4096
00225         //#define HAVE_FFDHE_6144
00226         //#define HAVE_FFDHE_8192
00227     #endif
00228
00229     #ifdef HAVE_FIPS
00230         #define WOLFSSL_VALIDATE_FFC_IMPORT
00231         #define HAVE_FFDHE_Q
00232     #endif
00233 #else
00234     #define NO_DH
00235 #endif
00236
00237
00238 /* AES */
00239 #undef NO_AES
00240 #if 1
00241     #undef  HAVE_AES_CBC
00242     #define HAVE_AES_CBC
00243
00244     #undef  HAVE_AESGCM
00245     #define HAVE_AESGCM
00246
00247     /* GCM Method: GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00248     #define GCM_SMALL
00249
00250     #undef  WOLFSSL_AES_DIRECT
00251     //#define WOLFSSL_AES_DIRECT
00252
00253     #undef  HAVE_AES_ECB
00254     //#define HAVE_AES_ECB
00255
00256     #undef  WOLFSSL_AES_COUNTER
00257     //#define WOLFSSL_AES_COUNTER
00258
00259     #undef  HAVE_AESCCM
00260     #define HAVE_AESCCM
00261 #else
00262     #define NO_AES
00263 #endif
00264
00265
00266 /* DES3 */
00267 #undef NO_DES3
00268 #if 1
00269 #else
00270     #define NO_DES3
00271 #endif
00272
00273 /* ChaCha20 / Poly1305 */
00274 //#undef HAVE_CHACHA
00275 //#undef HAVE_POLY1305
00276 //#if 0
00277     #define HAVE_CHACHA
00278     #define HAVE_POLY1305
00279
00280     /* Needed for Poly1305 */
00281     #undef  HAVE_ONE_TIME_AUTH
00282     #define HAVE_ONE_TIME_AUTH
```

```
00283 //#endif
00284
00285 /* Ed25519 / Curve25519 */
00286 //#undef HAVE_CURVE25519
00287 //#undef HAVE_ED25519
00288 //#if 0
00289     #define HAVE_CURVE25519
00290     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00291
00292     /* Optionally use small math (less flash usage, but much slower) */
00293     #if 1
00294         #define CURVED25519_SMALL
00295     #endif
00296 //#endif
00297
00298
00299 /* --------------------------------------------------------------------------- */
00300 /* Hashing */
00301 /* --------------------------------------------------------------------------- */
00302 /* Sha */
00303 //#undef WOLFSSL_STATIC_RSA
00304 #if 1
00305     /* 1k smaller, but 25% slower */
00306     //#define USE_SLOW_SHA
00307 #else
00308     #define NO_SHA
00309 #endif
00310
00311 /* Sha256 */
00312 #undef NO_SHA256
00313 #if 1
00314     /* not unrolled - ~2k smaller and ~25% slower */
00315     //#define USE_SLOW_SHA256
00316
00317     /* Sha224 */
00318     #if 0
00319         #define WOLFSSL_SHA224
00320     #endif
00321 #else
00322     #define NO_SHA256
00323 #endif
00324
00325 /* Sha512 */
00326 //#undef WOLFSSL_SHA512
00327 //#if 0
00328     #define WOLFSSL_SHA512
00329
00330     /* Sha384 */
00331     //#undef  WOLFSSL_SHA384
00332     //#if 0
00333         #define WOLFSSL_SHA384
00334     //#endif
00335
00336     /* over twice as small, but 50% slower */
00337     //#define USE_SLOW_SHA512
00338 //#endif
00339
00340 /* Sha3 */
00341 #undef WOLFSSL_SHA3
00342 #if 0
00343     #define WOLFSSL_SHA3
00344 #endif
00345
00346 /* MD5 */
00347 #undef  NO_MD5
00348 #if 0
00349
00350 #else
00351     #define NO_MD5
00352 #endif
00353
00354 /* HKDF */
00355 #undef HAVE_HKDF
00356 #if 1
00357     #define HAVE_HKDF
00358 #endif
00359
00360 /* CMAC */
00361 #undef WOLFSSL_CMAC
00362 #if 0
00363     #define WOLFSSL_CMAC
00364 #endif
00365
00366
00367 /* --------------------------------------------------------------------------- */
00368 /* Benchmark / Test */
00369 /* --------------------------------------------------------------------------- */
```

```
00370 /* Use reduced benchmark / test sizes */
00371 #undef  BENCH_EMBEDDED
00372 //#define BENCH_EMBEDDED
00373
00374 #undef  USE_CERT_BUFFERS_2048
00375 //#define USE_CERT_BUFFERS_2048
00376
00377 #undef  USE_CERT_BUFFERS_1024
00378 //#define USE_CERT_BUFFERS_1024
00379
00380 #undef  USE_CERT_BUFFERS_256
00381 //#define USE_CERT_BUFFERS_256
00382
00383
00384 /* ---------------------------------------------------------------------------- */
00385 /* Debugging */
00386 /* ---------------------------------------------------------------------------- */
00387
00388 #undef DEBUG_WOLFSSL
00389 #undef NO_ERROR_STRINGS
00390 #if 0
00391     #define DEBUG_WOLFSSL
00392 #else
00393     #if 0
00394         #define NO_ERROR_STRINGS
00395     #endif
00396 #endif
00397
00398 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00399 // #define SHOW_SECRETS
00400 // #define HAVE_SECRET_CALLBACK
00401
00402 /* ---------------------------------------------------------------------------- */
00403 /* Memory */
00404 /* ---------------------------------------------------------------------------- */
00405
00406 /* Override Memory API's */
00407 #if 0
00408     #undef  XMALLOC_OVERRIDE
00409     #define XMALLOC_OVERRIDE
00410
00411     /* prototypes for user heap override functions */
00412     /* Note: Realloc only required for normal math */
00413     #include <stddef.h>  /* for size_t */
00414     extern void *myMalloc(size_t n, void* heap, int type);
00415     extern void myFree(void *p, void* heap, int type);
00416     extern void *myRealloc(void *p, size_t n, void* heap, int type);
00417
00418     #define XMALLOC(n, h, t)     myMalloc(n, h, t)
00419     #define XFREE(p, h, t)       myFree(p, h, t)
00420     #define XREALLOC(p, n, h, t) myRealloc(p, n, h, t)
00421 #endif
00422
00423 #if 0
00424     /* Static memory requires fast math */
00425     #define WOLFSSL_STATIC_MEMORY
00426
00427     /* Disable fallback malloc/free */
00428     #define WOLFSSL_NO_MALLOC
00429     #if 1
00430         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00431     #endif
00432 #endif
00433
00434 /* Memory callbacks */
00435 #if 0
00436     #undef  USE_WOLFSSL_MEMORY
00437     #define USE_WOLFSSL_MEMORY
00438
00439     /* Use this to measure / print heap usage */
00440     #if 1
00441         #undef  WOLFSSL_TRACK_MEMORY
00442         #define WOLFSSL_TRACK_MEMORY
00443
00444         #undef  WOLFSSL_DEBUG_MEMORY
00445         #define WOLFSSL_DEBUG_MEMORY
00446     #endif
00447 #else
00448     #ifndef WOLFSSL_STATIC_MEMORY
00449         #define NO_WOLFSSL_MEMORY
00450         /* Otherwise we will use stdlib malloc, free and realloc */
00451     #endif
00452 #endif
00453
00454
00455 /* ---------------------------------------------------------------------------- */
00456 /* Port */
```

```
00457 /* ---------------------------------------------------------------------------- */
00458
00459 /* Override Current Time */
00460 /* Allows custom "custom_time()" function to be used for benchmark */
00461 #define WOLFSSL_USER_CURRTIME
00462 #define WOLFSSL_GMTIME
00463 #define USER_TICKS
00464 extern unsigned long my_time(unsigned long* timer);
00465 #define XTIME my_time
00466
00467
00468 /* ---------------------------------------------------------------------------- */
00469 /* RNG */
00470 /* ---------------------------------------------------------------------------- */
00471
00472 /* Seed Source */
00473 /* Size of returned HW RNG value */
00474 #define CUSTOM_RAND_TYPE      unsigned int
00475 extern unsigned int my_rng_seed_gen(void);
00476 #undef  CUSTOM_RAND_GENERATE
00477 #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00478
00479 // NetBurner specific define for enabling hardware random number generation for M7
00480 #define GATHER_RANDOM_USE_HW
00481
00482 /* Choose RNG method */
00483 #if 1
00484     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00485     /* P-RNG + HW RNG (P-RNG is ~8K) */
00486     #undef  HAVE_HASHDRBG
00487     //#define HAVE_HASHDRBG
00488 #else
00489     #undef  WC_NO_HASHDRBG
00490     #define WC_NO_HASHDRBG
00491
00492     /* Bypass P-RNG and use only HW RNG */
00493     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00494     #undef  CUSTOM_RAND_GENERATE_BLOCK
00495     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00496 #endif
00497
00498
00499 /* ---------------------------------------------------------------------------- */
00500 /* Custom Standard Lib */
00501 /* ---------------------------------------------------------------------------- */
00502 /* Allows override of all standard library functions */
00503 #undef STRING_USER
00504 #if 0
00505     #define STRING_USER
00506
00507     #include <string.h>
00508
00509     #undef  USE_WOLF_STRSEP
00510     #define USE_WOLF_STRSEP
00511     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00512
00513     #undef  USE_WOLF_STRTOK
00514     #define USE_WOLF_STRTOK
00515     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00516
00517     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00518
00519     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00520     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00521     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00522     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00523
00524     #define XSTRLEN(s1)       strlen((s1))
00525     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00526     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00527
00528     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00529     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00530     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00531
00532     #define XSNPRINTF snprintf
00533 #endif
00534
00535
00536
00537 /* ---------------------------------------------------------------------------- */
00538 /* Enable Features */
00539 /* ---------------------------------------------------------------------------- */
00540 #undef WOLFSSL_TLS13
00541 #if 1
00542     #define WOLFSSL_TLS13
00543 #endif
```

```
00544
00545 #undef WOLFSSL_KEY_GEN
00546 #if 1
00547     #define WOLFSSL_KEY_GEN
00548 #endif
00549
00550 #if defined(HAVE_FIPS) && !defined(WOLFSSL_KEY_GEN)
00551     #define WOLFSSL_OLD_PRIME_CHECK
00552 #endif
00553
00554 #undef  KEEP_PEER_CERT
00555 #define KEEP_PEER_CERT
00556
00557 #undef  HAVE_COMP_KEY
00558 //#define HAVE_COMP_KEY
00559
00560 #undef  HAVE_TLS_EXTENSIONS
00561 #define HAVE_TLS_EXTENSIONS
00562
00563 #undef  HAVE_SUPPORTED_CURVES
00564 #define HAVE_SUPPORTED_CURVES
00565
00566 #undef  WOLFSSL_BASE64_ENCODE
00567 #define WOLFSSL_BASE64_ENCODE
00568
00569 #define SMALL_SESSION_CACHE
00570 #define HAVE_SESSION_TICKET
00571
00572 /* ---------------------------------------------------------------------------- */
00573 /* Disable Features */
00574 /* ---------------------------------------------------------------------------- */
00575 #undef  NO_WOLFSSL_SERVER
00576 //#define NO_WOLFSSL_SERVER
00577
00578 #undef  NO_WOLFSSL_CLIENT
00579 //#define NO_WOLFSSL_CLIENT
00580
00581 #undef  NO_CRYPT_TEST
00582 //#define NO_CRYPT_TEST
00583
00584 #undef  NO_CRYPT_BENCHMARK
00585 //#define NO_CRYPT_BENCHMARK
00586
00587 #undef  WOLFCRYPT_ONLY
00588 //#define WOLFCRYPT_ONLY
00589
00590 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00591 #define WOLFSSL_TICKET_NONCE_MALLOC
00592
00593 /* In-lining of misc.c functions */
00594 /* If defined, must include wolfcrypt/src/misc.c in build */
00595 /* Slower, but about 1k smaller */
00596 #undef  NO_INLINE
00597 //#define NO_INLINE
00598
00599 #undef WOLFSSL_NO_SOCK
00600 #define WOLFSSL_NO_SOCK
00601
00602 #undef NO_WOLFSSL_DIR
00603 #define NO_WOLFSSL_DIR
00604
00605 #undef  NO_FILESYSTEM
00606 #define NO_FILESYSTEM
00607
00608 #undef  NO_WRITEV
00609 #define NO_WRITEV
00610
00611 #undef  NO_MAIN_DRIVER
00612 #define NO_MAIN_DRIVER
00613
00614 #undef  NO_DEV_RANDOM
00615 #define NO_DEV_RANDOM
00616
00617 #undef  NO_DSA
00618 #define NO_DSA
00619
00620 #undef  NO_RC4
00621 //#define NO_RC4
00622
00623 #undef  NO_OLD_TLS
00624 #define NO_OLD_TLS
00625
00626 #undef  NO_HC128
00627 #define NO_HC128
00628
00629 #undef  NO_RABBIT
```

```
00630 #define NO_RABBIT
00631
00632 #undef  NO_PSK
00633 #define NO_PSK
00634
00635 #undef  NO_MD4
00636 #define NO_MD4
00637
00638 #undef  NO_PWDBASED
00639 #define NO_PWDBASED
00640
00641 #undef  NO_CODING
00642 //#define NO_CODING
00643
00644 #undef  NO_ASN_TIME
00645 //#define NO_ASN_TIME
00646
00647 #undef  NO_CERTS
00648 //#define NO_CERTS
00649
00650 #undef  NO_SIG_WRAPPER
00651 //#define NO_SIG_WRAPPER
00652
00653 #undef NO_TLS
00654
00655 // Settings made for compatibility
00656 #define WOLFSSL_STATIC_RSA // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00657 #define WOLFSSL_AES_128 // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00658 #define WOLFSSL_AES_256 // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00659 #define WOLFSSL_STATIC_DH // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00660
00661 #define WOLFSSL_CERT_REQ
00662 #define WOLFSSL_CERT_GEN
00663 #define WOLFSSL_ALT_NAMES
00664 #define WOLFSSL_DER_TO_PEM
00665 #define WOLFSSL_KEY_GEN
00666
00667 #define ENABLE_ECCKEY_CREATE // Custom define, maybe should move to predef?
00668 #define ENABLE_RSAKEY_CREATE // Custom define, maybe should move to predef?
00669
00670 // For wolfSSH
00671 //#undef WOLFSSH_SFTP
00672 //#define WOLFSSH_SFTP
00673
00674 //#undef WOLFSSH_SCP
00675 //#define WOLFSSH_SCP
00676
00677 #undef  WOLFSSH_USER_IO
00678 #define WOLFSSH_USER_IO
00679
00680 #ifdef __cplusplus
00681 }
00682 #endif
00683
00684 #endif /* WOLFSSL_USER_SETTINGS_H */
00685
```

24.256 MON_SAME70/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
```

```
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ---------------------------------------------------------------------- */
00034 /* Platform */
00035 /* ---------------------------------------------------------------------- */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ---------------------------------------------------------------------- */
00058 /* Math Configuration */
00059 /* ---------------------------------------------------------------------- */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ---------------------------------------------------------------------- */
00105 /* Crypto */
00106 /* ---------------------------------------------------------------------- */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
```

```
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
```

```
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* ---------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* ---------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
```

```
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
```

```
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
```

```
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* --------------------------------------------------------------------------- */
00469 /* Custom Standard Lib */
00470 /* --------------------------------------------------------------------------- */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* --------------------------------------------------------------------------- */
00505 /* Enable Features */
00506 /* --------------------------------------------------------------------------- */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* --------------------------------------------------------------------------- */
00534 /* Disable Features */
00535 /* --------------------------------------------------------------------------- */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
```

```
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.257 NANO54415/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
```

```
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030 #include <endian.h>
00031
00032 #define TARGET_EMBEDDED
00033
00034 /* ---------------------------------------------------------------------- */
00035 /* Platform */
00036 /* ---------------------------------------------------------------------- */
00037 #define BIG_ENDIAN_ORDER
00038 #define WOLFSSL_GENERAL_ALIGNMENT 4
00039 #define SIZEOF_LONG_LONG 8
00040 #if 0
00041     #define NO_64BIT /* disable use of 64-bit variables */
00042 #endif
00043
00044 #ifdef TARGET_EMBEDDED
00045     /* disable mutex locking */
00046     // #define SINGLE_THREADED
00047
00048     /* reduce stack use. For variables over 100 bytes allocate from heap */
00049     #define WOLFSSL_SMALL_STACK
00050
00051     /* disable the built-in socket support and use the IO callbacks.
00052      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00053      */
00054     #define WOLFSSL_USER_IO
00055 #endif
00056
00057 #define WOLFSSL_32BIT_MILLI_TIME
00058
00059 /* ---------------------------------------------------------------------- */
00060 /* Math Configuration */
00061 /* ---------------------------------------------------------------------- */
00062 #define ULLONG_MAX 18446744073709551615ULL
00063 #define SP_WORD_SIZE 32
00064
00065 #undef USE_FAST_MATH
00066 #if 0
00067     /* fast math (tfmc.) (stack based and timing resistant) */
00068     #define USE_FAST_MATH
00069     #define TFM_TIMING_RESISTANT
00070 #else
00071     /* normal heap based integer.c (not timing resistant) */
00072 #endif
00073
00074 /* Wolf Single Precision Math */
00075 #undef WOLFSSL_SP
00076 #if 1
00077     #define WOLFSSL_SP
00078     #define WOLFSSL_HAVE_SP_RSA
00079     #define WOLFSSL_HAVE_SP_DH
00080     #define WOLFSSL_HAVE_SP_ECC
00081     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00082     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00083
00084     #define WOLFSSL_SP_CACHE_RESISTANT
00085     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00086     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00087
00088     //#define WOLFSSL_SP_NO_MALLOC
00089     // #define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00090
00091     // #define WOLFSSL_SP_SMALL
00092     // #define WOLFSSL_SP_ASM
00093
00094     // #define WOLFSSL_SP_LARGE_CODE
00095
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     //#define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ---------------------------------------------------------------------- */
00105 /* Crypto */
00106 /* ---------------------------------------------------------------------- */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
```

```
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
```

```
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
```

```
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
```

```
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     // #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
```

```
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* ---------------------------------------------------------------------------- */
00469 /* Custom Standard Lib */
00470 /* ---------------------------------------------------------------------------- */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ---------------------------------------------------------------------------- */
00505 /* Enable Features */
00506 /* ---------------------------------------------------------------------------- */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ---------------------------------------------------------------------------- */
00534 /* Disable Features */
00535 /* ---------------------------------------------------------------------------- */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
```

```
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA   // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128      // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256      // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH    // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.258 RT10XX_RAM/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
```

```
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100    //#define WOLFSSL_SP_ARM_THUMB_ASM
00101    #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
```

```
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
```

```
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
```

```
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
```

```
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
```

```
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* ------------------------------------------------------------------------ */
00469 /* Custom Standard Lib */
00470 /* ------------------------------------------------------------------------ */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
```

```
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.259 SB800EX/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
```

```
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030 #include <endian.h>
00031
00032 #define TARGET_EMBEDDED
00033
00034 /* ---------------------------------------------------------------------------- */
00035 /* Platform */
00036 /* ---------------------------------------------------------------------------- */
00037 #define BIG_ENDIAN_ORDER
00038 #define WOLFSSL_GENERAL_ALIGNMENT 4
00039 #define SIZEOF_LONG_LONG 8
00040 #if 0
00041     #define NO_64BIT /* disable use of 64-bit variables */
00042 #endif
00043
00044 #ifdef TARGET_EMBEDDED
00045     /* disable mutex locking */
00046     // #define SINGLE_THREADED
00047
00048     /* reduce stack use. For variables over 100 bytes allocate from heap */
00049     #define WOLFSSL_SMALL_STACK
00050
00051     /* disable the built-in socket support and use the IO callbacks.
00052      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00053      */
00054     #define WOLFSSL_USER_IO
00055 #endif
00056
00057 #define WOLFSSL_32BIT_MILLI_TIME
00058
00059 /* ---------------------------------------------------------------------------- */
00060 /* Math Configuration */
00061 /* ---------------------------------------------------------------------------- */
00062 #define ULLONG_MAX 18446744073709551615ULL
00063 #define SP_WORD_SIZE 32
00064
00065 #undef USE_FAST_MATH
00066 #if 0
00067     /* fast math (tfmc.) (stack based and timing resistant) */
00068     #define USE_FAST_MATH
00069     #define TFM_TIMING_RESISTANT
00070 #else
00071     /* normal heap based integer.c (not timing resistant) */
00072 #endif
00073
00074 /* Wolf Single Precision Math */
00075 #undef WOLFSSL_SP
00076 #if 1
00077     #define WOLFSSL_SP
00078     #define WOLFSSL_HAVE_SP_RSA
00079     #define WOLFSSL_HAVE_SP_DH
00080     #define WOLFSSL_HAVE_SP_ECC
00081     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00082     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00083
00084     #define WOLFSSL_SP_CACHE_RESISTANT
00085     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00086     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00087
00088     //#define WOLFSSL_SP_NO_MALLOC
00089     // #define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00090
00091     // #define WOLFSSL_SP_SMALL
00092     // #define WOLFSSL_SP_ASM
00093
00094     // #define WOLFSSL_SP_LARGE_CODE
00095
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     //#define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
```

```
00104 /* ------------------------------------------------------------------------ */
00105 /* Crypto */
00106 /* ------------------------------------------------------------------------ */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
```

```
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
```

```
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* ---------------------------------------------------------------------------- */
00361
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
```

```
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     // #define GATHER_RANDOM_USE_HW
```

```
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* ------------------------------------------------------------------------ */
00469 /* Custom Standard Lib */
00470 /* ------------------------------------------------------------------------ */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
00535 /* ------------------------------------------------------------------------ */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
```

```
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA   // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128      // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256      // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH    // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.260 SBE70LC/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ------------------------------------------------------------------------ */
00034 /* Platform */
00035 /* ------------------------------------------------------------------------ */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ------------------------------------------------------------------------ */
00058 /* Math Configuration */
00059 /* ------------------------------------------------------------------------ */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100    //#define WOLFSSL_SP_ARM_THUMB_ASM
```

```
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ---------------------------------------------------------------------- */
00105 /* Crypto */
00106 /* ---------------------------------------------------------------------- */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
```

```
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* --------------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* --------------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
```

```
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* --------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* --------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* --------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* --------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* --------------------------------------------------------------------------- */
00359 /* Memory */
00360 /* --------------------------------------------------------------------------- */
00361
```

```
00362 /* Override Memory API's */
00363 #ifdef SSL_CUSTOM_MALLOC
00364     #define XMALLOC_OVERRIDE
00365
00366     /* prototypes for user heap override functions */
00367     /* Note: Realloc only required for normal math */
00368     #include <stddef.h>  /* for size_t */
00369
00370     extern void* NBMalloc(size_t n);
00371     extern void NBFree(void *p);
00372     extern void* NBRealloc(void *p, size_t n);
00373
00374     #define XMALLOC(n, h, t)     NBMalloc(n)
00375     #define XFREE(p, h, t)       NBFree(p)
00376     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00377
00378     // Platform specific fastest memory location
00379     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00380     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00381     #elif SSL_CUSTOM_MALLOC == 2
00382     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00383     #elif SSL_CUSTOM_MALLOC == 3
00384     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00385     #elif SSL_CUSTOM_MALLOC == 4
00386     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00387     #endif
00388 #endif
00389
00390 #if 0
00391     /* Static memory requires fast math */
00392     #define WOLFSSL_STATIC_MEMORY
00393
00394     /* Disable fallback malloc/free */
00395     #define WOLFSSL_NO_MALLOC
00396     #if 1
00397         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00398     #endif
00399 #endif
00400
00401 /* Memory callbacks */
00402 #if 0
00403     #undef  USE_WOLFSSL_MEMORY
00404     #define USE_WOLFSSL_MEMORY
00405
00406     /* Use this to measure / print heap usage */
00407     #if 1
00408         #define WOLFSSL_TRACK_MEMORY
00409         #define WOLFSSL_DEBUG_MEMORY
00410     #endif
00411 #else
00412     #ifndef WOLFSSL_STATIC_MEMORY
00413         #define NO_WOLFSSL_MEMORY
00414         /* Otherwise we will use stdlib malloc, free and realloc */
00415     #endif
00416 #endif
00417
00418
00419 /* --------------------------------------------------------------------------- */
00420 /* Port */
00421 /* --------------------------------------------------------------------------- */
00422
00423 /* Override Current Time */
00424 #if 1
00425     /* Allows custom "custom_time()" function to be used for benchmark */
00426     #define WOLFSSL_USER_CURRTIME
00427     // #define WOLFSSL_GMTIME
00428     #define USER_TICKS
00429     #include <time.h>
00430     extern unsigned long my_time(time_t *timer);
00431     #define XTIME my_time
00432 #endif
00433
00434
00435 /* --------------------------------------------------------------------------- */
00436 /* RNG */
00437 /* --------------------------------------------------------------------------- */
00438
00439 /* Choose RNG method */
00440 #if 1
00441     /* Custom Seed Source */
00442     #if 1
00443         /* Size of returned HW RNG value */
00444         #define CUSTOM_RAND_TYPE      unsigned int
00445         extern unsigned int my_rng_seed_gen(void);
00446         #undef  CUSTOM_RAND_GENERATE
00447         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00448     #endif
```

```
00449
00450     // NetBurner specific define for enabling hardware random number generation for M7
00451     #define GATHER_RANDOM_USE_HW
00452
00453     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00454     /* P-RNG + HW RNG (P-RNG is ~8K) */
00455     #undef  HAVE_HASHDRBG
00456     #define HAVE_HASHDRBG
00457 #else
00458     #undef  WC_NO_HASHDRBG
00459     #define WC_NO_HASHDRBG
00460
00461     /* Bypass P-RNG and use only HW RNG */
00462     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00463     #undef  CUSTOM_RAND_GENERATE_BLOCK
00464     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00465 #endif
00466
00467
00468 /* ------------------------------------------------------------------------ */
00469 /* Custom Standard Lib */
00470 /* ------------------------------------------------------------------------ */
00471 /* Allows override of all standard library functions */
00472 #undef STRING_USER
00473 #if 0
00474     #define STRING_USER
00475
00476     #include <string.h>
00477
00478     #define USE_WOLF_STRSEP
00479     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00480
00481     #define USE_WOLF_STRTOK
00482     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00483
00484     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00485
00486     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00487     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00488     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00489     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00490
00491     #define XSTRLEN(s1)       strlen((s1))
00492     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00493     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00494
00495     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00496     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00497     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00498
00499     #define XSNPRINTF snprintf
00500 #endif
00501
00502
00503
00504 /* ------------------------------------------------------------------------ */
00505 /* Enable Features */
00506 /* ------------------------------------------------------------------------ */
00507
00508 #define WOLFSSL_TLS13
00509 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00510 #define HAVE_TLS_EXTENSIONS
00511 #define HAVE_SUPPORTED_CURVES
00512 #define WOLFSSL_BASE64_ENCODE
00513
00514
00515 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00516 #define KEEP_PEER_CERT
00517 //#define HAVE_COMP_KEY
00518
00519 /* TLS Session Cache */
00520 #if 1
00521     #define SMALL_SESSION_CACHE
00522 #else
00523     #define NO_SESSION_CACHE
00524 #endif
00525
00526 #define HAVE_ONE_TIME_AUTH
00527 #define HAVE_SNI
00528 #define HAVE_SESSION_TICKET
00529
00530 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
00531 #define WOLFSSL_TICKET_NONCE_MALLOC
00532
00533 /* ------------------------------------------------------------------------ */
00534 /* Disable Features */
```

```
00535 /* ---------------------------------------------------------------------------- */
00536 //#define NO_WOLFSSL_SERVER
00537 //#define NO_WOLFSSL_CLIENT
00538 //#define NO_CRYPT_TEST
00539 //#define NO_CRYPT_BENCHMARK
00540 //#define WOLFCRYPT_ONLY
00541
00542 /* In-lining of misc.c functions */
00543 /* If defined, must include wolfcrypt/src/misc.c in build */
00544 /* Slower, but about 1k smaller */
00545 //#define NO_INLINE
00546
00547 #define WOLFSSL_NO_SOCK
00548 #define NO_WOLFSSL_DIR
00549
00550 #ifdef TARGET_EMBEDDED
00551     #define NO_FILESYSTEM
00552     #define NO_WRITEV
00553     #define NO_MAIN_DRIVER
00554     #define NO_DEV_RANDOM
00555 #endif
00556
00557 #define NO_OLD_TLS
00558 #define NO_PSK
00559
00560 #define NO_DSA
00561 // #define NO_RC4
00562 #define NO_MD4
00563 #define NO_PWDBASED
00564 //#define NO_CODING
00565 //#define NO_ASN_TIME
00566 //#define NO_CERTS
00567 //#define NO_SIG_WRAPPER
00568
00569 #define NO_HC128
00570 #define NO_RABBIT
00571
00572 #define WOLFSSL_IGNORE_FILE_WARN
00573
00574 #undef NO_TLS
00575
00576 // Settings made for compatibility
00577 #define WOLFSSL_STATIC_RSA   // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00578 #define WOLFSSL_AES_128      // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00579 #define WOLFSSL_AES_256      // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00580 #define WOLFSSL_STATIC_DH    // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00581
00582 #define WOLFSSL_CERT_REQ
00583 #define WOLFSSL_CERT_GEN
00584 #define WOLFSSL_ALT_NAMES
00585 #define WOLFSSL_DER_TO_PEM
00586 #define WOLFSSL_KEY_GEN
00587
00588 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00589 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00590
00591     // For wolfSSH
00592     //#undef WOLFSSH_SFTP
00593     //#define WOLFSSH_SFTP
00594
00595     //#undef WOLFSSH_SCP
00596     //#define WOLFSSH_SCP
00597
00598 #undef WOLFSSH_USER_IO
00599 #define WOLFSSH_USER_IO
00600
00601 #ifdef __cplusplus
00602 }
00603 #endif
00604
00605 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.261 SOMRT1061/user_settings.h

```
00001 /* user_settings_template.h
00002  *
00003  * Copyright (C) 2006-2023 wolfSSL Inc.
00004  *
00005  * This file is part of wolfSSL.
00006  *
00007  * wolfSSL is free software; you can redistribute it and/or modify
00008  * it under the terms of the GNU General Public License as published by
00009  * the Free Software Foundation; either version 2 of the License, or
00010  * (at your option) any later version.
```

```
00011  *
00012  * wolfSSL is distributed in the hope that it will be useful,
00013  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00015  * GNU General Public License for more details.
00016  *
00017  * You should have received a copy of the GNU General Public License
00018  * along with this program; if not, write to the Free Software
00019  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00020  */
00021
00022 #ifndef WOLFSSL_USER_SETTINGS_H
00023 #define WOLFSSL_USER_SETTINGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 #include <predef.h>
00030
00031 #define TARGET_EMBEDDED
00032
00033 /* ---------------------------------------------------------------------------- */
00034 /* Platform */
00035 /* ---------------------------------------------------------------------------- */
00036 #define WOLFSSL_GENERAL_ALIGNMENT 4
00037 #define SIZEOF_LONG_LONG 8
00038 #if 0
00039     #define NO_64BIT /* disable use of 64-bit variables */
00040 #endif
00041
00042 #ifdef TARGET_EMBEDDED
00043     /* disable mutex locking */
00044     // #define SINGLE_THREADED
00045
00046     /* reduce stack use. For variables over 100 bytes allocate from heap */
00047     #define WOLFSSL_SMALL_STACK
00048
00049     /* disable the built-in socket support and use the IO callbacks.
00050      * Set with wolfSSL_CTX_SetIORecv/wolfSSL_CTX_SetIOSend
00051      */
00052     #define WOLFSSL_USER_IO
00053 #endif
00054
00055 #define WOLFSSL_32BIT_MILLI_TIME
00056
00057 /* ---------------------------------------------------------------------------- */
00058 /* Math Configuration */
00059 /* ---------------------------------------------------------------------------- */
00060 #define ULLONG_MAX 18446744073709551615ULL
00061 #define SP_WORD_SIZE 32
00062
00063 #undef USE_FAST_MATH
00064 #if 0
00065     /* fast math (tfmc.) (stack based and timing resistant) */
00066     #define USE_FAST_MATH
00067     #define TFM_TIMING_RESISTANT
00068 #else
00069     /* normal heap based integer.c (not timing resistant) */
00070 #endif
00071
00072 /* Wolf Single Precision Math */
00073 #undef WOLFSSL_SP
00074 #if 1
00075     #define WOLFSSL_SP
00076     #define WOLFSSL_HAVE_SP_RSA
00077     #define WOLFSSL_HAVE_SP_DH
00078     #define WOLFSSL_HAVE_SP_ECC
00079     //#define WOLFSSL_SP_4096 /* Enable RSA/RH 4096-bit support */
00080     #define WOLFSSL_SP_384 /* Enable ECC 384-bit SECP384R1 support */
00081
00082     #define WOLFSSL_SP_CACHE_RESISTANT
00083     // #define WOLFSSL_SP_MATH     /* only SP math - disables integer.c/tfm.c */
00084     #define WOLFSSL_SP_MATH_ALL /* use SP math for all key sizes and curves */
00085
00086     //#define WOLFSSL_SP_NO_MALLOC
00087     //#define WOLFSSL_SP_DIV_32 /* do not use 64-bit divides */
00088
00089     #ifdef TARGET_EMBEDDED
00090         /* use smaller version of code */
00091         #define WOLFSSL_SP_SMALL
00092     #else
00093         /* SP Assembly Speedups - specific to chip type */
00094         #define WOLFSSL_SP_ASM
00095     #endif
00096     //#define WOLFSSL_SP_X86_64
00097     //#define WOLFSSL_SP_X86
```

```
00098     //#define WOLFSSL_SP_ARM32_ASM
00099     //#define WOLFSSL_SP_ARM64_ASM
00100     //#define WOLFSSL_SP_ARM_THUMB_ASM
00101     #define WOLFSSL_SP_ARM_CORTEX_M_ASM
00102 #endif
00103
00104 /* ---------------------------------------------------------------------- */
00105 /* Crypto */
00106 /* ---------------------------------------------------------------------- */
00107 /* RSA */
00108 #undef NO_RSA
00109 #if 1
00110     #ifdef USE_FAST_MATH
00111         /* Maximum math bits (Max RSA key bits * 2) */
00112         #define FP_MAX_BITS 4096
00113     #endif
00114
00115     /* half as much memory but twice as slow */
00116     //#define RSA_LOW_MEM
00117
00118     /* Enables blinding mode, to prevent timing attacks */
00119     #define WC_RSA_BLINDING
00120
00121     /* RSA PSS Support */
00122     #define WC_RSA_PSS
00123 #else
00124     #define NO_RSA
00125 #endif
00126
00127 /* DH */
00128 #undef  NO_DH
00129 #if 1
00130     /* Use table for DH instead of -lm (math) lib dependency */
00131     #if 1
00132         #define WOLFSSL_DH_CONST
00133         #define HAVE_FFDHE_2048
00134         //#define HAVE_FFDHE_4096
00135         //#define HAVE_FFDHE_6144
00136         //#define HAVE_FFDHE_8192
00137     #endif
00138 #else
00139     #define NO_DH
00140 #endif
00141
00142 /* ECC */
00143 #undef HAVE_ECC
00144 #if 1
00145     #define HAVE_ECC
00146
00147     /* Manually define enabled curves */
00148     #define ECC_USER_CURVES
00149
00150     #ifdef ECC_USER_CURVES
00151         /* Manual Curve Selection */
00152         // #define HAVE_ECC192
00153         // #define HAVE_ECC224
00154         #undef NO_ECC256
00155         #ifdef ENABLE_ECC384
00156         #define HAVE_ECC384
00157         #endif
00158         #ifdef ENABLE_ECC521
00159         // #define HAVE_ECC521
00160         #endif
00161     #endif
00162
00163     /* Fixed point cache (speeds repeated operations against same private key) */
00164     #define FP_ECC
00165     #ifdef FP_ECC
00166         /* Bits / Entries */
00167         #define FP_ENTRIES  15
00168         #define FP_LUT      4
00169     #endif
00170
00171     /* Optional ECC calculation method */
00172     /* Note: doubles heap usage, but slightly faster */
00173     #define ECC_SHAMIR
00174
00175     /* Reduces heap usage, but slower */
00176     // #define ECC_TIMING_RESISTANT
00177
00178     /* Compressed ECC Key Support */
00179     //#define HAVE_COMP_KEY
00180
00181     /* Use alternate ECC size for ECC math */
00182     #ifdef USE_FAST_MATH
00183         /* MAX ECC BITS = ROUND8(MAX ECC) * 2 */
00184         #if defined(NO_RSA) && defined(NO_DH)
```

```
00185             /* Custom fastmath size if not using RSA/DH */
00186             #define FP_MAX_BITS     (256 * 2)
00187         #else
00188             /* use heap allocation for ECC points */
00189             #define ALT_ECC_SIZE
00190
00191             /* wolfSSL will compute the FP_MAX_BITS_ECC, but it can be overriden */
00192             //#define FP_MAX_BITS_ECC (256 * 2)
00193         #endif
00194
00195         /* Speedups specific to curve */
00196         #ifndef NO_ECC256
00197             #define TFM_ECC256
00198         #endif
00199     #endif
00200 #endif
00201
00202
00203 /* AES */
00204 #undef NO_AES
00205 #if 1
00206     #define HAVE_AES_CBC
00207
00208     /* GCM Method: GCM_TABLE_4BIT, GCM_SMALL, GCM_WORD32 or GCM_TABLE */
00209     #define HAVE_AESGCM
00210     #ifdef TARGET_EMBEDDED
00211         #define GCM_SMALL
00212     #else
00213         #define GCM_TABLE_4BIT
00214     #endif
00215
00216     //#define WOLFSSL_AES_DIRECT
00217     //#define HAVE_AES_ECB
00218     //#define WOLFSSL_AES_COUNTER
00219     #define HAVE_AESCCM
00220 #else
00221     #define NO_AES
00222 #endif
00223
00224
00225 /* DES3 */
00226 #undef NO_DES3
00227 #if 1
00228 #else
00229     #define NO_DES3
00230 #endif
00231
00232 /* ChaCha20 / Poly1305 */
00233 #undef HAVE_CHACHA
00234 #undef HAVE_POLY1305
00235 #if 1
00236     #define HAVE_CHACHA
00237     #define HAVE_POLY1305
00238
00239     /* Needed for Poly1305 */
00240     #define HAVE_ONE_TIME_AUTH
00241 #endif
00242
00243 /* Ed25519 / Curve25519 */
00244 #undef HAVE_CURVE25519
00245 #undef HAVE_ED25519
00246 #if 1
00247     #define HAVE_CURVE25519
00248     #define HAVE_ED25519 /* ED25519 Requires SHA512 */
00249
00250     /* Optionally use small math (less flash usage, but much slower) */
00251     #if 0
00252         #define CURVED25519_SMALL
00253     #endif
00254 #endif
00255
00256
00257 /* ---------------------------------------------------------------------- */
00258 /* Hashing */
00259 /* ---------------------------------------------------------------------- */
00260 /* Sha */
00261 #undef NO_SHA
00262 #if 1
00263     /* 1k smaller, but 25% slower */
00264     //#define USE_SLOW_SHA
00265 #else
00266     #define NO_SHA
00267 #endif
00268
00269 /* Sha256 */
00270 #undef NO_SHA256
00271 #if 1
```

```
00272     /* not unrolled - ~2k smaller and ~25% slower */
00273     //#define USE_SLOW_SHA256
00274
00275     /* Sha224 */
00276     #if 0
00277         #define WOLFSSL_SHA224
00278     #endif
00279 #else
00280     #define NO_SHA256
00281 #endif
00282
00283 /* Sha512 */
00284 #undef WOLFSSL_SHA512
00285 #if 1
00286     #define WOLFSSL_SHA512
00287
00288     /* Sha384 */
00289     #undef  WOLFSSL_SHA384
00290     #if 1
00291         #define WOLFSSL_SHA384
00292     #endif
00293
00294     /* over twice as small, but 50% slower */
00295     //#define USE_SLOW_SHA512
00296 #endif
00297
00298 /* Sha3 */
00299 #undef WOLFSSL_SHA3
00300 #if 0
00301     #define WOLFSSL_SHA3
00302 #endif
00303
00304 /* MD5 */
00305 #undef  NO_MD5
00306 #if 0
00307
00308 #else
00309     #define NO_MD5
00310 #endif
00311
00312 /* HKDF */
00313 #undef HAVE_HKDF
00314 #if 1
00315     #define HAVE_HKDF
00316 #endif
00317
00318 /* CMAC */
00319 #undef WOLFSSL_CMAC
00320 #if 0
00321     #define WOLFSSL_CMAC
00322 #endif
00323
00324
00325 /* ---------------------------------------------------------------------------- */
00326 /* Benchmark / Test */
00327 /* ---------------------------------------------------------------------------- */
00328 #ifdef TARGET_EMBEDDED
00329     /* Use reduced benchmark / test sizes */
00330     #define BENCH_EMBEDDED
00331 #endif
00332
00333 /* Use test buffers from array (not filesystem) */
00334 #ifndef NO_FILESYSTEM
00335 #define USE_CERT_BUFFERS_256
00336 #define USE_CERT_BUFFERS_2048
00337 #endif
00338
00339 /* ---------------------------------------------------------------------------- */
00340 /* Debugging */
00341 /* To enable, call wolfSSL_Debugging_ON(); where debug output is wanted      */
00342 /* ---------------------------------------------------------------------------- */
00343
00344 #undef DEBUG_WOLFSSL
00345 #undef NO_ERROR_STRINGS
00346 #if 0
00347     #define DEBUG_WOLFSSL
00348 #else
00349     #if 0
00350         #define NO_ERROR_STRINGS
00351     #endif
00352 #endif
00353
00354 // Prints out the TLS secrets to the console, allowing for decryption of the TLS stream
00355 // #define SHOW_SECRETS
00356 // #define HAVE_SECRET_CALLBACK
00357
00358 /* ---------------------------------------------------------------------------- */
```

```
00359 /* Memory */
00360 /* ------------------------------------------------------------------------ */
00361
00362 /* Override Memory API's */
00363 /* Override Memory API's */
00364 #ifdef SSL_CUSTOM_MALLOC
00365     #define XMALLOC_OVERRIDE
00366
00367     /* prototypes for user heap override functions */
00368     /* Note: Realloc only required for normal math */
00369     #include <stddef.h>  /* for size_t */
00370
00371     extern void* NBMalloc(size_t n);
00372     extern void NBFree(void *p);
00373     extern void* NBRealloc(void *p, size_t n);
00374
00375     #define XMALLOC(n, h, t)     NBMalloc(n)
00376     #define XFREE(p, h, t)       NBFree(p)
00377     #define XREALLOC(p, n, h, t) NBRealloc(p, n)
00378
00379     // Platform specific fastest memory location
00380     #if SSL_CUSTOM_MALLOC == 1 // Fastest memory on platform
00381     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00382     #elif SSL_CUSTOM_MALLOC == 2
00383     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_TCM(name,size)
00384     #elif SSL_CUSTOM_MALLOC == 3
00385     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SRAM(name,size)
00386     #elif SSL_CUSTOM_MALLOC == 4
00387     #define CREATE_MEMORY_ALLOCATOR(name,size) CREATE_MEMORY_ALLOCATOR_SDRAM(name,size)
00388     #endif
00389 #endif
00390
00391 #if 0
00392     /* Static memory requires fast math */
00393     #define WOLFSSL_STATIC_MEMORY
00394
00395     /* Disable fallback malloc/free */
00396     #define WOLFSSL_NO_MALLOC
00397     #if 1
00398         #define WOLFSSL_MALLOC_CHECK /* trap malloc failure */
00399     #endif
00400 #endif
00401
00402 /* Memory callbacks */
00403 #if 0
00404     #undef  USE_WOLFSSL_MEMORY
00405     #define USE_WOLFSSL_MEMORY
00406
00407     /* Use this to measure / print heap usage */
00408     #if 1
00409         #define WOLFSSL_TRACK_MEMORY
00410         #define WOLFSSL_DEBUG_MEMORY
00411     #endif
00412 #else
00413     #ifndef WOLFSSL_STATIC_MEMORY
00414         #define NO_WOLFSSL_MEMORY
00415         /* Otherwise we will use stdlib malloc, free and realloc */
00416     #endif
00417 #endif
00418
00419
00420 /* ------------------------------------------------------------------------ */
00421 /* Port */
00422 /* ------------------------------------------------------------------------ */
00423
00424 /* Override Current Time */
00425 #if 1
00426     /* Allows custom "custom_time()" function to be used for benchmark */
00427     #define WOLFSSL_USER_CURRTIME
00428     // #define WOLFSSL_GMTIME
00429     #define USER_TICKS
00430     #include <time.h>
00431     extern unsigned long my_time(time_t *timer);
00432     #define XTIME my_time
00433 #endif
00434
00435
00436 /* ------------------------------------------------------------------------ */
00437 /* RNG */
00438 /* ------------------------------------------------------------------------ */
00439
00440 /* Choose RNG method */
00441 #if 1
00442     /* Custom Seed Source */
00443     #if 1
00444         /* Size of returned HW RNG value */
00445         #define CUSTOM_RAND_TYPE      unsigned int
```

```
00446         extern unsigned int my_rng_seed_gen(void);
00447         #undef  CUSTOM_RAND_GENERATE
00448         #define CUSTOM_RAND_GENERATE  my_rng_seed_gen
00449     #endif
00450
00451     // NetBurner specific define for enabling hardware random number generation for M7
00452     // #define GATHER_RANDOM_USE_HW
00453
00454     /* Use built-in P-RNG (SHA256 based) with HW RNG */
00455     /* P-RNG + HW RNG (P-RNG is ~8K) */
00456     #undef  HAVE_HASHDRBG
00457     #define HAVE_HASHDRBG
00458 #else
00459     #undef  WC_NO_HASHDRBG
00460     #define WC_NO_HASHDRBG
00461
00462     /* Bypass P-RNG and use only HW RNG */
00463     extern int my_rng_gen_block(unsigned char* output, unsigned int sz);
00464     #undef  CUSTOM_RAND_GENERATE_BLOCK
00465     #define CUSTOM_RAND_GENERATE_BLOCK  my_rng_gen_block
00466 #endif
00467
00468
00469 /* ---------------------------------------------------------------------------- */
00470 /* Custom Standard Lib */
00471 /* ---------------------------------------------------------------------------- */
00472 /* Allows override of all standard library functions */
00473 #undef STRING_USER
00474 #if 0
00475     #define STRING_USER
00476
00477     #include <string.h>
00478
00479     #define USE_WOLF_STRSEP
00480     #define XSTRSEP(s1,d)     wc_strsep((s1),(d))
00481
00482     #define USE_WOLF_STRTOK
00483     #define XSTRTOK(s1,d,ptr) wc_strtok((s1),(d),(ptr))
00484
00485     #define XSTRNSTR(s1,s2,n) mystrnstr((s1),(s2),(n))
00486
00487     #define XMEMCPY(d,s,l)    memcpy((d),(s),(l))
00488     #define XMEMSET(b,c,l)    memset((b),(c),(l))
00489     #define XMEMCMP(s1,s2,n)  memcmp((s1),(s2),(n))
00490     #define XMEMMOVE(d,s,l)   memmove((d),(s),(l))
00491
00492     #define XSTRLEN(s1)       strlen((s1))
00493     #define XSTRNCPY(s1,s2,n) strncpy((s1),(s2),(n))
00494     #define XSTRSTR(s1,s2)    strstr((s1),(s2))
00495
00496     #define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
00497     #define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
00498     #define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
00499
00500     #define XSNPRINTF snprintf
00501 #endif
00502
00503
00504
00505 /* ---------------------------------------------------------------------------- */
00506 /* Enable Features */
00507 /* ---------------------------------------------------------------------------- */
00508
00509 #define WOLFSSL_TLS13
00510 #define WOLFSSL_OLD_PRIME_CHECK /* Use faster DH prime checking */
00511 #define HAVE_TLS_EXTENSIONS
00512 #define HAVE_SUPPORTED_CURVES
00513 #define WOLFSSL_BASE64_ENCODE
00514
00515
00516 #define WOLFSSL_KEY_GEN /* For RSA Key gen only */
00517 #define KEEP_PEER_CERT
00518 //#define HAVE_COMP_KEY
00519
00520 /* TLS Session Cache */
00521 #if 1
00522     #define SMALL_SESSION_CACHE
00523 #else
00524     #define NO_SESSION_CACHE
00525 #endif
00526
00527 #define HAVE_ONE_TIME_AUTH
00528 #define HAVE_SNI
00529 #define HAVE_SESSION_TICKET
00530
00531 // Allows WolfSSL to malloc the tls 1.3 ticket nonce, instead of using a static buffer. This supports
      large ticket nonces
```

```
00532 #define WOLFSSL_TICKET_NONCE_MALLOC
00533
00534 /* ---------------------------------------------------------------------- */
00535 /* Disable Features */
00536 /* ---------------------------------------------------------------------- */
00537 //#define NO_WOLFSSL_SERVER
00538 //#define NO_WOLFSSL_CLIENT
00539 //#define NO_CRYPT_TEST
00540 //#define NO_CRYPT_BENCHMARK
00541 //#define WOLFCRYPT_ONLY
00542
00543 /* In-lining of misc.c functions */
00544 /* If defined, must include wolfcrypt/src/misc.c in build */
00545 /* Slower, but about 1k smaller */
00546 //#define NO_INLINE
00547
00548 #define WOLFSSL_NO_SOCK
00549 #define NO_WOLFSSL_DIR
00550
00551 #ifdef TARGET_EMBEDDED
00552     #define NO_FILESYSTEM
00553     #define NO_WRITEV
00554     #define NO_MAIN_DRIVER
00555     #define NO_DEV_RANDOM
00556 #endif
00557
00558 #define NO_OLD_TLS
00559 #define NO_PSK
00560
00561 #define NO_DSA
00562 // #define NO_RC4
00563 #define NO_MD4
00564 #define NO_PWDBASED
00565 //#define NO_CODING
00566 //#define NO_ASN_TIME
00567 //#define NO_CERTS
00568 //#define NO_SIG_WRAPPER
00569
00570 #define NO_HC128
00571 #define NO_RABBIT
00572
00573 #define WOLFSSL_IGNORE_FILE_WARN
00574
00575 #undef NO_TLS
00576
00577 // Settings made for compatibility
00578 #define WOLFSSL_STATIC_RSA    // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA
00579 #define WOLFSSL_AES_128       // Needed to support TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA256
00580 #define WOLFSSL_AES_256       // Needed to support TLS_RSA_WITH_AES_256_CBC_SHA256
00581 #define WOLFSSL_STATIC_DH     // Needed to support TLS_ECDH_ECDSA_WITH_RC4_128_SHA
00582
00583 #define WOLFSSL_CERT_REQ
00584 #define WOLFSSL_CERT_GEN
00585 #define WOLFSSL_ALT_NAMES
00586 #define WOLFSSL_DER_TO_PEM
00587 #define WOLFSSL_KEY_GEN
00588
00589 #define ENABLE_ECCKEY_CREATE   // Custom define, maybe should move to predef?
00590 #define ENABLE_RSAKEY_CREATE   // Custom define, maybe should move to predef?
00591
00592     // For wolfSSH
00593     //#undef WOLFSSH_SFTP
00594     //#define WOLFSSH_SFTP
00595
00596     //#undef WOLFSSH_SCP
00597     //#define WOLFSSH_SCP
00598
00599 #undef WOLFSSH_USER_IO
00600 #define WOLFSSH_USER_IO
00601
00602 #ifdef __cplusplus
00603 }
00604 #endif
00605
00606 #endif /* WOLFSSL_USER_SETTINGS_H */
```

24.262 NbWolfSsh.h File Reference

NetBurner SSH API.

```
#include <nettypes.h>
#include <stdio.h>
#include <basictypes.h>
```

```
#include <ssh/wolfssh/ssh.h>
#include <ssh/NetBurner/UserAuthManager.h>
#include <ssh/NetBurner/SshSocket.h>
```

Macros

- #define **SSH_SUCCESS** (0)

  The function completed successfully.
- #define **SSH_ERROR_FAILED_SESSION_FAILED** (-300)

  Not currently used, kept for backward compatibility.
- #define **SSH_ERROR_FAILED_NEGOTIATION** (-301)

  The connection failed the SSH negotiation.
- #define **SSH_ERROR_FAILED_INITIALIZATION** (-302)

  The SSH system failed to initialize.
- #define **SSH_ERROR_FAILED_CONTEXT_INIT** (-303)

  Unable to instantiate SSH client/server context.
- #define **SSH_ERROR_BAD_KEY** (-304)

  The key provided was invalid.
- #define **SSH_ERROR_BAD_ARGUMENT** (-305)

  A bad argument was provided to the function.
- #define SSH_FAILED_KEY_CHECK (-306)

Typedefs

- typedef int(∗ sshUserAuthenticateFn) (const char ∗usernamePtr, const char ∗passwordPtr)

  [DEPRECATED] User provided SSH username and password authenticate routine for a server. Please consider sshUserAuthenticateWithTypeFn.
- typedef int(∗ sshUserAuthenticateWithTypeFn) (const char ∗usernamePtr, const char ∗authValPtr, AuthType authType)

  User provided SSH user authenticate routine for a server.
- typedef int(∗ sshGetUserPwFn) (const NBString &usernamePtr, NBString &passwordPtr)

  User provided SSH user password authentication routine for clients.
- typedef int(∗ sshGetUserKeyFn) (const NBString &usernamePtr, NBString &publicKey, NBString &privateKey, NBString &keyType)

  User provided SSH user key authenticate routine for clients.
- typedef int(∗ sshUserGetKeyFn) (int keyRequested, const unsigned char ∗∗keyBufferPtr, int ∗keyLengthPtr)

  The user defined callback to get the server key used during the initial SSH negotiation.

Functions

- void SshSetUserAuthenticate (sshUserAuthenticateFn sshUserAuthenticateFnPtr)

  [DEPRECATED] Sets the user defined server authentication function. Please consider sshUserAuthenticateWith↩ TypeFn.
- sshUserAuthenticateFn SshGetUserAuthenticate (void)

  [DEPRECATED] Gets the user defined server authentication function. Please consider SshGetUserAuthenticate↩ WithType.
- void SshSetUserAuthenticateWithType (sshUserAuthenticateWithTypeFn sshUserAuthenticateFnPtr)

  Sets the user defined server authentication function.
- sshUserAuthenticateWithTypeFn SshGetUserAuthenticateWithType (void)

  Gets the user defined server authentication function..
- void SshClientSetGetUserPaswordFn (sshGetUserPwFn sshGetUserPwFnPtr)

  Sets the user defined client authentication function for getting user passwords during SSH authentication.

- sshGetUserPwFn SshClientGetUserPaswordFn (void)

  Gets the user defined client authentication function for getting a user password during authentication.
- void SshClientSetGetUserKeyFn (sshGetUserKeyFn sshGetUserKeyFnPtr)

  Sets the user defined client authentication function for getting user keys during SSH authentication.
- sshGetUserKeyFn SshClientGetUserKeyFn (void)

  Gets the user defined client authentication function for getting a user key during authentication.
- void SshSetUserGetKey (sshUserGetKeyFn sshUserGetKeyFnPtr)

  Sets the user defined callback method to provide the server key.
- sshUserGetKeyFn SshGetUserGetKey (void)

  Gets the user defined callback method to provide the server key.
- bool SshValidateKey (const char *candidateKey, int candidateKeySize, int *keyTypePtr, int key↩ Format=WOLFSSH_FORMAT_ASN1)

  Takes a key and returns if it's valid or not.
- bool SshWritePublicKey (int publicKeyFd, unsigned char *candidateKey, int candidateKeySize)

  Write public key to file descriptor. Takes both PEM and ANS1 formats.
- int NbSshInit ()

  Initializes the underlying SSH framework. This will start a background task used to handle negotiations and SSH traffic. This is automatically called by SshAccept and SshConnect, and doesn't need to be called directly except in special circumstances.
- int SshAccept (int listenFd, IPADDR *clientAddress, uint16_t *securePort, uint16_t timeout)

  Accepts and negotiates SSH session. Automatically calls NbSshInit() if required.
- int SshConnect (IPADDR clientAddress, uint16_t securePort, uint16_t localPort, uint16_t timeout, const char *username)

  Issues a connect request to negotiates an SSH session. Automatically calls NbSshInit() if required.
- SshSocket * SshNegotiateSession (int fd)

  Negotiates an SSH server session on an open file descriptor.
- SshSocket * SshNegotiateSessionClient (int secureFd, const char *username)

  Negotiates an SSH client session on an open file descriptor.
- void SshPrintStatistics (int secureFd)

  Negotiates an SSH client session on an open file descriptor.
- int SshGetKeySize ()

  Determins and returns SSH's installed key size.
- int SshSetBannerText (const char *banner)

  Sets the banner text displayed by the SSH server on connection.
- int SshSetSockOption (int fd, int option)

  Set SSH TCP socket options.
- int SshClrSockOption (int fd, int option)

  Clear SSH TCP socket options.
- int SshGetSockOption (int fd)

  Returns the options for the specified SSH TCP socket.

24.262.1 Detailed Description

NetBurner SSH API.

24.263 NbWolfSsh.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00016 #ifndef _NB_SSH_H_
00017 #define _NB_SSH_H_
00018 #include <nettypes.h>
00019 #include <stdio.h>
00020 #include <basictypes.h>
00021
00022 #ifdef NB_SSH_SUPPORTED
00023
00024 #ifndef WOLFSSL_USER_SETTINGS
00025 #define WOLFSSL_USER_SETTINGS   // This wasn't getting defined in the project source files in
      NBEclipse
00026 #endif
00027
00028 #include <ssh/wolfssh/ssh.h>
00029 #include <ssh/NetBurner/UserAuthManager.h>
00030 #include <ssh/NetBurner/SshSocket.h>
00031
00032 /*
00033  ******************************************************************************
00034  *
00035  * Definitions
00036  *
00037  ******************************************************************************
00038  */
00039 /* Default SSH Server Port*/
00040 #define SSH_SECURE_SHELL_IANA_ASSIGNED_PORT (22)
00041
00042 /* Connection timeout */
00043 #ifdef _DEBUG
00044 #define SSH_CONNECTION_TIMEOUT_IN_SECS (360)
00045 #else /* #ifdef _DEBUG */
00046 #define SSH_CONNECTION_TIMEOUT_IN_SECS (180)
00047 #endif /* #ifdef _DEBUG */
00048
00049 /* Default static key size, no choice is 1024 */
00050 /* #define SSH_RSA_KEY_DEFAULT_512                 (  512 ) */
00051 #define SSH_RSA_KEY_DEFAULT_1024 (1024)
00052 /* #define SSH_RSA_KEY_DEFAULT_2048                ( 2048 ) */
00053
00054 #define SSH_DSS_KEY_DEFAULT_512 (512)
00055 /* #define SSH_DSS_KEY_DEFAULT_1024                ( 1024 ) */
00056 /* #define SSH_DSS_KEY_DEFAULT_2048                ( 2048 ) */
00057
00063 #define SSH_SUCCESS (0)
00064 #define SSH_ERROR_FAILED_SESSION_FAILED (-300)
00065 #define SSH_ERROR_FAILED_NEGOTIATION (-301)
00066 #define SSH_ERROR_FAILED_INITIALIZATION (-302)
00067 #define SSH_ERROR_FAILED_CONTEXT_INIT (-303)
00068 #define SSH_ERROR_BAD_KEY (-304)
00069 #define SSH_ERROR_BAD_ARGUMENT (-305)
00070 #define SSH_FAILED_KEY_CHECK \
00071     (-306)
00073
00075 /* SSH Key */
00076 #define SSH_KEY_RSA (1)
00077 #define SSH_KEY_DSS (2)
00078 #define SSH_KEY_ECC (3)
00079
00080 /*
00081  ******************************************************************************
00082  *
00083  * SSH "C" Library Interface
00084  *
00085  ******************************************************************************
00086  */
00087 #ifdef __cplusplus
00088 extern "C"
00089 {
00090 #endif
00091
00092     /*Functions:*/
00093     /*Group:SSH functions*/
00094
00095     // Server
00096     // Deprecated, only left for backwards compatibility. Should use sshUserAuthenticateWithTypeFn
      instead.
00107     typedef int (*sshUserAuthenticateFn)(const char *usernamePtr, const char *passwordPtr);
00108
00119     typedef int (*sshUserAuthenticateWithTypeFn)(const char *usernamePtr, const char *authValPtr,
```

```
    AuthType authType);
00120
00121     // Client
00131     typedef int (*sshGetUserPwFn)(const NBString &usernamePtr, NBString &passwordPtr);
00132
00144     typedef int (*sshGetUserKeyFn)(const NBString &usernamePtr, NBString &publicKey, NBString
    &privateKey, NBString &keyType);
00145
00146     // Server Callbacks
00154     void SshSetUserAuthenticate(sshUserAuthenticateFn sshUserAuthenticateFnPtr);
00155
00163     sshUserAuthenticateFn SshGetUserAuthenticate(void);
00164
00172     void SshSetUserAuthenticateWithType(sshUserAuthenticateWithTypeFn sshUserAuthenticateFnPtr);
00173
00181     sshUserAuthenticateWithTypeFn SshGetUserAuthenticateWithType(void);
00182
00183     // Client Callbacks
00193     void SshClientSetGetUserPaswordFn(sshGetUserPwFn sshGetUserPwFnPtr);
00194
00204     sshGetUserPwFn SshClientGetUserPaswordFn(void);
00205
00215     void SshClientSetGetUserKeyFn(sshGetUserKeyFn sshGetUserKeyFnPtr);
00216
00226     sshGetUserKeyFn SshClientGetUserKeyFn(void);
00227
00241     typedef int (*sshUserGetKeyFn)(int keyRequested, const unsigned char **keyBufferPtr, int
    *keyLengthPtr);
00242
00251     void SshSetUserGetKey(sshUserGetKeyFn sshUserGetKeyFnPtr);
00252
00261     sshUserGetKeyFn SshGetUserGetKey(void);
00262
00274     bool SshValidateKey(const char *candidateKey, int candidateKeySize, int *keyTypePtr, int keyFormat
    = WOLFSSH_FORMAT_ASN1);
00275
00286     bool SshWritePublicKey(int publicKeyFd, unsigned char *candidateKey, int candidateKeySize);
00287
00300     int NbSshInit();
00301
00316     int SshAccept(int listenFd, IPADDR *clientAddress, uint16_t *securePort, uint16_t timeout);
00317
00335     int SshConnect(IPADDR clientAddress, uint16_t securePort, uint16_t localPort, uint16_t timeout,
    const char *username);
00336
00344     SshSocket *SshNegotiateSession(int fd);
00345
00354     SshSocket *SshNegotiateSessionClient(int secureFd, const char *username);
00355
00361     void SshPrintStatistics(int secureFd);
00362
00368     int SshGetKeySize();
00369
00378     int SshSetBannerText(const char *banner);
00379
00391     int SshSetSockOption(int fd, int option);
00392
00404     int SshClrSockOption(int fd, int option);
00405
00416     int SshGetSockOption(int fd);
00417
00418
00419 #ifdef __cplusplus
00420 };
00421 #endif
00422
00423 #endif /* NB_SSH_SUPPORTED */
00424 #endif /* _NB_SSH_H_ */
00425
```

24.264 SshSocket.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_SSH_SOCKET_H_
00006 #define _NB_SSH_SOCKET_H_
00007
00008 #include <predef.h>
00009
00010 #ifdef NB_SSH_SUPPORTED
00011 #include <nbrtos.h>
00012
00013 #include <crypto/NetBurner/CryptoSocket.h>
```

```
00014 #include <crypto/wolfssl/wolfcrypt/sha256.h>
00015 #include <crypto/wolfssl/wolfcrypt/coding.h>
00016 #include <ssh/wolfssh/ssh.h>
00017
00018 class SshSocket : public CryptoSocket
00019 {
00020    public:
00021     SshSocket();
00022     ~SshSocket();
00023
00024     int InitSocket(int tcpFd, WOLFSSH_CTX *ctx, int sockFlags = 0);
00025
00026     void CleanupTcpClose();
00027
00028     // Called from wolf ssl loop mostly to handle notifications and
00029     // time out stuff for all async process except read
00030     void ProcessAsyncStuff() override;
00031     uint32_t GetCheckableIndexFromSocket() override;
00032     int CheckSocketRecv() override;
00033     uint32_t SocketRead(char *buf, uint32_t len) override;
00034     uint32_t SocketWrite(const char *buf, uint32_t len) override;
00035     SocketHasData_t SocketHasData() override;
00036
00037     // Need to add a GetNewSocket() for each derived type
00038     static SshSocket *GetSocketFromIndex(uint32_t index);
00039     static SshSocket *GetSocketFromTcpFd(int tcpFd);
00040     static SshSocket *GetSocketFromCryptoFd(int cryptoFd);
00041     static SshSocket *GetNewSocket(int tcpFd, WOLFSSH_CTX *ctx, int sockFlags = 0);
00042
00043     WOLFSSH *GetWolfSsh() { return (WOLFSSH *)m_wolfCtx; }
00044
00045    private:
00046     void WriteUnwrittenData() override;
00047 };
00048
00049 #endif   // NB_SSH_SUPPORTED
00050 #endif   // _NB_SSH_SOCKET_H_
```

24.265 UserAuthManager.h File Reference

NetBurner User Authorization Manager.

```
#include <predef.h>
#include <basictypes.h>
#include <nbstring.h>
#include <crypto/wolfssl/wolfcrypt/sha256.h>
```

Classes

- struct UserAuthRecord

  A stored record of a user's authorization credentials. The value is hashed when saved so it can't be read directly. User's can currently only have one authorization record per user name.

- class UserAuthManager

  The user authorization manager class allows application developers the ability to manage user authorization records. The can be loaded and saved to any storage space, including the config system or UserParams. Authorization values are hashed before being saved. Validation compares both the hash as well as the authorization type. Adding, updating, and removing records will automatically call the user devined save functions. For usage, please see the example found in `examples/SSH/sshServerUserAuth`.

Typedefs

- typedef int(∗ SaveAuthRecordsFn) (const UserAuthRecord ∗authRec)

  User provided function for saving user authorization records. This allows the users to dictate where their authorization records are stored.

- typedef int(∗ LoadAuthRecordsFn) (UserAuthRecord ∗authRec)

  User provided function for loading user authorization records. This allows the users to dictate where their authorization records are stored.

Enumerations

- enum AuthType : int8_t { eAuthTypeDefault = 0 , eAuthTypePassword = 1 , eAuthTypeKey = 2 }

  The types of authorization requests that are managed. These just indicate what the has value is, and don't provide any specific processing logic.
- enum AuthResponse : int16_t { eAuthSuccess = 1 , eAuthErrorUserExists = -1 , eAuthErrorUserDoesNotExist = -2 , eAuthErrorNoEmptyUserAuthRecords = -3 , eAuthErrorUnableToCreateHash = -4 , eAuthErrorAuthCheckFailed = -5 , eAuthErrorAuthTypeMismatch = -6 , eAuthErrorFailedRecordUpdate = -7 , eAuthErrorUnableToAddUser = -8 , eAuthErrorSaveFailed = -9 }

  Response return codes when checking for the authorization status of a user.

24.265.1 Detailed Description

NetBurner User Authorization Manager.

24.266 UserAuthManager.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00016 #ifndef _USER_AUTH_MANAGER_H_
00017 #define _USER_AUTH_MANAGER_H_
00018
00019 #include <predef.h>
00020
00021 #ifdef NB_SSH_SUPPORTED
00022
00023 #ifndef WOLFSSL_USER_SETTINGS
00024 #define WOLFSSL_USER_SETTINGS   // This wasn't getting defined in the project source files in
      NBEclipse
00025 #endif
00026
00027 #include <basictypes.h>
00028 #include <nbstring.h>
00029
00030 #include <crypto/wolfssl/wolfcrypt/sha256.h>
00031
00032 #define MAX_AUTH_RECORDS 25
00033 #define WC_SHA256_DIGEST_SIZE 32
00034
00043 enum AuthType : int8_t
00044 {
00045     eAuthTypeDefault = 0,
00046     eAuthTypePassword = 1,
00047     eAuthTypeKey = 2,
00048 };
00058 enum AuthResponse : int16_t
00059 {
00060     eAuthSuccess = 1,
00061     eAuthErrorUserExists = -1,
00062     eAuthErrorUserDoesNotExist = -2,
00063     eAuthErrorNoEmptyUserAuthRecords = -3,
00064     eAuthErrorUnableToCreateHash = -4,
00065     eAuthErrorAuthCheckFailed = -5,
00066     eAuthErrorAuthTypeMismatch = -6,
00067     eAuthErrorFailedRecordUpdate = -7,
00068     eAuthErrorUnableToAddUser = -8,
00069     eAuthErrorSaveFailed = -9,
00070 };
00077 struct UserAuthRecord
00078 {
00079     NBString m_userName;
00080     uint8_t m_authHash[WC_SHA256_DIGEST_SIZE];
00081     AuthType m_authType;
00082     uint32_t m_authLevel;
00084 };
00085
00086 // Callbacks used to load and save user authentication information
00096 typedef int (*SaveAuthRecordsFn)(const UserAuthRecord *authRec);
00097
00107 typedef int (*LoadAuthRecordsFn)(UserAuthRecord *authRec);
00108
```

```
00115 class UserAuthManager
00116 {
00117    public:
00121     UserAuthManager() {}
00122
00126     ~UserAuthManager() {}
00127
00137     bool Init(SaveAuthRecordsFn svRcFn, LoadAuthRecordsFn ldRcFn);
00138
00147     bool UserExists(const NBString &userName);
00148
00159     AuthResponse AddUserAuth(const NBString &userName, const NBString &auth, AuthType authType);
00160
00172     AuthResponse CheckUserAuth(const NBString &userName, const NBString &auth, AuthType authType);
00173
00185     AuthResponse CheckUserAuth(const NBString &userName, byte *auth, AuthType authType);
00186
00198     AuthResponse UpdateUserAuth(const NBString &userName, const NBString &newAuth, AuthType authType);
00199
00208     AuthResponse RemoveUserAuth(const NBString &userName);
00209
00222     AuthResponse CheckUserAuthLevel(const NBString &userName, uint32_t authLevel, bool hasAll = true);
00223
00235     AuthResponse SetUserAuthLevel(const NBString &userName, uint32_t authLevel);
00236
00247     AuthResponse ClrUserAuthLevel(const NBString &userName, uint32_t authLevel);
00248
00252     void ListUsers();
00253
00257     int GetMaxAuthRecords() { return MAX_AUTH_RECORDS; }
00258
00259    private:
00266     int16_t GetNextEmptyRecord();
00267
00274     int16_t FindUser(const NBString &userName);
00275
00285     bool CreateHash(const NBString &auth, uint8_t *outHash);
00286
00287     // User defined functions for loading and saving auth data.
00296     int (*m_SaveAuthRecords)(const UserAuthRecord *authRec) = nullptr;
00297
00306     int (*m_LoadAuthRecords)(UserAuthRecord *authRec) = nullptr;
00307
00308     UserAuthRecord m_authRecords[MAX_AUTH_RECORDS];
00309 };
00310
00311 #endif   /* NB_SSH_SUPPORTED */
00312 #endif   // _USER_AUTH_MANAGER_H_
00313
```

24.267 nbWifiApi.h File Reference

```
#include <buffers.h>
#include <nettypes.h>
#include <wifi/nbWifiConstants.h>
```

24.268 nbWifiApi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef __NBWIFIAPI_H
00010 #define __NBWIFIAPI_H
00011
00012 #include <buffers.h>
00013 #include <nettypes.h>
00014 #include <wifi/nbWifiConstants.h>
00015
00016 // Scanning and connecting uses the SAME exact AP deifition.
00017 struct APDefinition
00018 {
00019     char SSID[SSID_MAX_LEN + 1];
00020     char PassPhrase[PASS_MAX_LEN + 1];       // Should be "" empty string for open networks.
00021     MACADR BSSID;                            // All Zeros ignored on connect.
00022     bool bAdhoc;                             // True if its an or to create an Adhoc network.
00023     char Security[SEC_HUMAN_MAX_LEN + 1];    // On scan empty string for  open on connect empty string
      for don't care OPEN for forcing open.
```

```
00024     char Cipher[CIPHER_HUMAN_MAX_LEN + 1];   // On Scan indicates the Cipher used with the security,
      ignored on connect.
00025     NB::Constants::RadioBand band;
00026     uint16_t Channel;
00027     int rssi;   // Ignored on connect calls....
00028 };
00029
00030 #define WIFI_Result_Success (1)
00031 #define WIFI_Result_BadPass (-1)
00032 #define WIFI_Result_NoAPFound (-2)
00033 #define WIFI_Result_BSSID_NotFound (-3)
00034 #define WIFI_Result_TimedOut (-4)
00035
00036 typedef void (*ReceiveScanResultFunc)(const APDefinition &ap);
00037
00038 class WifiInteface
00039 {
00040    public:
00041     // Connection functions
00042     // If all three SSID, BSSID and PassPhrase are empty, connects to strongest open network
00043     int Connect();   // Use the SSID and PassPhrase stored in the config record.
00044     int Connect(const char *SSID, const char *Pass, bool bAdHoc = false);
00045     int Connect(APDefinition &ap);
00046     int Connect(APDefinition *pAps);   // Connect to the AP deifitions on the list in order of
      preference.
00047
00048     void Disconnect();
00049
00050     // Scan Functions:
00051
00052     // Syncronous Scan...
00053     APDefinition *Scan(const char *ssid = nullptr);   // nullptr SSID means scan for all
00054     APDefinition *ScanWithSettings(const char *ssid = nullptr,
00055                                    uint8_t channelCount = 0,
00056                                    const uint16_t *channelList = nullptr,
00057                                    uint8_t band = 0,
00058                                    uint8_t infrastructureType = 0);
00059     // Returns a pointer to a list of APDefinitions
00060     // SSID, nullptr on the last one in the list.
00061     // This function is not reentrant and the result is only valid until the Next call to Scan
00062
00063     // Asyncronous Scan....
00064
00065     // If the underlying WIFI driver has an internally timed scan then the
00066     // call back funciton will receive an AP with SSID =nullptr as the last indicator...
00067     // Otherwise you shoudl probably call StopAsyncScan
00068     int StartAsyncScan(ReceiveScanResultFunc *pCallBackFunc, const char *ssid = nullptr);
00069     int StartAsyncScanWithSettings(ReceiveScanResultFunc *pCallBackFunc,
00070                                    const char *ssid = nullptr,
00071                                    uint8_t channelCount = 0,
00072                                    const uint16_t *channelList = nullptr,
00073                                    uint8_t band = 0,
00074                                    uint8_t infrastructureType = 0);
00075     // In some implmentaions this may be an empty function....
00076     int StopAsyncScan();
00077
00078     // Status functions:
00079     int GetCurSSID(char *buf, int maxlen);
00080     MACADR GetCurBSSID();
00081     int GetCurrentAP(APDefinition &ap);
00082     bool Connected();
00083     int GetSignalStrength();
00084 };
00085
00086 #endif /* ----- #ifndef __NBWIFIAPI_H  ----- */
```

24.269 nbWifiDriver.h File Reference

```
#include <buffers.h>
#include <nettypes.h>
#include <netinterface.h>
#include <constants.h>
#include <utils.h>
#include <wifi/wifiDriver.h>
#include <wifi/nbWifiConstants.h>
#include <wifi/nbwifi/nbWifiMsgStructs.h>
```

24.270 nbWifiDriver.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef __NBWIFIDRIVER_H
00010 #define __NBWIFIDRIVER_H
00011
00012 #include <buffers.h>
00013 #include <nettypes.h>
00014 #include <netinterface.h>
00015 #include <constants.h>
00016 #include <utils.h>
00017 #include <wifi/wifiDriver.h>
00018 #include <wifi/nbWifiConstants.h>
00019 #include <wifi/nbwifi/nbWifiMsgStructs.h>
00020
00021 #define WIFI_API_TASK_PRIO WIFI_RX_TASK_BASE_PRIO + MAX_WIFI_INTERFACES
00022 #define MAX_OPTION_TABLES 256   // options have an 8-bit table id
00023 #define WIFI_MAX_QUEUED_BUFFERS 8
00024 #define WIFI_RX_MIN_BUFFER_THRESHHOLD 20
00025 #define WIFI_INIT_BAUDRATE 230400
00026 #define WIFI_RX_DELAY (TICKS_PER_SECOND / 4)
00027 #define WIFI_PEND_TIMEOUT (TICKS_PER_SECOND / 2)
00028 #define WIFI_UPDATE_MAX_PAYLOAD \
00029     (ETH_MAX_SIZE - sizeof(NB::NBWifiAPIMessage::Header::APIRequest) -
      sizeof(NB::NBWifiAPIMessage::Request::FirmwareChunk))
00030
00031 #define WIFI_SERIAL_SYNC_SEQ "a\x1B\x0B"
00032 #define TOKEN_FIRST 'a'
00033 #define TOKEN_MID '\x1B'
00034 #define TOKEN_LAST '\x0B'
00035 #define TOKEN_FIRST_COUNT 3
00036 #define TOKEN_MID_COUNT 2
00037 #define TOKEN_LAST_COUNT 1
00038
00039 #define MAX_PENDING_COMMANDS 5
00040 #define READ_ERROR_LIMIT 5
00041 #define READ_CREDIT_REPORT_THRESHOLD 8
00042 #define REQUEST_CREDIT_REPORT_THRESHOLD 4
00043
00044 #ifndef _STRUCT_PACKED
00045 #define _STRUCT_PACKED __attribute__((__packed__))
00046 #endif
00047
00048 typedef void (*ReceiveScanResultFunc)(const nbWifiScanResult &ap);
00049
00050 extern void SetWifiSPISpeed(int);
00051
00052 namespace NB
00053 {
00054
00055 class NBWifi : public Wifi
00056 {
00057    private:
00058     /* ---- Private Structure definitions ---- */
00059     struct busWait
00060     {
00061         OS_MBOX mbox;
00062         uint8_t error;
00063         uint32_t code;
00064     };
00065
00066     struct responseWaiter
00067     {
00068         bool valid;
00069         uint8_t requestType;
00070         uint8_t commandID;
00071         OS_MBOX *pMbox;
00072     };
00073
00074     struct commandsWaitingResponseStruct
00075     {
00076         int lastKnownFree;
00077         responseWaiter waitList[MAX_PENDING_COMMANDS];
00078     };
00079     /* ---- End Private Structure definitions ---- */
00080
00081     /* ---- Data members ---- */
00082     volatile uint8_t currentCommandID;
00083
00084     volatile bool waitingForStatus;
00085     OS_MBOX statusMailbox;
00086     OS_SEM startupSem;
```

```
00087     commandsWaitingResponseStruct commandsWaitingResponse;
00088     PoolPtr options[MAX_OPTION_TABLES];
00089     PoolPtr connectConfigBuf;
00090
00091     char hardwareTypeOverflow[MAX_HARDWARE_TYPE_LENGTH + 1];   // <- this must follow the device info
      struct
00092                                                                //  it is extra space for a nullbyte
      for full length strings
00093     uint32_t slaveFirmwareOffset;                             // used when updating slave firmware
00094     bool slaveUpdateReady;
00095     /* ---- End Data members ---- */
00096
00097     /* ---- Constructors ---- */
00098     NBWifi();
00099     NBWifi(const NBWifi &rhs);
00100     /* ---- End Constructors ---- */
00101
00102     /* ---- Private methods ---- */
00103
00104     bool ProcessInternal(PoolPtr messageBuffer, NB::NBWifiAPIMessage::Internal::InternalTypes
      internalType);
00105
00106     void SendReconnectRequest();
00107
00108     bool ProcessRequest(PoolPtr messageBuffer, uint8_t requestType);
00109     bool ProcessResponse(PoolPtr messageBuffer, uint8_t responseType);
00110     bool ProcessResponseID(PoolPtr messageBuffer);
00111     bool ProcessScanResult(PoolPtr messageBuffer);
00112     bool ProcessStatus(PoolPtr messageBuffer);
00113     bool ProcessDeviceInfo(PoolPtr messageBuffer);
00114     bool ProcessOptionTableResponse(PoolPtr messageBuffer);
00115
00116     bool ProcessCreditReport(PoolPtr messageBuffer);
00117
00118     bool DevFWSupported();
00119
00120     int AddWaitingCommand(uint8_t requestType, uint8_t commandID, OS_MBOX *pmbox);
00121     void RemoveWaitingCommand(uint8_t commandID);
00122     int ReadyInterface();
00123
00124     void RetrieveOptionTable(uint8_t tableNum);
00125     void RetrieveAllOptionTables();
00126
00127     PoolPtr GetNetworkStatus();
00128     PoolPtr GetDeviceInformation();
00129     /* ---- End Private methods ---- */
00130
00131   protected:
00132     typedef enum
00133     {
00134         State_NotRegistered = 0,
00135         State_TaskNotRunning = 1,
00136         State_NoDevice = 2,         // Beginning driver initialization
00137         State_NoInfo = 3,           //
00138         State_InvalidDevice = 4,    // Unsupported device or firmware, halting
00139         State_NoOptions = 5,        //
00140         State_NotConfigured = 6,    // Driver initialized
00141         State_NotConnected = 7,     //
00142         State_Connected = 8,        // Connection established
00143         State_Reconnecting = 9,    // Attempting to reconnect to last successful AP connection.
      Disconnected due to host triggering a chipset
00144                                    // reset due to detecting a crash
00145     } DriverState;
00146
00147     static OS_FIFO WifiAPIRxFifo;
00148
00149     int myInterfaceNumber;
00150     OS_SEM taskPauseSem;
00151     OS_FIFO BusMsgFifo;
00152     OS_FIFO BufferTxFifo;
00153     uint32_t lastCreditCheckTick;
00154     bool creditReportQueued;
00155     bool creditReporting;
00156     bool slaveHalted;
00157     uint32_t lastReportTick;
00158     NBWifiAPIMessage::Response::DeviceInfo devInfo;
00159     DriverState currentState;
00160
00161     ReceiveScanResultFunc pfScanResultsCallback;
00162     bool scanActive;
00163     uint8_t availableCredits;
00164     uint16_t pendingTxBuffers;
00165     uint8_t rxTaskID;
00166     OS_CRIT creditCheckCrit;
00167     OS_CRIT commandsWaitingCrit;
00168     OS_CRIT networkStatusCrit;
00169     OS_CRIT bufferTxFifoCrit;
```

```
00170
00171     static uint32_t DriverStk[MAX_WIFI_INTERFACES][WIFI_TASK_STACK_SIZE];
00172     static uint32_t APITaskStk[WIFI_TASK_STACK_SIZE];
00173
00174     NBWifi(CommBus busType, int resetPinConnector, int resetPinMum, const char *name);
00175     ~NBWifi();
00176
00177     void SendBusMessage(PoolPtr txMessage, bool waitForCredits = true);
00178     virtual void SendBusMessage_Core(PoolPtr txMessage) = 0;
00179     virtual void RXTask() = 0;
00180     int RegisterDriver();
00181     uint8_t GetNextCommandID();
00182     void PushRXCreditReport(PoolPtr pp);
00183     void PushPayloadCreditReport(PoolPtr &pp);
00184     void ExternalReconnect(PoolPtr configMsg);
00185     void InternalReset();
00186
00187    public:
00188     static void RXTaskLauncher(void *wifiDriverObj);
00189     static void APITask(void *throwaway);
00190
00191     // static functions for use with interface blocks
00192     static void TransmitBuffer_0(PoolPtr txBuffer);
00193     static void TransmitBuffer_1(PoolPtr txBuffer);
00194     static void kill_0();
00195     static void kill_1();
00196     static BOOL linkActive_0();
00197     static BOOL linkActive_1();
00198     static void enab_multicast_0(MACADR macAddress, BOOL addAddress);
00199     static void enab_multicast_1(MACADR macAddress, BOOL addAddress);
00200
00201     void TransmitBuffer(PoolPtr txBuffer);
00202     virtual void TransmitBuffer_Core(PoolPtr txBuffer, uint8_t credits) = 0;
00203     void enab_multicast(MACADR macAddress, BOOL addAddress);
00204
00205     // APITask control methods
00206     int APIStart();
00207     static int APIStartTask();
00208     static void APIPause();
00209     static void APIResume();
00210     static void APIKill();
00211
00212     // RXTask control methods
00213     int Start();
00214     void Pause();
00215     void Resume();
00216     void Kill();
00217
00218     bool isRegistered() const;
00219     int GetWifiInterfaceNumber() const;
00220     int GetSystemInterfaceNumber() const;
00221
00222     // WLAN manipulation and statistics functions
00223     int ConnectWithOptions(const char *ssid = nullptr,
00224                            const char *passwd = "",
00225                            uint8_t retryCount = CONNECT_RETRIES,
00226                            uint8_t optionCount = 0,
00227                            uint16_t *optionList = nullptr,
00228                            uint8_t ssidLen = 0);
00229
00230     int ConnectToAP(const char *ssid = nullptr, const char *passwd = "", uint8_t retryCount =
      CONNECT_RETRIES);
00231
00232     int StartAP(const char *ssid = nullptr,
00233                 const char *passwd = "",
00234                 uint8_t channel = DEFAULT_TABLE_LABEL_CHANNEL,
00235                 uint8_t security = DEFAULT_TABLE_LABEL_SEC,
00236                 uint8_t cipher = DEFAULT_TABLE_LABEL_CIPH,
00237                 uint8_t ssidLen = 0) override;
00238
00239     int StartConfigAP(uint8_t channel = DEFAULT_TABLE_LABEL_CHANNEL) override;
00240
00241     void Disconnect() override;
00242
00243     // Results of Scan are valid until the next call to 'Scan' or 'FreeScanList'
00244     nbWifiScanResult *Scan(const char *ssid = nullptr, uint8_t optionCount = 0, uint16_t *optionList =
      nullptr) override;
00245     void FreeScanList();
00246
00247     int StartAsyncScan(ReceiveScanResultFunc pfCallbackFunc,
00248                        const char *ssid = nullptr,
00249                        uint8_t optionCount = 0,
00250                        uint16_t *optionList = nullptr);
00251
00252     void StopAsyncScan();
00253
00254     // Status functions:
```

```
00255     int GetCurSSID(char *buf, int maxlen) override;
00256     MACADR GetCurBSSID() override;
00257     void GetCurrentAP(driverStatusStruct *ap) override;
00258     void GetDeviceInformation(nbWifiDeviceInfo *ap) override;
00259     bool Connected() override;
00260     int GetSignalStrength() override;
00261     int GetCurChannel() override;
00262     int GetSecurity() override;
00263     int GetCipher() override;
00264     uint8_t GetAvailableCredits() { return availableCredits; };
00265
00266     // Utility Functions
00267     int StoreSSIDPWToConfig(char *ssid, char *password, int ssidLen = 0) override;
00268     int GetSSIDFromConfig(char *returnBuf, int maxLen) override;
00269     int GetKeyFromConfig(char *returnBuf, int maxLen) override;
00270
00271     virtual void SendCreditRequest() = 0;
00272     virtual void SendCreditReport(uint8_t credits = 0xFF) = 0;
00273     void SetMAC(const MACADR *newMAC, bool waitForResponse = true);
00274     int SetITUCountry(NB::ITU_Country::CountryCode_t country);
00275     bool GetLinkStatus();   // does not perform a bus communication, fast
00276     void Print();
00277     bool UpdateSlaveFirmware(uint32_t imageLength, const uint8_t *imageBuffer) override;
00278     bool ReadySlaveUpdate();
00279     bool SendFirmwareChunk(uint32_t length, const uint8_t *data);
00280     bool FinishedSendingFirmware(uint32_t length, const uint8_t *data);
00281     virtual bool SetBusSpeed(uint32_t busSpeed) = 0;
00282
00283     friend void ::SetWifiSPISpeed(int busSpeed);
00284     friend class Wifi;
00285
00286     int ConnectToAP(const char *ssid = nullptr,
00287                     const char *passwd = "",
00288                     uint8_t retryCount = CONNECT_RETRIES,
00289                     uint8_t security = DEFAULT_WIFI_SEC_ALL) override;
00290     int ConnectToAP(nbWifiConnect connect) override;
00291
00292     bool DisplayOptionTable(uint8_t n);
00293     void DisplayAllOptionTables();
00294
00295     static NB::Wifi *GetNewNBWifiSPIDriver();
00296     static NB::Wifi *GetNewNBWifiSerialDriver();
00297 };
00298 }   // namespace NB
00299
00300 extern NB::Wifi *theWifiIntf;
00301
00302 #endif /* ----- #ifndef __NBWIFIDRIVER_H  ----- */
```

24.271 nbWifiMsgStructs.h File Reference

```
#include <nettypes.h>
#include <string.h>
#include <wifi/nbWifiConstants.h>
```

24.272 nbWifiMsgStructs.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef __NBWICEDMSGSTRUCTS_H
00010 #define __NBWICEDMSGSTRUCTS_H
00011
00012 #include <nettypes.h>
00013 #include <string.h>
00014 #include <wifi/nbWifiConstants.h>
00015
00016 // #define HTONS(x) (x)
00017 // #define HTONL(x) (x)
00018 // #define htons(x) (x)
00019 // #define htonl(x) (x)
00020
00021 #define BufferToObjPtr(buffer, objType) ((objType *)((uint8_t *)((buffer)->pData) +
      (buffer)->usedsize))
00022 #define AddToBuffLen(buffer, msgPtr) (buffer)->usedsize += sizeof(*msgPtr)
00023 #define GetSafeLength(val, max) ((val) <= (max)) ? (val) : (max)
```

```
00024
00025 #define MESSAGE_TYPE_BIT_OFFSET 4
00026 #define TYPELEN_FIELD_LEN (sizeof(NB::NBWifiAPIMessage::Header::lenType))
00027 #define _STRUCT_PACKED __attribute__((__packed__))
00028 #define MAX_HARDWARE_TYPE_LENGTH 64
00029
00031 //
00032 // Enum Definitions
00033 //
00035 namespace NB
00036 {
00037 namespace NBWifiAPIMessage
00038 {
00039 namespace Header
00040 {
00041 typedef enum
00042 {
00043     Type_API_Request = 0x0,
00044     Type_API_Response = 0x1,
00045     Type_Credit_Report = 0x2,
00046     Type_Data_Raw = 0x3,
00047     Type_Data_CompressedIP = 0x4,
00048     Type_Internal = 0x7,
00049 } ApiMessageTypes;
00050 }
00051 namespace Request
00052 {
00053 typedef enum
00054 {
00055     Type_Config = 0x0,
00056     Type_Connect_Start = 0x1,
00057     Type_Disconnect_Stop = 0x2,
00058     Type_Scan = 0x3,
00059     Type_SetMAC = 0x4,
00060     Type_SaveConfig = 0x5,
00061     Type_GetCredits = 0x6,
00062     Type_SetBusSpeed = 0x7,
00063     Type_ReadyFirmware = 0x8,
00064     Type_FirmwareChunk = 0x9,
00065     Type_VerifyAndProgFirm = 0xA,
00066     Type_AddRemoveMulticast = 0xB,
00067     Type_SetITUCountry = 0xC,
00068     Type_GetStatus = 0xFD,
00069     Type_GetDeviceInfo = 0xFE,
00070     Type_GetOptionTable = 0xFF
00071 } RequestTypes;
00072 }
00073 namespace Response
00074 {
00075 typedef enum
00076 {
00077     Type_Scan_Result = 0x0,
00078     Type_Status = 0x1,
00079     Type_DeviceInfo = 0x2,
00080     Type_OptionTableResp = 0x3,
00081 } ResponseTypes;
00082 }
00083 namespace Internal
00084 {
00085 typedef enum
00086 {
00087     Type_Internal_JoinAP = 0x0,
00088     Type_Internal_SendStatus = 0x1,
00089     Type_Internal_Reconnect = 0x2,
00090 } InternalTypes;
00091 }
00092
00093 }   // namespace NBWifiAPIMessage
00094 }   // namespace NB
00095
00097 //
00098 // Structure Definitions
00099 //
00101 namespace NB
00102 {
00103 namespace NBWifiAPIMessage
00104 {
00105 // Header structures
00106 namespace Header
00107 {
00108 struct _STRUCT_PACKED lenType
00109 {
00110     beuint16_t val;
00111     beuint16_t xorVal;
00112 };
00113 struct _STRUCT_PACKED APIRequest
00114 {
```

```
00115     lenType lengthAndType;
00116     uint8_t requestType;
00117     uint8_t commandID;
00118     uint8_t pData[];
00119 };
00120 struct _STRUCT_PACKED APIResponse
00121 {
00122     lenType lengthAndType;
00123     uint8_t responseType;
00124     uint8_t commandID;
00125     int32_t returnCode;
00126     // beint32_t     returnCode;
00127     // uint8_t     returnCode;
00128     uint8_t pData[];
00129 };
00130 struct _STRUCT_PACKED CreditReport
00131 {
00132     lenType lengthAndType;
00133     uint8_t availableCredits;
00134 };
00135 struct _STRUCT_PACKED Data_Raw
00136 {
00137     lenType lengthAndType;
00138     uint8_t availableCredits;
00139     uint8_t pData[];
00140 };
00141 struct _STRUCT_PACKED Data_CompressedIP
00142 {
00143     lenType lengthAndType;
00144     uint8_t connectionID;
00145     uint8_t availableCredits;
00146     uint8_t pData[];
00147 };
00148 struct _STRUCT_PACKED Internal
00149 {
00150     lenType lengthAndType;
00151     uint8_t messageType;
00152     uint8_t commandID;
00153 };
00154 }   // namespace Header
00155
00156 // Request structures
00157 namespace Request
00158 {
00159 struct _STRUCT_PACKED ConfigReq
00160 {
00161     uint8_t optionCount;
00162     uint8_t ssidLength;
00163     uint8_t passwdLength;
00164     uint8_t padding;
00165     char options[];   // Placeholder for char array pobe_inter
00166 };
00167 struct _STRUCT_PACKED ConnectStart
00168 {
00169     uint8_t configNum;
00170     uint8_t flags;
00171     uint8_t retryCount;
00172 };
00173 struct _STRUCT_PACKED Scan
00174 {
00175     uint8_t optionCount;
00176     uint8_t ssidLength;
00177     char options[];   // Placeholder for char array pobe_inter
00178 };
00179 struct _STRUCT_PACKED SetMAC
00180 {
00181     MACADR newMAC;
00182 };
00183 struct _STRUCT_PACKED SaveConfig
00184 {
00185     uint8_t configNum;
00186 };
00187 struct _STRUCT_PACKED SetBusSpeed
00188 {
00189     beuint32_t busSpeed;
00190 };
00191 struct _STRUCT_PACKED FirmwareChunk
00192 {
00193     beuint32_t offset;
00194     beuint16_t length;
00195     beuint16_t checksum;
00196     uint8_t pData[];
00197 };
00198 struct _STRUCT_PACKED VerifyAndProgFirm
00199 {
00200     beuint32_t imageLength;
00201     beuint32_t checksum[4];
```

```
00202 };
00203 struct _STRUCT_PACKED AddRemoveMulticast
00204 {
00205     uint8_t flags;
00206     MACADR theMac;
00207 };
00208 struct _STRUCT_PACKED SetITUCountry
00209 {
00210     uint8_t length;
00211     uint8_t pData[];
00212 };
00213 struct _STRUCT_PACKED GetTableReq
00214 {
00215     uint8_t tableNum;
00216 };
00217 }   // namespace Request
00218
00219 // Response structures
00220 namespace Response
00221 {
00222 struct _STRUCT_PACKED ScanResult
00223 {
00224     uint8_t lastAndBand;
00225     uint8_t bssType;
00226     uint8_t channel;
00227     uint8_t security;
00228     uint8_t cipher;
00229     uint8_t ssidLength;
00230     beint16_t rssi;
00231     MACADR bssid;
00232     char ssid[];   // Placeholder for char array pobe_inter
00233 };
00234 struct _STRUCT_PACKED Status
00235 {
00236     uint8_t connected;
00237     uint8_t bssType;
00238     uint8_t security;
00239     uint8_t cipher;
00240     beuint32_t maxTxRate;
00241     beuint16_t rssi;
00242     uint8_t band;
00243     uint8_t channel;
00244     beuint16_t txPower;
00245     MACADR bssid;
00246     uint8_t ssidLength;
00247     char ssid[];   // Placeholder for char array pobe_inter
00248 };
00249 struct _STRUCT_PACKED DeviceInfo
00250 {
00251     uint8_t hardwareMajorRev;
00252     uint8_t hardwareMinorRev;
00253     uint8_t softwareMajorRev;
00254     uint8_t softwareMinorRev;
00255     MACADR hardwareID;
00256     uint8_t hardwareTypeLength;
00257     char hardwareType[MAX_HARDWARE_TYPE_LENGTH + 1];   // allow extra space for null terminating char
00258 };
00259 struct _STRUCT_PACKED OptionTable
00260 {
00261     struct Entry
00262     {
00263         uint8_t value;
00264         uint8_t labelLength;
00265         char label[];   // Placeholder for char array
00266     };
00267     beuint16_t flagsAndEntryCount;
00268     uint8_t tableNum;
00269     Entry entries[];
00270 };
00271 }   // namespace Response
00272 }   // namespace NBWifiAPIMessage
00273 }   // namespace NB
00274
00276 //
00277 // Function declarations
00278 //
00280 namespace NB
00281 {
00282 namespace NBWifiAPIMessage
00283 {
00284 namespace Header
00285 {
00286 void WriteHeader_APIRequest(PoolPtr messageBuffer,
00287                             bool additionalFlag,
00288                             NB::NBWifiAPIMessage::Request::RequestTypes requestType,
00289                             uint8_t commandID);
00290 void WriteHeader_APIResponse(PoolPtr messageBuffer,
```

```
00291                             bool additionalFlag,
00292                             NB::NBWifiAPIMessage::Response::ResponseTypes responseType,
00293                             uint8_t commandID,
00294                             int32_t returnCode);
00295 void FixHeader_Length(PoolPtr messageBuffer);
00296 void WriteHeader_Data_Raw(PoolPtr messageBuffer, bool additionalFlag, uint16_t length);
00297 void WriteHeader_Data_CompressedIP(PoolPtr messageBuffer, bool additionalFlag, uint16_t length,
      uint8_t connectionID);
00298 void WriteCreditReport(PoolPtr messageBuffer, bool additionalFlag, uint16_t length, uint8_t
      availableCredits);
00299 bool VerifyChecksum(NB::NBWifiAPIMessage::Header::lenType lengthAndType);
00300 }   // namespace Header
00301 namespace Request
00302 {
00303 void WriteMsg_ConfigReq(PoolPtr messageBuffer,
00304                         const char *ssid,
00305                         uint8_t ssidLength,
00306                         const char *passwd,
00307                         uint8_t passwdLength,
00308                         uint16_t *optionList,
00309                         uint8_t optionCount);
00310 void WriteMsg_ConnectStart(PoolPtr messageBuffer, uint8_t configNumber, bool scan, uint8_t
      retryCount);
00311 void WriteMsg_SetMAC(PoolPtr messageBuffer, const MACADR *newMAC);
00312 void WriteMsg_SetITUCountry(PoolPtr messageBuffer, NB::ITU_Country::CountryCode_t country);
00313 void WriteMsg_SetBusSpeed(PoolPtr messageBuffer, uint32_t busSpeed);
00314 void WriteMsg_SaveConfig(PoolPtr messageBuffer, uint32_t busSpeed);
00315 void WriteMsg_FirmwareChunk(PoolPtr messageBuffer, uint32_t offset, uint16_t dataLen, uint16_t
      checksum, const uint8_t *data);
00316 void WriteMsg_VerifyAndProgFirm(PoolPtr messageBuffer, uint32_t imageLen, const uint32_t *checksum);
00317 void WriteMsg_AddRemoveMulticast(PoolPtr messageBuffer, bool AddNotRemove, const MACADR *mac);
00318 void WriteMsg_Scan(PoolPtr messageBuffer, uint8_t optionCount, uint16_t *optionList, uint8_t
      ssidLength, const char *ssid);
00319 }   // namespace Request
00320 namespace Response
00321 {
00322 void WriteMsg_ScanResult(PoolPtr messageBuffer,
00323                          bool last,
00324                          Constants::RadioBand band,
00325                          Constants::BssType bssType,
00326                          uint8_t channel,
00327                          uint8_t security,
00328                          uint8_t cipher,
00329                          int16_t rssi,
00330                          const MACADR *pBssid,
00331                          const char *ssid,
00332                          int ssidLength);
00333 void WriteMsg_Status(PoolPtr messageBuffer,
00334                      uint8_t connected,
00335                      int16_t txPower,
00336                      int16_t rssi,
00337                      uint16_t channel,
00338                      uint32_t maxTxRate,
00339                      uint32_t security,
00340                      const MACADR *pBssid,
00341                      uint8_t bssType,
00342                      const char *ssid,
00343                      uint8_t ssidLength);
00344 void WriteMsg_DeviceInfo(PoolPtr messageBuffer,
00345                          uint8_t hardwareMajorRev,
00346                          uint8_t hardwareMinorRev,
00347                          uint8_t softwareMajorRev,
00348                          uint8_t softwareMinorRev,
00349                          const MACADR *pHardwareID,
00350                          const char *hardwareType,
00351                          uint8_t hardwareTypeLength);
00352 }   // namespace Response
00353 }   // namespace NBWifiAPIMessage
00354 }   // namespace NB
00355
00356 #endif /* ----- #ifndef __NBWICEDMSGSTRUCTS_H  ----- */
```

24.273 nbWifiSerial.h File Reference

```
#include <buffers.h>
#include <nettypes.h>
#include <netinterface.h>
#include <constants.h>
#include <utils.h>
#include <wifi/wifiDriver.h>
#include <wifi/nbwifi/nbWifiDriver.h>
```

24.274 nbWifiSerial.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef __NBWIFISERIAL_H
00010 #define __NBWIFISERIAL_H
00011
00012 #include <buffers.h>
00013 #include <nettypes.h>
00014 #include <netinterface.h>
00015 #include <constants.h>
00016 #include <utils.h>
00017 #include <wifi/wifiDriver.h>
00018 #include <wifi/nbwifi/nbWifiDriver.h>
00019
00020 namespace NB
00021 {
00022 class NBWifiSerial : public NBWifi
00023 {
00024     int fd_serial;
00025     int portNum;
00026     uint32_t baudRate;
00027     uint32_t rxCount, txCount;
00028     int SyncSerialStream(int syncRemaining);
00029
00030     void RXTask() override;
00031
00032     // all return the amount of additional space used in the rxBuffer
00033     int ReadMessage(PoolPtr pp, uint16_t messageLength, bool overwrite = false);
00034
00035     virtual void SendBusMessage_Core(PoolPtr txMessage) override;
00036     virtual void TransmitBuffer_Core(PoolPtr txBuffer, uint8_t credits) override;
00037
00038     friend class Master;
00039     friend class NBWifi;
00040
00041   public:
00042     /* InterfaceBlock virtual functions */
00043     void send_func(PoolPtr poolPtr) override;
00044     void kill_if() override;
00045     void EnableMulticast(MACADR macAddress, BOOL addAddress) override;
00046     bool LinkActive() override;
00047     int LinkSpeed() override;
00048     bool LinkDuplex() override;
00049     const char *GetInterfaceName() override;
00050
00051     void SendCreditRequest() override;
00052     void SendCreditReport(uint8_t credits = 0xFF) override;
00053     virtual bool SetBusSpeed(uint32_t busSpeed) override;
00054     NBWifiSerial(int portNum, int resetPinNum, const char *name);
00055 };
00056 }   // namespace NB
00057
00058 #endif /* ----- #ifndef __NBWIFISERIAL_H  ----- */
```

24.275 nbWifiSpi.h File Reference

```
#include <buffers.h>
#include <nettypes.h>
#include <netinterface.h>
#include <constants.h>
#include <utils.h>
#include <wifi/nbwifi/nbWifiDriver.h>
```

24.276 nbWifiSpi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
```

```
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef __NBWIFISPI_H
00010 #define __NBWIFISPI_H
00011
00012 #include <buffers.h>
00013 #include <nettypes.h>
00014 #include <netinterface.h>
00015 #include <constants.h>
00016 #include <utils.h>
00017 #include <wifi/nbwifi/nbWifiDriver.h>
00018
00019 namespace NB
00020 {
00021
00022 class NBWifiSPI : public NBWifi
00023 {
00024     enum BusDirection
00025     {
00026         MASTER_RX = 0x0,
00027         MASTER_TX = 0x1,
00028     };
00029     int m_moduleNum;      // Select SPI for platforms with multiple SPI
00030     int m_csNum;          // SPI chip select, or -1 to use a GPIO as a chip select
00031     int m_IRQNum;         // IRQ number/priority
00032     int m_csConnector;    // Connector containing the chip select
00033     int m_gpioPinNum;     // Used only if a GPIO signal is used for SPI chip select
00034     OS_SEM BusTransferSem;
00035     OS_SEM SlaveReadySem;
00036
00037     static void ISR_0();
00038     static void ISR_1();
00039
00040     void RXTask() override;
00041     virtual void SendBusMessage_Core(PoolPtr txMessage) override;
00042     virtual void TransmitBuffer_Core(PoolPtr txBuffer, uint8_t credits) override;
00043
00044     friend class Wifi;
00045     friend class NBWifi;
00046
00047   public:
00048     /* InterfaceBlock virtual functions */
00049     void send_func(PoolPtr poolPtr) override;
00050     void kill_if() override;
00051     void EnableMulticast(MACADR macAddress, BOOL addAddress) override;
00052     bool LinkActive() override;
00053     int LinkSpeed() override;
00054     bool LinkDuplex() override;
00055     const char *GetInterfaceName() override;
00056
00057     void SendCreditRequest() override;
00058     void SendCreditReport(uint8_t credits = 0xFF) override;
00059     virtual bool SetBusSpeed(uint32_t busSpeed) override;
00060     NBWifiSPI(int irqNum, int moduleNum, int csNum, int connectorNum, int csGpioPinNum, int
    resetPinNum, const char *name);
00061 };
00062
00063 }   // namespace NB
00064
00065 #endif /* ----- #ifndef __NBWIFISPI_H  ----- */
```

24.277 nbWifiConstants.h File Reference

Wifi Constants.

`#include <endian.h>`

Namespaces

- namespace NB::Error
- namespace NB::Error::Init
- namespace NB::Error::Scan
- namespace NB::Error::Connect

Macros

- #define **SEC_VALUE_OPEN** (0x00)

Option list security value: Open network.

- #define **SEC_VALUE_WEP** (0x01)

  Option list security value: WEP.
- #define **SEC_VALUE_WPA** (0x02)

  Option list security value: WPA.
- #define **SEC_VALUE_WPA2** (0x03)

  Option list security value: WPA2.
- #define **SEC_VALUE_WPS** (0x04)

  Option list security value: WPS.
- #define **SEC_VALUE_ANY** (0xFE)

  Option list security value: Any, used to connect as a client regardless of how security is configured for AP.
- #define **SEC_VALUE_UNKNOWN** (0xFF)

  Option list security value: Unknown security.
- #define **CIPH_VALUE_NONE** (0x00)

  Option list cipher value: No cipher.
- #define **CIPH_VALUE_TKIP** (0x01)

  Option list cipher value: TKIP.
- #define **CIPH_VALUE_AES** (0x02)

  Option list cipher value: AES.
- #define **CIPH_VALUE_MIXED** (0x03)

  Option list cipher value: Mixed TKIP/AES.
- #define **CIPH_VALUE_UNKNOWN** (0xFF)

  Option list cipher value: Unkown cipher.
- #define **BSSTYPE_VALUE_INFR** (0x00)

  Infrastructure.
- #define **BSSTYPE_VALUE_ADHOC** (0x01)

  ADHOC.
- #define **BSSTYPE_VALUE_ANY** (0x02)

  Any.
- #define **BSSTYPE_VALUE_UNKNOWN** (0xFF)

  Unknown.
- #define **CONFIG_ERR_SUCCESS** 0

  Success.
- #define **CONFIG_ERR_MSG_LENGTH** -1

  Message length.
- #define **CONFIG_ERR_SSID_LEN_SHORT** -2

  SSID length too short.
- #define **CONFIG_ERR_SSID_LEN_LONG** -3

  SSID length too long.
- #define **CONFIG_ERR_PASSWD_LEN_LONG** -4

  Password length too long.
- #define **CONFIG_ERR_INVALID_TABLE** -5

  Invalid table.
- #define **CONFIG_ERR_INVALID_OPTION** -6

  Invalid option.
- #define **CONFIG_ERR_BSSID_OVERRUN** -7

  BSSID overrun.
- #define **CONFIG_ERR_BSSID_UNDERRUN** -8

  BSSID underrun.
- #define **CONFIG_ERR_MULTI_CHANNEL** -9

  Multi-channel.

- #define **CONFIG_ERR_CONNECTED** -10

  Already connected.
- #define **CONFIG_ERR_UNKNOWN** 1

  Unknown.
- #define **CONNECT_ERR_SUCCESS** 0

  Successfull connect request.
- #define **CONNECT_ERR_NOT_CONFIG** -1

  Paramaters not configured.
- #define **CONNECT_ERR_INVALID_CONFIG_NUM** -2

  Invalid config number.
- #define **CONNECT_ERR_CONNECTED** -3

  Already connected.
- #define **CONNECT_ERR_SSID_NOT_FOUND** -4

  SSID not found.
- #define **CONNECT_ERR_BSSID_NOT_FOUND** -5

  BSSID not found.
- #define **CONNECT_ERR_SEC_MISMATCH** -6

  Security mismatch.
- #define **CONNECT_ERR_CIPH_MISMATCH** -7

  Cipher mismatch.
- #define **CONNECT_ERR_INVALID_KEY** -8

  Invalid password.
- #define **CONNECT_ERR_UNKNOWN** 1

  Unknown error.
- #define **SCAN_ERR_SUCCESS** 0

  Success.
- #define **SCAN_ERR_MSG_LENGTH** -1

  Invalid message length.
- #define **SCAN_ERR_IN_PROGRESS** -2

  Scan in progress.
- #define **SCAN_ERR_SSID_LEN_LONG** -3

  SSID length too long.
- #define **SCAN_ERR_INVALID_TABLE** -4

  Invalid table.
- #define **SCAN_ERR_INVALID_OPTION** -5

  Invalid option.
- #define **SCAN_ERR_TOO_MANY_CHANNELS** -6

  Too many channels.
- #define **SCAN_ERR_UNKNOWN** 1

  Unknown error.
- #define **SAVE_CONF_ERR_SUCCESS** 0

  Success.
- #define **SAVE_CONF_ERR_INVALID_CONFIG_NUM** -1

  Invalid configuration number.
- #define **SAVE_CONF_ERR_NOT_CONFIGURED** -2

  Not configured.
- #define **SAVE_CONF_ERR_UNKNOWN** 1

  Unknown error.

Enumerations

- enum NB::Constants::TaskStartError { NB::Constants::TaskStart_Err_NoError = 0 , NB::Constants::TaskStart_Err_Running = -1 , NB::Constants::TaskStart_Err_NotRegistered = -2 }

  NBWIFI TaskStartError Values.
- enum NB::Constants::TaskKillError { NB::Constants::TaskKill_Err_NoError = 0 , NB::Constants::TaskKill_Err_NotRunning = -1 }

  NBWIFI TaskKillError Values.
- enum NB::Constants::ConnectResult { NB::Constants::Connect_Success = 0 , NB::Constants::Connect_BadPass = -1 , NB::Constants::Connect_NoAPFound = -2 , NB::Constants::Connect_BSSID_NotFound = -3 , NB::Constants::Connect_TimedOut = -4 }

  NBWIFI ConnectResult Values.
- enum NB::Constants::RadioBand { NB::Constants::Band_All = 0 , NB::Constants::Band_2_4GHz = 1 , NB::Constants::Band_5GHz = 2 }

  NBWIFI RadioBand Values.
- enum NB::Constants::BssType { NB::Constants::BssType_Infrastructure = 0 , NB::Constants::BssType_AdHoc = 1 , NB::Constants::BssType_Any = 63 , NB::Constants::BssType_Unknown = 255 }

  NBWIFI BssType Values.
- enum NB::Constants::ScanMethods { NB::Constants::Scan_Passive = 0 , NB::Constants::Scan_Active = 1 }

  NBWIFI ScanMethods Values.
- enum NB::Error::GeneralErrors { NB::Error::NoError = 0 , NB::Error::Timeout = -256 , NB::Error::BusTimeout = -257 , NB::Error::InvalidArgument = -258 , NB::Error::TooManyPendingCommands = -259 , NB::Error::InvalidRequest = -512 }
- enum NB::Error::Init::InitializationErrors { NB::Error::Init::Success = 0 , NB::Error::Init::AlreadyInit = -1 , NB::Error::Init::NoDevice = -2 , NB::Error::Init::InvalidInfo = -3 , NB::Error::Init::DevFirmVer = -4 , NB::Error::Init::DevHwVer = -5 , NB::Error::Init::OptionTables = -6 }
- enum NB::Error::Scan::ScanErrors { NB::Error::Scan::Success = 0 , NB::Error::Scan::NotInitialized = -1 , NB::Error::Scan::InProgress = -2 , NB::Error::Scan::Option = -3 }
- enum NB::Error::Connect::ConnectErrors { NB::Error::Connect::Success = 0 , NB::Error::Connect::NotInitialized = -1 , NB::Error::Connect::AlreadyConnected = -2 , NB::Error::Connect::Option = -3 , NB::Error::Connect::CouldNotConfig = -4 , NB::Error::Connect::SSID_NotFound = -5 , NB::Error::Connect::BSSID_NotFound = -6 , NB::Error::Connect::Sec_NotFound = -7 , NB::Error::Connect::Cipher_NotFound = -8 , NB::Error::Connect::ConnectFailed = -9 }

24.277.1 Detailed Description

Wifi Constants.

24.277.2 Enumeration Type Documentation

24.277.2.1 BssType

`enum NB::Constants::BssType`

NBWIFI BssType Values.

Enumerator

BssType_Infrastructure	BSS Infrastructure mode.
BssType_AdHoc	BSS AdHoc mode.
BssType_Any	BSS arbitrary, but chosen based as max n-bit number (in this case 6)
BssType_Unknown	BSS unknown mode.

24.277.2.2 ConnectResult

enum NB::Constants::ConnectResult
NBWIFI ConnectResult Values.

Enumerator

Connect_Success	Connection successful.
Connect_BadPass	Bad password.
Connect_NoAPFound	No access point found.
Connect_BSSID_NotFound	BSS ID not found.
Connect_TimedOut	Time out.

24.277.2.3 RadioBand

enum NB::Constants::RadioBand
NBWIFI RadioBand Values.

Enumerator

Band_All	All Bands.
Band_2_4GHz	2.4GHz Band
Band_5GHz	5GHz Band

24.277.2.4 ScanMethods

enum NB::Constants::ScanMethods
NBWIFI ScanMethods Values.

Enumerator

Scan_Passive	Passive scan.
Scan_Active	Active scan.

24.277.2.5 TaskKillError

enum NB::Constants::TaskKillError
NBWIFI TaskKillError Values.

Enumerator

TaskKill_Err_NoError	No error has occurred.
TaskKill_Err_NotRunning	Task is already running.

24.277.2.6 TaskStartError

enum NB::Constants::TaskStartError

NBWIFI TaskStartError Values.

Enumerator

TaskStart_Err_NoError	No error has occurred.
TaskStart_Err_Running	Task is already running.
TaskStart_Err_NotRegistered	Driver is not registered.

24.278 nbWifiConstants.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef __NBWIFIDCONSTANTS_H
00016 #define __NBWIFIDCONSTANTS_H
00017
00018 #include <endian.h>
00019
00020 #define SSID_MAX_LEN 32           // Set by the IEEE 802.11 PHY specifications
00021 #define PASS_MAX_LEN 64           // Try to avoid running out of memory on small devices...
00022 #define SEC_HUMAN_MAX_LEN 32      // max length for security option string, human readable
00023 #define CIPHER_HUMAN_MAX_LEN 32   // max length for cipher option string, human readable
00024
00025 #define LIST_TABLE_NUM (0x00)
00026 #define SEC_TABLE_NUM (0x01)
00027 #define CIPH_TABLE_NUM (0x02)
00028 #define BAND_TABLE_NUM (0x03)
00029 #define CHANNEL_TABLE_NUM (0x04)
00030 #define BSSTYPE_TABLE_NUM (0x05)
00031 #define BSSID_TABLE_NUM (0x06)
00032 #define SCAN_TABLE_NUM (0x07)
00033 #define CONNMODE_TABLE_NUM (0x08)
00034 #define ITU_CTRY_TABLE_NUM (0x09)
00035 #define DEFAULT_TABLE_TABLE_NUM (0xFF)
00036
00037 #define LIST_TABLE_SIZE (10)
00038
00039 #define LIST_LABEL_MASTER "Master Table"
00040 #define LIST_LABEL_SEC "Security"
00041 #define LIST_LABEL_CIPH "Cipher"
00042 #define LIST_LABEL_BAND "Band"
00043 #define LIST_LABEL_CHANNEL "Channel"
00044 #define LIST_LABEL_BSSTYPE "BssType"
00045 #define LIST_LABEL_BSSID "BSSID"
00046 #define LIST_LABEL_SCAN "Scan Method"
00047 #define LIST_LABEL_CONNMODE "Connection Mode"
00048 #define LIST_LABEL_ITU_CTRY "Country"
00049 #define LIST_LABEL_DEFAULT "Default Values"
00050
00051 #define LIST_VALUE_MASTER LIST_TABLE_NUM
00052 #define LIST_VALUE_SEC SEC_TABLE_NUM
00053 #define LIST_VALUE_CIPH CIPH_TABLE_NUM
00054 #define LIST_VALUE_BAND BAND_TABLE_NUM
00055 #define LIST_VALUE_CHANNEL CHANNEL_TABLE_NUM
00056 #define LIST_VALUE_BSSTYPE BSSTYPE_TABLE_NUM
00057 #define LIST_VALUE_BSSID BSSID_TABLE_NUM
00058 #define LIST_VALUE_SCAN SCAN_TABLE_NUM
00059 #define LIST_VALUE_CONNMODE CONNMODE_TABLE_NUM
00060 #define LIST_VALUE_ITU_CTRY ITU_CTRY_TABLE_NUM
00061 #define LIST_VALUE_DEFAULT DEFAULT_TABLE_TABLE_NUM
00062
00063 #define SEC_TABLE_SIZE (6)
00064
00065 #define SEC_LABEL_OPEN "Open"
00066 #define SEC_LABEL_WEP "WEP"
00067 #define SEC_LABEL_WPA "WPA"
00068 #define SEC_LABEL_WPA2 "WPA2"
00069 #define SEC_LABEL_WPS "WPS"
00070 #define SEC_LABEL_UNKNOWN "Unknown"
00071
00077 #define SEC_VALUE_OPEN (0x00)
00078 #define SEC_VALUE_WEP (0x01)
00079 #define SEC_VALUE_WPA (0x02)
00080 #define SEC_VALUE_WPA2 (0x03)
00081 #define SEC_VALUE_WPS (0x04)
00082 #define SEC_VALUE_ANY (0xFE)
00083 #define SEC_VALUE_UNKNOWN (0xFF)
```

```
00086 #define CIPH_TABLE_SIZE (5)
00087
00088 #define CIPH_LABEL_NONE "None"
00089 #define CIPH_LABEL_TKIP "TKIP"
00090 #define CIPH_LABEL_AES "AES"
00091 #define CIPH_LABEL_MIXED "Mixed"
00092 #define CIPH_LABEL_UNKNOWN "Unknown"
00093
00099 #define CIPH_VALUE_NONE (0x00)
00100 #define CIPH_VALUE_TKIP (0x01)
00101 #define CIPH_VALUE_AES (0x02)
00102 #define CIPH_VALUE_MIXED (0x03)
00103 #define CIPH_VALUE_UNKNOWN (0xFF)
00106 /***
00107  * Option list band values
00108  * \section BANDSECTION
00109  * @anchor OPTIONLISTBAND
00110  */
00111 #define BAND_TABLE_SIZE (2)
00112
00113 #define BAND_LABEL_5_GHZ "5 GHz"
00114 #define BAND_LABEL_2_4_GHZ "2.4 GHz"
00115
00116 #define BAND_VALUE_5_GHZ (0x00)
00117 #define BAND_VALUE_2_4_GHZ (0x01)
00118
00119 #define BSSTYPE_TABLE_SIZE (4)
00120
00121 #define BSSTYPE_LABEL_INFR "Infrastructure"
00122 #define BSSTYPE_LABEL_ADHOC "AdHoc"
00123 #define BSSTYPE_LABEL_ANY "Any"
00124 #define BSSTYPE_LABEL_UNKNOWN "Unknown"
00125
00132 #define BSSTYPE_VALUE_INFR (0x00)
00133 #define BSSTYPE_VALUE_ADHOC (0x01)
00134 #define BSSTYPE_VALUE_ANY (0x02)
00135 #define BSSTYPE_VALUE_UNKNOWN (0xFF)
00138 #define SCAN_TABLE_SIZE (2)
00139
00140 #define SCAN_LABEL_ACTIVE "Active"
00141 #define SCAN_LABEL_PASSIVE "Passive"
00142
00143 #define SCAN_VALUE_ACTIVE (0x00)
00144 #define SCAN_VALUE_PASSIVE (0x01)
00145
00146 #define CONNMODE_TABLE_SIZE (2)
00147
00148 #define CONNMODE_LABEL_CLIENT "Client"
00149 #define CONNMODE_LABEL_AP "Access Point"
00150
00151 #define CONNMODE_VALUE_CLIENT (0x00)
00152 #define CONNMODE_VALUE_AP (0x01)
00153
00154 #define DEFAULT_TABLE_TABLE_SIZE (LIST_TABLE_SIZE - 1)
00155
00156 #define DEFAULT_TABLE_LABEL_SEC SEC_VALUE_WPA2
00157 #define DEFAULT_TABLE_LABEL_CIPH CIPH_VALUE_AES
00158 #define DEFAULT_TABLE_LABEL_BAND BAND_VALUE_2_4_GHZ
00159 #define DEFAULT_TABLE_LABEL_CHANNEL (0x06)
00160 #define DEFAULT_TABLE_LABEL_BSSTYPE BSSTYPE_VALUE_ANY
00161 #define DEFAULT_TABLE_LABEL_BSSID (0x00)
00162 #define DEFAULT_TABLE_LABEL_SCAN SCAN_VALUE_PASSIVE
00163 #define DEFAULT_TABLE_LABEL_CONNMODE CONNMODE_VALUE_CLIENT
00164 #define DEFAULT_TABLE_LABEL_ITU_CTRY (0x00)
00165
00166 #define DEFAULT_TABLE_VALUE_SEC LIST_VALUE_SEC
00167 #define DEFAULT_TABLE_VALUE_CIPH LIST_VALUE_CIPH
00168 #define DEFAULT_TABLE_VALUE_BAND LIST_VALUE_BAND
00169 #define DEFAULT_TABLE_VALUE_CHANNEL LIST_VALUE_CHANNEL
00170 #define DEFAULT_TABLE_VALUE_BSSTYPE LIST_VALUE_BSSTYPE
00171 #define DEFAULT_TABLE_VALUE_BSSID LIST_VALUE_BSSID
00172 #define DEFAULT_TABLE_VALUE_SCAN LIST_VALUE_SCAN
00173 #define DEFAULT_TABLE_VALUE_CONNMODE LIST_VALUE_CONNMODE
00174 #define DEFAULT_TABLE_VALUE_ITU_CTRY LIST_VALUE_ITU_CTRY
00175
00181 #define CONFIG_ERR_SUCCESS 0
00182 #define CONFIG_ERR_MSG_LENGTH -1
00183 #define CONFIG_ERR_SSID_LEN_SHORT -2
00184 #define CONFIG_ERR_SSID_LEN_LONG -3
00185 #define CONFIG_ERR_PASSWD_LEN_LONG -4
00186 #define CONFIG_ERR_INVALID_TABLE -5
00187 #define CONFIG_ERR_INVALID_OPTION -6
00188 #define CONFIG_ERR_BSSID_OVERRUN -7
00189 #define CONFIG_ERR_BSSID_UNDERRUN -8
00190 #define CONFIG_ERR_MULTI_CHANNEL -9
00191 #define CONFIG_ERR_CONNECTED -10
00192 #define CONFIG_ERR_UNKNOWN 1
```

```
00200 #define CONNECT_ERR_SUCCESS 0
00201 #define CONNECT_ERR_NOT_CONFIG -1
00202 #define CONNECT_ERR_INVALID_CONFIG_NUM -2
00203 #define CONNECT_ERR_CONNECTED -3
00204 #define CONNECT_ERR_SSID_NOT_FOUND -4
00205 #define CONNECT_ERR_BSSID_NOT_FOUND -5
00206 #define CONNECT_ERR_SEC_MISMATCH -6
00207 #define CONNECT_ERR_CIPH_MISMATCH -7
00208 #define CONNECT_ERR_INVALID_KEY -8
00209 #define CONNECT_ERR_UNKNOWN 1
00217 #define SCAN_ERR_SUCCESS 0
00218 #define SCAN_ERR_MSG_LENGTH -1
00219 #define SCAN_ERR_IN_PROGRESS -2
00220 #define SCAN_ERR_SSID_LEN_LONG -3
00221 #define SCAN_ERR_INVALID_TABLE -4
00222 #define SCAN_ERR_INVALID_OPTION -5
00223 #define SCAN_ERR_TOO_MANY_CHANNELS -6
00224 #define SCAN_ERR_UNKNOWN 1
00232 #define SAVE_CONF_ERR_SUCCESS 0
00233 #define SAVE_CONF_ERR_INVALID_CONFIG_NUM -1
00234 #define SAVE_CONF_ERR_NOT_CONFIGURED -2
00235 #define SAVE_CONF_ERR_UNKNOWN 1
00238 #define DRIVER_ERR_SUCCESS 0
00239 #define DRIVER_ERR_UNKNOWN 1
00240
00241 #define GETOPT_ERR_SUCCESS 0
00242 #define GETOPT_ERR_INVALID_TABLE -1
00243 #define GETOPT_ERR_UNKNOWN 1
00244
00245 #define CONN_STATE_SCAN_FOR_CONN -1
00246 #define CONN_STATE_NOT_CONNECTED 0
00247 #define CONN_STATE_CONNECTED 1
00248
00249 #define MIN_FW_MAJOR_VER 0x00
00250 #define MIN_FW_MINOR_VER 0x00
00251 #define MAX_FW_MAJOR_VER 0xFF
00252 #define MAX_FW_MINOR_VER 0xFF
00253
00254 namespace NB
00255 {
00256 namespace Constants
00257 {
00261 typedef enum
00262 {
00263     TaskStart_Err_NoError = 0,
00264     TaskStart_Err_Running = -1,
00265     TaskStart_Err_NotRegistered = -2,
00266 } TaskStartError;
00267
00271 typedef enum
00272 {
00273     TaskKill_Err_NoError = 0,
00274     TaskKill_Err_NotRunning = -1,
00275 } TaskKillError;
00276
00280 typedef enum
00281 {
00282     Connect_Success = 0,
00283     Connect_BadPass = -1,
00284     Connect_NoAPFound = -2,
00285     Connect_BSSID_NotFound = -3,
00286     Connect_TimedOut = -4,
00287 } ConnectResult;
00288
00292 typedef enum
00293 {
00294     Band_All = 0,
00295     Band_2_4GHz = 1,
00296     Band_5GHz = 2,
00297 } RadioBand;
00298
00302 typedef enum
00303 {
00304     BssType_Infrastructure = 0,
00305     BssType_AdHoc = 1,
00306     BssType_Any = 63,
00307     BssType_Unknown = 255,
00308 } BssType;
00309
00313 typedef enum
00314 {
00315     Scan_Passive = 0,
00316     Scan_Active = 1,
00317 } ScanMethods;
00318 }   // namespace Constants
00319
00327 namespace Error
```

```
00328 {
00329
00336 typedef enum
00337 {
00338     NoError = 0,
00339     Timeout = -256,
00340     BusTimeout = -257,
00341     InvalidArgument = -258,
00342     TooManyPendingCommands = -259,
00343     InvalidRequest = -512,
00344 } GeneralErrors;
00345
00350 namespace Init
00351 {
00352
00357 typedef enum
00358 {
00359     Success = 0,
00360     AlreadyInit = -1,
00361     NoDevice = -2,
00362     InvalidInfo = -3,
00363     DevFirmVer = -4,
00364     DevHwVer = -5,
00365     OptionTables = -6,
00366 } InitializationErrors;
00367 }   // namespace Init
00368
00373 namespace Scan
00374 {
00375
00380 typedef enum
00381 {
00382     Success = 0,
00383     NotInitialized = -1,
00384     InProgress = -2,
00385     Option = -3,
00386 } ScanErrors;
00387 }   // namespace Scan
00388
00393 namespace Connect
00394 {
00395
00400 typedef enum
00401 {
00402     Success = 0,
00403     NotInitialized = -1,
00404     AlreadyConnected = -2,
00405     Option = -3,
00406     CouldNotConfig = -4,
00407     SSID_NotFound = -5,
00408     BSSID_NotFound = -6,
00409     Sec_NotFound = -7,
00410     Cipher_NotFound = -8,
00411     ConnectFailed = -9,
00412 } ConnectErrors;
00413 }   // namespace Connect
00414 }   // namespace Error
00415
00416 // We use __COUNTER__ to be able to calculate the total number of
00417 //  country codes. However, since it must not affect the value of the code
00418 //  itself, we perform the '& 0x0' which will cause the compiler to helpfully
00419 //  remove the __COUNTER__ from the resulting constant.
00420 #ifdef NB_BIG_ENDIAN
00421 #define CTRY_CODE(a, b, rev) (((uint8_t)(b)) + (((uint8_t)(a)) « 8) + (((uint8_t)(rev)) « 24) +
      (__COUNTER__ & 0x0))
00422 #elif defined NB_LITTLE_ENDIAN
00423 #define CTRY_CODE(a, b, rev) (((uint8_t)(a)) + (((uint8_t)(b)) « 8) + (((uint8_t)(rev)) « 24) +
      (__COUNTER__ & 0x0))
00424 #else
00425 #error Must define device endianess
00426 #endif
00427 namespace ITU_Country
00428 {
00429 enum
00430 {
00431     _COUNTRY_BASE = __COUNTER__
00432 };
00433
00434 typedef enum
00435 {
00436     Afghanistan = CTRY_CODE('A', 'F', 0),
00437     Albania = CTRY_CODE('A', 'L', 0),
00438     Algeria = CTRY_CODE('D', 'Z', 0),
00439     American_Samoa = CTRY_CODE('A', 'S', 0),
00440     Angola = CTRY_CODE('A', 'O', 0),
00441     Anguilla = CTRY_CODE('A', 'I', 0),
00442     Antigua_And_Barbuda = CTRY_CODE('A', 'G', 0),
```

```
00443     Argentina = CTRY_CODE('A', 'R', 0),
00444     Armenia = CTRY_CODE('A', 'M', 0),
00445     Aruba = CTRY_CODE('A', 'W', 0),
00446     Australia = CTRY_CODE('A', 'U', 0),
00447     Austria = CTRY_CODE('A', 'T', 0),
00448     Azerbaijan = CTRY_CODE('A', 'Z', 0),
00449     Bahamas = CTRY_CODE('B', 'S', 0),
00450     Bahrain = CTRY_CODE('B', 'H', 0),
00451     Baker_Island = CTRY_CODE('0', 'B', 0),
00452     Bangladesh = CTRY_CODE('B', 'D', 0),
00453     Barbados = CTRY_CODE('B', 'B', 0),
00454     Belarus = CTRY_CODE('B', 'Y', 0),
00455     Belgium = CTRY_CODE('B', 'E', 0),
00456     Belize = CTRY_CODE('B', 'Z', 0),
00457     Benin = CTRY_CODE('B', 'J', 0),
00458     Bermuda = CTRY_CODE('B', 'M', 0),
00459     Bhutan = CTRY_CODE('B', 'T', 0),
00460     Bolivia = CTRY_CODE('B', 'O', 0),
00461     Bosnia_And_Herzegovina = CTRY_CODE('B', 'A', 0),
00462     Botswana = CTRY_CODE('B', 'W', 0),
00463     Brazil = CTRY_CODE('B', 'R', 0),
00464     British_Indian_Ocean_Territory = CTRY_CODE('I', 'O', 0),
00465     Brunei_Darussalam = CTRY_CODE('B', 'N', 0),
00466     Bulgaria = CTRY_CODE('B', 'G', 0),
00467     Burkina_Faso = CTRY_CODE('B', 'F', 0),
00468     Burundi = CTRY_CODE('B', 'I', 0),
00469     Cambodia = CTRY_CODE('K', 'H', 0),
00470     Cameroon = CTRY_CODE('C', 'M', 0),
00471     Canada = CTRY_CODE('C', 'A', 0),
00472     Cape_Verde = CTRY_CODE('C', 'V', 0),
00473     Cayman_Islands = CTRY_CODE('K', 'Y', 0),
00474     Central_African_Republic = CTRY_CODE('C', 'F', 0),
00475     Chad = CTRY_CODE('T', 'D', 0),
00476     Chile = CTRY_CODE('C', 'L', 0),
00477     China = CTRY_CODE('C', 'N', 0),
00478     Christmas_Island = CTRY_CODE('C', 'X', 0),
00479     Colombia = CTRY_CODE('C', 'O', 0),
00480     Comoros = CTRY_CODE('K', 'M', 0),
00481     Congo = CTRY_CODE('C', 'G', 0),
00482     Congo_The_Democratic_Republic_Of_The = CTRY_CODE('C', 'D', 0),
00483     Costa_Rica = CTRY_CODE('C', 'R', 0),
00484     Cote_Divoire = CTRY_CODE('C', 'I', 0),
00485     Croatia = CTRY_CODE('H', 'R', 0),
00486     Cuba = CTRY_CODE('C', 'U', 0),
00487     Cyprus = CTRY_CODE('C', 'Y', 0),
00488     Czech_Republic = CTRY_CODE('C', 'Z', 0),
00489     Denmark = CTRY_CODE('D', 'K', 0),
00490     Djibouti = CTRY_CODE('D', 'J', 0),
00491     Dominica = CTRY_CODE('D', 'M', 0),
00492     Dominican_Republic = CTRY_CODE('D', 'O', 0),
00493     Down_Under = CTRY_CODE('A', 'U', 0),
00494     Ecuador = CTRY_CODE('E', 'C', 0),
00495     Egypt = CTRY_CODE('E', 'G', 0),
00496     El_Salvador = CTRY_CODE('S', 'V', 0),
00497     Equatorial_Guinea = CTRY_CODE('G', 'Q', 0),
00498     Eritrea = CTRY_CODE('E', 'R', 0),
00499     Estonia = CTRY_CODE('E', 'E', 0),
00500     Ethiopia = CTRY_CODE('E', 'T', 0),
00501     Falkland_Islands_Malvinas = CTRY_CODE('F', 'K', 0),
00502     Faroe_Islands = CTRY_CODE('F', 'O', 0),
00503     Fiji = CTRY_CODE('F', 'J', 0),
00504     Finland = CTRY_CODE('F', 'I', 0),
00505     France = CTRY_CODE('F', 'R', 0),
00506     French_Guina = CTRY_CODE('G', 'F', 0),
00507     French_Polynesia = CTRY_CODE('P', 'F', 0),
00508     French_Southern_Territories = CTRY_CODE('T', 'F', 0),
00509     Gabon = CTRY_CODE('G', 'A', 0),
00510     Gambia = CTRY_CODE('G', 'M', 0),
00511     Georgia = CTRY_CODE('G', 'E', 0),
00512     Germany = CTRY_CODE('D', 'E', 0),
00513     Ghana = CTRY_CODE('G', 'H', 0),
00514     Gibraltar = CTRY_CODE('G', 'I', 0),
00515     Greece = CTRY_CODE('G', 'R', 0),
00516     Grenada = CTRY_CODE('G', 'D', 0),
00517     Guadeloupe = CTRY_CODE('G', 'P', 0),
00518     Guam = CTRY_CODE('G', 'U', 0),
00519     Guatemala = CTRY_CODE('G', 'T', 0),
00520     Guernsey = CTRY_CODE('G', 'G', 0),
00521     Guinea = CTRY_CODE('G', 'N', 0),
00522     Guinea_Bissau = CTRY_CODE('G', 'W', 0),
00523     Guyana = CTRY_CODE('G', 'Y', 0),
00524     Haiti = CTRY_CODE('H', 'T', 0),
00525     Holy_See_Vatican_City_State = CTRY_CODE('V', 'A', 0),
00526     Honduras = CTRY_CODE('H', 'N', 0),
00527     Hong_Kong = CTRY_CODE('H', 'K', 0),
00528     Hungary = CTRY_CODE('H', 'U', 0),
00529     Iceland = CTRY_CODE('I', 'S', 0),
```

```
00530     India = CTRY_CODE('I', 'N', 0),
00531     Indonesia = CTRY_CODE('I', 'D', 0),
00532     Iran_Islamic_Republic_Of = CTRY_CODE('I', 'R', 0),
00533     Iraq = CTRY_CODE('I', 'Q', 0),
00534     Ireland = CTRY_CODE('I', 'E', 0),
00535     Israel = CTRY_CODE('I', 'L', 0),
00536     Italy = CTRY_CODE('I', 'T', 0),
00537     Jamaica = CTRY_CODE('J', 'M', 0),
00538     Japan = CTRY_CODE('J', 'P', 2),
00539     Jersey = CTRY_CODE('J', 'E', 0),
00540     Jordan = CTRY_CODE('J', 'O', 0),
00541     Kazakhstan = CTRY_CODE('K', 'Z', 0),
00542     Kenya = CTRY_CODE('K', 'E', 0),
00543     Kiribati = CTRY_CODE('K', 'I', 0),
00544     Korea_Republic_Of = CTRY_CODE('K', 'R', 0),
00545     Kosovo = CTRY_CODE('0', 'A', 0),
00546     Kuwait = CTRY_CODE('K', 'W', 0),
00547     Kyrgyzstan = CTRY_CODE('K', 'G', 0),
00548     Lao_Peoples_Democratic_Repubic = CTRY_CODE('L', 'A', 0),
00549     Latvia = CTRY_CODE('L', 'V', 0),
00550     Lebanon = CTRY_CODE('L', 'B', 0),
00551     Lesotho = CTRY_CODE('L', 'S', 0),
00552     Liberia = CTRY_CODE('L', 'R', 0),
00553     Libyan_Arab_Jamahiriya = CTRY_CODE('L', 'Y', 0),
00554     Liechtenstein = CTRY_CODE('L', 'I', 0),
00555     Lithuania = CTRY_CODE('L', 'T', 0),
00556     Luxembourg = CTRY_CODE('L', 'U', 0),
00557     Macao = CTRY_CODE('M', 'O', 0),
00558     Macedonia_Former_Yugoslav_Republic_Of = CTRY_CODE('M', 'K', 0),
00559     Madagascar = CTRY_CODE('M', 'G', 0),
00560     Malawi = CTRY_CODE('M', 'W', 0),
00561     Malaysia = CTRY_CODE('M', 'Y', 0),
00562     Maldives = CTRY_CODE('M', 'V', 0),
00563     Mali = CTRY_CODE('M', 'L', 0),
00564     Malta = CTRY_CODE('M', 'T', 0),
00565     Man_Isle_Of = CTRY_CODE('I', 'M', 0),
00566     Martinique = CTRY_CODE('M', 'Q', 0),
00567     Mauritania = CTRY_CODE('M', 'R', 0),
00568     Mauritius = CTRY_CODE('M', 'U', 0),
00569     Mayotte = CTRY_CODE('Y', 'T', 0),
00570     Mexico = CTRY_CODE('M', 'X', 0),
00571     Micronesia_Federated_States_Of = CTRY_CODE('F', 'M', 0),
00572     Moldova_Republic_Of = CTRY_CODE('M', 'D', 0),
00573     Monaco = CTRY_CODE('M', 'C', 0),
00574     Mongolia = CTRY_CODE('M', 'N', 0),
00575     Montenegro = CTRY_CODE('M', 'E', 0),
00576     Montserrat = CTRY_CODE('M', 'S', 0),
00577     Morocco = CTRY_CODE('M', 'A', 0),
00578     Mozambique = CTRY_CODE('M', 'Z', 0),
00579     Myanmar = CTRY_CODE('M', 'M', 0),
00580     Namibia = CTRY_CODE('N', 'A', 0),
00581     Nauru = CTRY_CODE('N', 'R', 0),
00582     Nepal = CTRY_CODE('N', 'P', 0),
00583     Netherlands = CTRY_CODE('N', 'L', 0),
00584     Netherlands_Antilles = CTRY_CODE('A', 'N', 0),
00585     New_Caledonia = CTRY_CODE('N', 'C', 0),
00586     New_Zealand = CTRY_CODE('N', 'Z', 0),
00587     Nicaragua = CTRY_CODE('N', 'I', 0),
00588     Niger = CTRY_CODE('N', 'E', 0),
00589     Nigeria = CTRY_CODE('N', 'G', 0),
00590     Norfolk_Island = CTRY_CODE('N', 'F', 0),
00591     Northern_Mariana_Islands = CTRY_CODE('M', 'P', 0),
00592     Norway = CTRY_CODE('N', 'O', 0),
00593     Oman = CTRY_CODE('O', 'M', 0),
00594     Pakistan = CTRY_CODE('P', 'K', 0),
00595     Palau = CTRY_CODE('P', 'W', 0),
00596     Panama = CTRY_CODE('P', 'A', 0),
00597     Papua_New_Guinea = CTRY_CODE('P', 'G', 0),
00598     Paraguay = CTRY_CODE('P', 'Y', 0),
00599     Peru = CTRY_CODE('P', 'E', 0),
00600     Philippines = CTRY_CODE('P', 'H', 0),
00601     Poland = CTRY_CODE('P', 'L', 0),
00602     Portugal = CTRY_CODE('P', 'T', 0),
00603     Pueto_Rico = CTRY_CODE('P', 'R', 0),
00604     Qatar = CTRY_CODE('Q', 'A', 0),
00605     Reunion = CTRY_CODE('R', 'E', 0),
00606     Romania = CTRY_CODE('R', 'O', 0),
00607     Russian_Federation = CTRY_CODE('R', 'U', 0),
00608     Rwanda = CTRY_CODE('R', 'W', 0),
00609     Saint_Kitts_And_Nevis = CTRY_CODE('K', 'N', 0),
00610     Saint_Lucia = CTRY_CODE('L', 'C', 0),
00611     Saint_Pierre_And_Miquelon = CTRY_CODE('P', 'M', 0),
00612     Saint_Vincent_And_The_Grenadines = CTRY_CODE('V', 'C', 0),
00613     Samoa = CTRY_CODE('W', 'S', 0),
00614     Sanit_Martin_Sint_Marteen = CTRY_CODE('M', 'F', 0),
00615     Sao_Tome_And_Principe = CTRY_CODE('S', 'T', 0),
00616     Saudi_Arabia = CTRY_CODE('S', 'A', 0),
```

```
00617     Senegal = CTRY_CODE('S', 'N', 0),
00618     Serbia = CTRY_CODE('R', 'S', 0),
00619     Seychelles = CTRY_CODE('S', 'C', 0),
00620     Sierra_Leone = CTRY_CODE('S', 'L', 0),
00621     Singapore = CTRY_CODE('S', 'G', 0),
00622     Slovakia = CTRY_CODE('S', 'K', 0),
00623     Slovenia = CTRY_CODE('S', 'I', 0),
00624     Solomon_Islands = CTRY_CODE('S', 'B', 0),
00625     Somalia = CTRY_CODE('S', 'O', 0),
00626     South_Africa = CTRY_CODE('Z', 'A', 0),
00627     Spain = CTRY_CODE('E', 'S', 0),
00628     Sri_Lanka = CTRY_CODE('L', 'K', 0),
00629     Suriname = CTRY_CODE('S', 'R', 0),
00630     Swaziland = CTRY_CODE('S', 'Z', 0),
00631     Sweden = CTRY_CODE('S', 'E', 0),
00632     Switzerland = CTRY_CODE('C', 'H', 0),
00633     Syrian_Arab_Republic = CTRY_CODE('S', 'Y', 0),
00634     Taiwan_Province_Of_China = CTRY_CODE('T', 'W', 0),
00635     Tajikistan = CTRY_CODE('T', 'J', 0),
00636     Tanzania_United_Republic_Of = CTRY_CODE('T', 'Z', 0),
00637     Thailand = CTRY_CODE('T', 'H', 0),
00638     Togo = CTRY_CODE('T', 'G', 0),
00639     Tonga = CTRY_CODE('T', 'O', 0),
00640     Trinidad_And_Tobago = CTRY_CODE('T', 'T', 0),
00641     Tunisia = CTRY_CODE('T', 'N', 0),
00642     Turkey = CTRY_CODE('T', 'R', 0),
00643     Turkmenistan = CTRY_CODE('T', 'M', 0),
00644     Turks_And_Caicos_Islands = CTRY_CODE('T', 'C', 0),
00645     Tuvalu = CTRY_CODE('T', 'V', 0),
00646     Uganda = CTRY_CODE('U', 'G', 0),
00647     Ukraine = CTRY_CODE('U', 'A', 0),
00648     United_Arab_Emirates = CTRY_CODE('A', 'E', 0),
00649     United_Kingdom = CTRY_CODE('G', 'B', 0),
00650     United_States = CTRY_CODE('U', 'S', 0),
00651     United_States_Rev4 = CTRY_CODE('U', 'S', 4),
00652     United_States_No_Dfs = CTRY_CODE('Q', '2', 0),
00653     United_States_Minor_Outlying_Islands = CTRY_CODE('U', 'M', 0),
00654     Uruguay = CTRY_CODE('U', 'Y', 0),
00655     Uzbekistan = CTRY_CODE('U', 'Z', 0),
00656     Vanuatu = CTRY_CODE('V', 'U', 0),
00657     Venezuela = CTRY_CODE('V', 'E', 0),
00658     Viet_Nam = CTRY_CODE('V', 'N', 0),
00659     Virgin_Islands_British = CTRY_CODE('V', 'G', 0),
00660     Virgin_Islands_Us = CTRY_CODE('V', 'I', 0),
00661     Wallis_And_Futuna = CTRY_CODE('W', 'F', 0),
00662     West_Bank = CTRY_CODE('0', 'C', 0),
00663     Western_Sahara = CTRY_CODE('E', 'H', 0),
00664     World_Wide_Xx = CTRY_CODE('X', 'X', 0),
00665     Yemen = CTRY_CODE('Y', 'E', 0),
00666     Zambia = CTRY_CODE('Z', 'M', 0),
00667     Zimbabwe = CTRY_CODE('Z', 'W', 0),
00668 } CountryCode_t;
00669
00670 enum
00671 {
00672     NUM_COUNTRIES = __COUNTER__ - _COUNTRY_BASE
00673 };
00674 }   // namespace ITU_Country
00675 }   // namespace NB
00676
00677 #endif /* ----- #ifndef __NBWIFIDCONSTANTS_H  ----- */
00678
```

24.279 nbWifiDebug.h File Reference

```
#include <stdio.h>
#include <syslog.h>
#include <utils.h>
#include <buffers.h>
```

24.280 nbWifiDebug.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef DBIPRINTF_H_
```

```
00010 #define DBIPRINTF_H_
00011
00012 #include <stdio.h>
00013 #include <syslog.h>
00014 #include <utils.h>
00015 #include <buffers.h>
00016 #include <syslog.h>
00017
00018 // #define ENABLE_FUNC_DBG      (1)
00019 // #define ENABLE_DBIPRINTF     (1)
00020 // #define ENABLE_M2M_ERR       (1)
00021 // #define ENABLE_SPI_DBG       (1)
00022 // #define ENABLE_M2M_DBG       (1)
00023
00024 // #define WIFI_DBG_SYSLOG
00025
00026 // IPADDR4 MBP_IP(10, 1, 1, 75);
00027 // SysLogAddress = MBP_IP;
00028
00029 #ifdef ENABLE_DBIPRINTF
00030 #define dbiprintf(...) iprintf(__VA_ARGS__);
00031 //#define dbiprintf(...) SysLogVia(1, __VA_ARGS__ );
00032 #else
00033 #define dbiprintf(...)
00034 #endif
00035
00036 #ifdef ENABLE_FUNC_DBG
00037 #define FUNC_DBG(...)                \
00038     iprintf("[%s] ", __FUNCTION__); \
00039     iprintf(__VA_ARGS__);
00040 // #define FUNC_DBG(...) SysLogVia(1, "[%s] ", __FUNCTION__); SysLogVia(1, __VA_ARGS__);
00041 #else
00042 #define FUNC_DBG(...)
00043 #endif
00044
00045 #ifdef ENABLE_M2M_ERR
00046 #define M2M_ERR(...) iprintf(__VA_ARGS__);
00047 //#define M2M_ERR(...) SysLogVia(1, __VA_ARGS__ );
00048 #else
00049 #define M2M_ERR(...)
00050 #endif
00051
00052 #ifdef ENABLE_SPI_DBG
00053 #define SPI_DBG(...)        \
00054     iprintf("[SPI_DBG] "); \
00055     iprintf(__VA_ARGS__);
00056 #else
00057 #define SPI_DBG(...)
00058 #endif
00059
00060 #ifdef ENABLE_M2M_DBG
00061 // #define M2M_DBG(...) iprintf(__VA_ARGS__);
00062 #define M2M_DBG(...)                 \
00063     iprintf("[%s] ", __FUNCTION__); \
00064     iprintf(__VA_ARGS__);
00065 #else
00066 #define M2M_DBG(...)
00067 #endif
00068
00069 #define LOG_DEST 0xC0A80146   // 192.168.1.70
00070 #define SET_LOG_DEST                 \
00071     {                                \
00072         extern IPADDR SysLogAddress; \
00073         SysLogAddress = LOG_DEST;    \
00074     }
00075 #define LOG_INTF 3
00076
00077 #ifdef WIFI_DBG_SYSLOG
00078 #define BuffLog_Get_Tx(p) SysLogVia(LOG_INTF, "GET TX  - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00079 #define BuffLog_Get_Rx(p) SysLogVia(LOG_INTF, "GET RX  - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00080 #define BuffLog_Free_Tx(p) SysLogVia(LOG_INTF, "FREE TX - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00081 #define BuffLog_Free_Rx(p) SysLogVia(LOG_INTF, "FREE RX - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00082 #define BuffLog_Inc(p) SysLogVia(LOG_INTF, "INC     - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00083 #define BuffLog_Write(p) SysLogVia(LOG_INTF, "WRITE   - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00084 #define BuffLog_Read(p) SysLogVia(LOG_INTF, "READ    - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
    __FILE__);
00085 #define LOGME                                                          \
00086     iprintf("[LOGME] Reached line %d in file %s\n", __LINE__, __FILE__); \
00087     OSTimeDly(2);
00088 #else
00089 #define BuffLog_Get_Tx(p)
```

```
00090 #define BuffLog_Get_Rx(p)
00091 #define BuffLog_Free_Tx(p)
00092 #define BuffLog_Free_Rx(p)
00093 #define BuffLog_Inc(p)
00094 #define BuffLog_Write(p)
00095 #define BuffLog_Read(p)
00096 #endif
00097
00098 #endif /* DBIPRINTF_H_ */
```

24.281 wifi.h File Reference

NetBurner Wifi API.

```
#include <wifi/wifiDriver.h>
#include <wifi/wilc/nbWifiWilcSpi.h>
#include <wifi/nbwifi/nbWifiSpi.h>
#include <wifi/nbwifi/nbWifiSerial.h>
```

Functions

- nbWifiScanResult * WifiInitScan_SPI (int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver over the SPI bus, and performs an AP scan.
- int WifiInitScanAndShow_SPI (int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver using the SPI bus, performs an AP scan, and prints the scan results via serial output.
- int InitWifi_SPI (const char *SSID="", const char *password="", int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver using the SPI bus, and attempts to establish the specified connection.
- int InitAP_SPI (const char *SSID="", const char *password="", uint8_t channel=NBWIFI_DEFAULT_WIFICHANNEL, int irqNum=-1, int moduleNum=-1, int csNum=-1, int connectorNum=-1, int gpioPinNum=-1, int resetPinNum=-1)

  Initializes the WiFi hardware, initializes the driver using the SPI bus, and attempts to establish the specified access point.
- nbWifiScanResult * WifiInitScan_Serial (int portNum=-1, int resetPinNum=-1, int connectorNum=-1)

  Initializes the WiFi hardware, initializes the driver using the UART interface, performs an AP scan, and prints the scan results via serial output.
- int WifiInitScanAndShow_Serial (int portNum=-1, int resetPinNum=-1, int connectorNum=-1)

  Initializes the WiFi hardware, initializes the driver using the UART interface, performs an AP scan, and prints the scan results via serial output.
- int InitWifi_Serial (const char *SSID="", const char *password="", int portNum=-1, int resetPinNum=-1, int connectorNum=-1)

  Initializes the WiFi hardware, initializes the driver using the UART interface, and attempts to establish the specified access point.
- void SetWifiSPISpeed (int busSpeed)

  Set SPI bus speed.
- void ScanAndShowNetworks ()

  Scan for surrounding access points and print the results via iprintf.
- nbWifiScanResult * ScanForNetworks ()

  Scan for surrounding access points.

24.281.1 Detailed Description

NetBurner Wifi API.

24.282 wifi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef __NBWIFI_H
00016 #define __NBWIFI_H
00017
00018 #include <wifi/wifiDriver.h>
00019 #include <wifi/wilc/nbWifiWilcSpi.h>
00020 #include <wifi/nbwifi/nbWifiSpi.h>
00021 #include <wifi/nbwifi/nbWifiSerial.h>
00022
00023 /*****************************************************************************
00024  *
00025  * Default WiFi Initialization Structure
00026  *
00027  *  Uncomment one macro to use the default pin/irq settings for that WiFi
00028  *  module. The default pin/irq settings are defined in the NBWIFI Getting
00029  *  Started Guide. To customize the pin/irq settings, include the WiFiInit
00030  *  structure with customized settings in your application.
00031  *
00032  *****************************************************************************/
00033 #define ENABLE_NBWIFIIN_SPI
00034 // #define ENABLE_NBWIFIIN_SERIAL
00035 // #define ENABLE_NBWIFIWILC
00036
00037 extern const int NBWIFI_PLAT_DEFAULT_IRQNUM;
00038 extern const int NBWIFI_PLAT_DEFAULT_SPINUM;
00039 extern const int NBWIFI_PLAT_DEFAULT_CSNUM;
00040 extern const int NBWIFI_PLAT_DEFAULT_CONNUM;
00041 extern const int NBWIFI_PLAT_DEFAULT_PINNUM;
00042 extern const int NBWIFI_PLAT_DEFAULT_RESETPIN;
00043 extern const int NBWIFI_PLAT_DEFAULT_UART;
00044 extern const int NBWIFI_PLAT_DEFAULT_CHIPEN;
00045
00046 #define NBWIFI_DEFAULT_WIFICHANNEL 6
00047
00076 nbWifiScanResult *WifiInitScan_SPI(int irqNum = -1,
00077                                    int moduleNum = -1,
00078                                    int csNum = -1,
00079                                    int connectorNum = -1,
00080                                    int gpioPinNum = -1,
00081                                    int resetPinNum = -1);
00082
00104 int WifiInitScanAndShow_SPI(int irqNum = -1,
00105                             int moduleNum = -1,
00106                             int csNum = -1,
00107                             int connectorNum = -1,
00108                             int gpioPinNum = -1,
00109                             int resetPinNum = -1);
00110
00136 int InitWifi_SPI(const char *SSID = "",
00137                  const char *password = "",
00138                  int irqNum = -1,
00139                  int moduleNum = -1,
00140                  int csNum = -1,
00141                  int connectorNum = -1,
00142                  int gpioPinNum = -1,
00143                  int resetPinNum = -1);
00144
00180 int InitAP_SPI(const char *SSID = "",
00181                const char *password = "",
00182                uint8_t channel = NBWIFI_DEFAULT_WIFICHANNEL,
00183                int irqNum = -1,
00184                int moduleNum = -1,
00185                int csNum = -1,
00186                int connectorNum = -1,
00187                int gpioPinNum = -1,
00188                int resetPinNum = -1);
00189
00209 nbWifiScanResult *WifiInitScan_Serial(int portNum = -1, int resetPinNum = -1, int connectorNum = -1);
00210
00227 int WifiInitScanAndShow_Serial(int portNum = -1, int resetPinNum = -1, int connectorNum = -1);
00228
00249 int InitWifi_Serial(const char *SSID = "", const char *password = "", int portNum = -1, int
      resetPinNum = -1, int connectorNum = -1);
00250
00257 void SetWifiSPISpeed(int busSpeed);
00258
00265 void ScanAndShowNetworks();
00266
00276 nbWifiScanResult *ScanForNetworks();
00277
```

```
00278 #endif /* ----- #ifndef __NBWIFI_H  ----- */
00279
```

24.283 wifiBsp.h File Reference

```
#include <wifi/wifiDriver.h>
```

24.284 wifiBsp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef __NBWIFIBSP_H
00010 #define __NBWIFIBSP_H
00011
00012 #include <wifi/wifiDriver.h>
00013
00014 #define SPI_DEFAULT_BAUD 10000000        // 10 MHz
00015 #define SPI_DEFAULT_WILC_BAUD 40000000   // 40 MHz
00016
00017 namespace NB
00018 {
00019 namespace WifiBSP
00020 {
00021
00022 enum GPIO_Slave_Select
00023 {
00024     ASSERT_SLAVE_SELECT = 0x0,
00025     DEASSERT_SLAVE_SELECT = 0x1,
00026 };
00027
00028 void SetupIOPins(int spiModuleNum, int csSpiNum, int connectorNum, int csGpioPinNum, int resetPinNum);
00029
00030 void InitSPI(int moduleNum, int csNum, uint32_t busSpeed = SPI_DEFAULT_BAUD);
00031
00032 void StartSPI(int moduleNum, uint8_t *TxBuffer, uint8_t *RxBuffer, uint32_t len, OS_SEM *SPI_SEM, bool
      keepCsAsserted);
00033
00034 void SetPin(int header, int pinNum, int state);
00035
00036 void InitIRQFunc(int IRQNum, void (*handler)(void), wifiModule wifiPlatform);
00037
00038 void ResetIRQ(int IRQNum);
00039
00040 void DisableIRQ(int IRQNum);
00041
00042 void WifiReset(int resetPinNum = -1, int connectorNum = -1, int chipEnablePinNum = -1);
00043
00044 }   // namespace WifiBSP
00045 }   // namespace NB
00046
00047 #endif /* ----- #ifndef __NBWIFIBSP_H  ----- */
```

24.285 wifiDriver.h File Reference

NetBurner Wifi API.

```
#include <buffers.h>
#include <nettypes.h>
#include <config_obj.h>
#include <netinterface.h>
#include <constants.h>
#include <utils.h>
#include <wifi/nbWifiConstants.h>
#include <wifi/nbwifi/nbWifiMsgStructs.h>
#include <wifi/wilc/microchip/driver/m2m_types.h>
#include <json_lexer.h>
```

Classes

- struct NB::Wifi::driverStatusStruct
- struct wifi_init

Functions

- void SetWifiSPISpeed (int busSpeed)

  Set SPI bus speed.
- void NB::PrintScanResult (nbWifiScanResult *scanResult)

  Print the results of one scan result using iprintf.
- ParsedJsonDataSet NB::ConvertScanResultsToJSON (nbWifiScanResult *scanResults)

  Convert the WiFi scan results into a JSON Object.

24.285.1 Detailed Description

NetBurner Wifi API.

24.285.2 Function Documentation

24.285.2.1 ConvertScanResultsToJSON()

```
ParsedJsonDataSet NB::ConvertScanResultsToJSON (
            nbWifiScanResult * scanResults )
```

Convert the WiFi scan results into a JSON Object.

Parameters

in	*scanResults*	Pointer to the scan result structure containing the scan result to be converted

Returns

ParsedJsonDataSet with the scan results if 'scanResults' is not empty. Otherwise, returns an empty ParsedJsonDataSet.

24.285.2.2 PrintScanResult()

```
void NB::PrintScanResult (
            nbWifiScanResult * scanResult )
```

Print the results of one scan result using iprintf.

Parameters

in	*scanResult*	Pointer to the scan result structure containing the scan result to be printed

24.286 wifiDriver.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef __WIFIDRIVER_H
00016 #define __WIFIDRIVER_H
```

```
00017
00018 #include <buffers.h>
00019 #include <nettypes.h>
00020 #include <config_obj.h>
00021 #include <netinterface.h>
00022 #include <constants.h>
00023 #include <utils.h>
00024 #include <wifi/nbWifiConstants.h>
00025 #include <wifi/nbwifi/nbWifiMsgStructs.h>
00026 #include <wifi/wilc/microchip/driver/m2m_types.h>
00027 #include <json_lexer.h>
00028
00029 #ifndef _STRUCT_PACKED
00030 #define _STRUCT_PACKED __attribute__((__packed__))
00031 #endif
00032
00033 #define WIFI_INTERFACE_NAME         "Wlan0"
00034 #define MAX_WIFI_INTERFACES 1
00035 #define WIFI_TASK_STACK_SIZE 4048
00036 #define WIFI_RX_TASK_BASE_PRIO 18
00037 #define WIFI_TX_FLOW_OFF_COUNT 120
00038
00039 #define CONNECT_RETRIES (5)
00040 #define DEFAULT_WIFI_SEC_ALL (255)
00041 #define DEFAULT_WIFI_CH_ALL (255)
00042
00043 #define STATUS_STALENESS_LIMIT (1 * TICKS_PER_SECOND)
00044 #define SCAN_STALENESS_LIMIT (20 * TICKS_PER_SECOND)
00045
00046 #define CONNECT_TIMEOUT (60 * TICKS_PER_SECOND)
00047 #define RECONNECT_TIMEOUT (20 * TICKS_PER_SECOND)
00048 #define SCAN_TIMEOUT (15 * TICKS_PER_SECOND)
00049
00050 #define DRIVER_STATUS_CHECK_INTERVAL (5 * TICKS_PER_SECOND)
00051 #define STATUS_AUTO_REQUEST_LIMIT (5 * TICKS_PER_SECOND)
00052 #define STATUS_ERROR_TIMEOUT (30 * TICKS_PER_SECOND)
00053 #define WIFI_RX_ERROR_TIMEOUT (300 * TICKS_PER_SECOND)
00054 #define MAX_RX_TASK_TIMEOUT_COUNT (2)
00055
00056 #define PHY_BRDCST_DEST(x)
      \
00057     ((((uint8_t *)(x))[0] == 0xFF) && (((uint8_t *)(x))[1] == 0xFF) && (((uint8_t *)(x))[2] == 0xFF)
      && (((uint8_t *)(x))[3] == 0xFF) && \
00058      (((uint8_t *)(x))[4] == 0xFF) & (((uint8_t *)(x))[5] == 0xFF))
00059
00060 void SetWifiSPISpeed(int busSpeed);
00061
00062 uint32_t FifoSize(const OS_FIFO *fifo);
00063
00064 struct _STRUCT_PACKED nbWifiScanResult
00065 {
00066     uint8_t lastAndBand;
00067     uint8_t bssType;
00068     uint8_t channel;
00069     uint8_t security;
00070     uint8_t cipher;
00071     uint8_t ssidLength;
00072     int16_t rssi;
00073     MACADR bssid;
00074     char ssid[SSID_MAX_LEN + 1];
00075     struct nbWifiScanResult *next;
00076 };
00077
00078 enum wifiModule
00079 {
00080     NBWIFIIN = 0,
00081     NBWIFIWILC
00082 };
00083
00084 enum busType
00085 {
00086     SPI = 0,
00087     SERIAL
00088 };
00089
00090 namespace NB
00091 {
00092 struct nbWifiDeviceInfo
00093 {
00094     uint8_t hardwareMajorRev;
00095     uint8_t hardwareMinorRev;
00096     uint8_t softwareMajorRev;
00097     uint8_t softwareMinorRev;
00098     MACADR hwAddr;
00099     uint8_t hardwareTypeLength;
00100     char hardwareType[];   // Placeholder for char array pobe_inter
00101 };
```

```
00102
00103 struct nbWifiScanResultList
00104 {
00105     PoolPtr headBuffer;
00106     PoolPtr tailBuffer;
00107     nbWifiScanResult *head;
00108     nbWifiScanResult *tail;
00109     uint8_t count;
00110     uint32_t tickLastUpdated;
00111 };
00112
00113 struct nbWifiConnect
00114 {
00115     char ssid[SSID_MAX_LEN + 1];
00116     uint8_t ssidLength;
00117     uint8_t security;
00118     char psk[PASS_MAX_LEN + 1];
00119     uint8_t channel;
00120     int retryCount;
00121 };
00122
00130 void PrintScanResult(nbWifiScanResult *scanResult);
00131 void PrintScanResultList(nbWifiScanResult *firstResult);
00132
00142 ParsedJsonDataSet ConvertScanResultsToJSON(nbWifiScanResult *scanResults);
00143
00144 class Wifi : public ::EtherLikeInterface
00145 {
00146    public:
00147     // NNDK 3.0: Testing config variables
00148     config_string  cfg_ssid{"", "SSID"};
00149     config_pass    cfg_key{"", "Password"};
00150     config_chooser mode{"Connection Mode", "Client", "Client,Access Point"};
00151     config_chooser security{"Security", "Any", "Open,WEP,WPA,WPA2,Any"};
00152     ConfigEndMarker;
00153
00154     volatile bool autoReconnect;
00158     struct driverStatusStruct
00159     {
00160         uint8_t connected;
00161         uint8_t ssidLength;
00162         uint16_t txPower;
00163         int16_t rssi;
00164         uint8_t band;
00165         uint8_t channel;
00166         uint32_t maxTxRate;
00167         uint8_t security;
00168         uint8_t cipher;
00169         MACADR bssid;
00170         uint8_t bssType;
00171         char ssid[SSID_MAX_LEN + 1];
00172
00173         uint32_t tickLastUpdated;
00174     };
00175
00195     virtual int ConnectToAP(const char *ssid = nullptr,
00196                             const char *passwd = "",
00197                             uint8_t retryCount = CONNECT_RETRIES,
00198                             uint8_t security = DEFAULT_WIFI_SEC_ALL) = 0;
00199
00222     virtual int ConnectToAP(nbWifiConnect connect) = 0;
00223
00252     virtual int StartAP(const char *ssid = nullptr,
00253                         const char *passwd = "",
00254                         uint8_t channel = DEFAULT_TABLE_LABEL_CHANNEL,
00255                         uint8_t security = DEFAULT_TABLE_LABEL_SEC,
00256                         uint8_t cipher = DEFAULT_TABLE_LABEL_CIPH,
00257                         uint8_t ssidLen = 0) = 0;
00258
00275     virtual int StartConfigAP(uint8_t channel = DEFAULT_TABLE_LABEL_CHANNEL) = 0;
00276
00295     virtual nbWifiScanResult *Scan(const char *ssid = nullptr, uint8_t optionCount = 0, uint16_t
      *optionList = nullptr) = 0;
00296
00303     virtual void Disconnect() = 0;
00304
00308     static Wifi *wifiDrivers[MAX_WIFI_INTERFACES];
00309
00320     static Wifi *GetDriverByInterfaceNumber(int interfaceNumber);
00321
00327     static void SetDefaultITUCountry(NB::ITU_Country::CountryCode_t country);
00336     virtual int SetITUCountry(NB::ITU_Country::CountryCode_t country) = 0;
00337
00344     virtual bool isRegistered() const = 0;
00345
00354     virtual int GetWifiInterfaceNumber() const = 0;
00355
```

```
00362     virtual int GetSystemInterfaceNumber() const = 0;
00363
00364     /************************ BEGIN Status Functions *************************/
00381     virtual int StoreSSIDPWToConfig(char *ssid, char *password, int ssidLen = 0) = 0;
00382
00389     virtual const char *GetDriverName();
00390
00403     virtual int GetSSIDFromConfig(char *returnBuf, int maxLen) = 0;
00404
00417     virtual int GetKeyFromConfig(char *returnBuf, int maxLen) = 0;
00418
00432     virtual int GetCurSSID(char *buf, int maxlen) = 0;
00433
00440     virtual MACADR GetCurBSSID() = 0;
00441
00453     virtual void GetCurrentAP(driverStatusStruct *ap) = 0;
00454
00464     virtual void GetDeviceInformation(nbWifiDeviceInfo *ap) = 0;
00465
00476     virtual bool Connected() = 0;
00477
00485     virtual int GetSignalStrength() = 0;
00486
00495     virtual int GetCurChannel() = 0;
00496
00505     virtual int GetSecurity() = 0;
00506
00515     virtual int GetCipher() = 0;
00516     /************************ END Status Functions **************************/
00517
00527     virtual void TransmitBuffer(PoolPtr txBuffer) = 0;
00528
00537     virtual bool GetLinkStatus() = 0;
00538
00551     virtual void enab_multicast(MACADR macAddress, BOOL addAddress) = 0;
00552
00561     virtual int APIStart() = 0;
00562
00563     /********************** BEGIN RXTask control methods ***********************/
00574     virtual int Start() = 0;
00575
00585     virtual void Pause() = 0;
00586
00596     virtual void Resume() = 0;
00597
00607     virtual void Kill() = 0;
00608     /********************** END RXTask control methods ***********************/
00609
00610     Wifi();
00611     Wifi(const Wifi &rhs);
00612     virtual bool UpdateSlaveFirmware(uint32_t imageLength, const uint8_t *imageBuffer) = 0;
00613
00614    protected:
00615     typedef enum
00616     {
00617         Bus_Serial = 0,
00618         Bus_SPI = 1,
00619         Bus_SDIO = 2,
00620         Bus_USB = 3,
00621         Bus_PCI_E = 4,
00622         Bus_Ethernet = 5,
00623         Bus_Internal = 6,
00624     } CommBus;
00625
00626     typedef enum
00627     {
00628         Mss_Serial = 1500,
00629         Mss_SPI = 1500,
00630         Mss_SDIO = 1500,
00631         Mss_USB = 1500,
00632         Mss_PCI_E = 1500,
00633         Mss_Ethernet = 1480,
00634         Mss_Internal = 1500,
00635     } MssSizes;
00636
00637     /* ---- Static members ---- */
00638     static bool APITaskStarted;
00639     static bool APITaskRunning;
00640     static bool APIAbortTask;
00641     static bool APIPauseTask;
00642     static OS_SEM APITaskPauseSem;
00643     static ITU_Country::CountryCode_t wifiCountry;
00644     /* ---- End Static members ---- */
00645
00646     /* ---- Data members ---- */
00647     nbWifiScanResultList ScanResults;
00648     bool scanComplete;
```

```
00649     OS_MBOX scanMailbox;
00650     /* ---- End Data members ---- */
00651
00652     void (*DisconnectCB)(Wifi *wifiIntf);
00653     driverStatusStruct connectionState;
00654     CommBus busType;
00655     OS_CRIT busTxCrit;
00656     int resetPinConnector;
00657     int resetPinNum;
00658     bool taskStarted;
00659     bool taskRunning;
00660     bool abortTask;
00661     bool pauseTask;
00662     int myWifiNum;   // WiFi Driver number, not the interface number
00663
00664     Wifi(CommBus busType, int resetPinConnector, int resetPinMum, const char *name);
00665     ~Wifi();
00666
00667     virtual bool SetBusSpeed(uint32_t busSpeed) = 0;
00668     friend void ::SetWifiSPISpeed(int busSpeed);
00669
00670     friend class WILC1000;
00671     friend class WICED;
00672 };
00673
00674 struct SSID_t
00675 {
00676     uint8_t len;
00677     uint8_t ssid[SSID_MAX_LEN + 1];
00678 };
00679 }   // namespace NB
00680
00691 typedef struct
00692 {
00693     int spiModuleNum;
00694     int connectorNum;
00695     int csNum;
00696     int csGpioPinNum;
00697     int resetPinNum;
00698     int irqNum;
00699     int serialPortNum;
00700     int chipEnablePinNum;
00701     NB::Wifi *(*GetDriver)();
00702 } wifi_init;
00703
00704 extern wifi_init wifiInit;
00705 extern NB::Wifi *theWifiIntf;
00706 extern NB::Wifi *wlan0;
00707
00708 #endif /* NBWIFIDRIVER_H_ */
00709
```

24.287 acmeRFC8555Servlet.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_ACME_H
00006 #define _NB_ACME_H
00007
00008 #include <predef.h>
00009 #include <ip.h>
00010 #include <nettypes.h>
00011
00012
00013 #include <base64.h>
00014 #include <json_lexer.h>
00015 #include <nbtime.h>
00016 #include <nbstring.h>
00017 #include <diagnostics.h>
00018 #include <webclient/http_funcs.h>
00019
00020 #include <crypto/wolfssl/wolfcrypt/ecc.h>
00021 #include <crypto/wolfssl/wolfcrypt/random.h>
00022 #include <crypto/wolfssl/wolfcrypt/sha256.h>
00023 #include <crypto/wolfssl/wolfcrypt/hash.h>
00024 #include <crypto/wolfssl/wolfcrypt/asn_public.h>
00025 #include <crypto/wolfssl/wolfcrypt/wolfmath.h>
00026 #include <crypto/certgen.h>
00027 #include <servlets.h>
00028
00029 extern const char * DefAcmeUrl;
00030
00031 const int ACME_ERR_NONE=0;
```

```
00032 const int ACME_ERR_HAL_SAVE_FAIL=-10;
00033 const int ACME_ERR_CSR_ERR=-11;
00034 const int ACME_ERR_FAIL_READ_EXPIRY=-12;
00035 const int ACME_ERR_STORED_CERT_INVALID=-14;
00036 const int ACME_FAILED_GET_DIR=-15;
00037 const int ACME_FAILED_GET_NONCE=-16;
00038 const int ACME_FAILED_GET_NONCE_DIR=-17;
00039 const int ACME_FAILED_GET_ACCOUNT=-18;
00040 const int ACME_ERR_GET_CSR_FAIL=-19;
00041 const int ACME_ERR_ORDER_FAIL=-20;
00042 const int ACME_ERR_MALLOC_FAIL=-21;
00043 const int ACME_ERR_SELF_FAIL=-22;
00044 const int ACME_ERR_DNS_FAIL=-23;
00045 const int ACME_ERR_TRANSACT_FAIL=-24;
00046
00047
00048
00049 //Forward declartions.
00050 class AcmeDataSet;
00051 class AcmeServletObject;
00052
00053
00054 //Class to hanle header processing as ACME needs both headers and body processed.
00055 class AcmeServletBuffer : public ParsedJsonDataSet
00056 {
00057 AcmeDataSet * pAcme;
00058     virtual void ProcessHeader(const char * hdr);
00059 public:
00060     AcmeServletBuffer(AcmeDataSet * pOwner) {pAcme=pOwner;}
00061
00062 };
00063
00064 class AcmeAuthItem  : public HtmlPageHandler
00065 {
00066     typedef enum {
00067             eAuthStart,
00068             eAuthCreated,
00069             eAuthRetrieved,
00070             eAuthActivated,
00071             eAuthValid,
00072             eAuthPending,
00073             eAuthInvalid
00074     } eAcmeAuthState_t;
00075
00076
00077 public:
00078 AcmeServletBuffer AuthBuffer;
00079 NBString AuthItem;//The URL from the  Order list
00080 NBString AuthActivation; //The URL for activation and status
00081 NBString AuthToken;//The Token to return
00082 NBString AuthPath;//The Path
00083 NBString AuthName;//The DNS name were doing all this for
00084 NBString UseThumb;
00085 AcmeServletObject &  Owner;
00086 int m_Index;
00087 int m_Query_Num;
00088 int m_Retry;
00089 eAcmeAuthState_t m_State;
00090
00091 //HtmlPageHandler callback used to serve up the response page
00092 //Possible this might change from is one to has one...
00093 virtual int ProcessRaw(int sock, HTTP_Request &pd);
00094
00095 AcmeAuthItem(AcmeServletObject & owner,AcmeDataSet * pData);
00096 void InitNew(const NBString & src, int index);
00097 ~AcmeAuthItem();
00098 void SendForState();
00099 bool ProcessForState();
00100 bool ProcessForTimeout();
00101 const char * GetStateCC();
00102 void clear();
00103 void SetTimeoutSecs(int secs);
00104 inline bool valid() {return (m_State==eAuthValid); };
00105
00106 };
00107
00108
00109
00110
00111 struct AcmeDataSet
00112 {
00113 AcmeServletBuffer DirListing;  //Holds the rood directoory listing of services.
00114 AcmeServletBuffer OrderResult; //Holds the order in process.
00115 AcmeServletBuffer TransactionResult; //Temp holds the transactions...
00116
00117
00118 NBString nonce;
```

```
00119 NBString jwk;
00120 NBString kid;
00121 NBString FinalizeUrl;
00122 NBString RetryAfter;
00123 NBString Location;
00124 NBString OrderUrl;
00125 AcmeAuthItem AuthItem;
00126 bool bDiag;
00127 uint8_t CertBigBuffer[16384];
00128 SimpleBufferObject sbo{CertBigBuffer,16384};
00129
00130 WC_RNG rng;
00131
00132 //Helper function Used to scan incomming headers for things of interest.
00133 bool ScanHeaderAndSet(const char * pTarget, NBString & setv,const char * hdr);
00134
00135 //Called for every header in incomming transactions.
00136 void ProcessHeader(const char * hdr);
00137
00138 AcmeDataSet(AcmeServletObject &
      owner):DirListing(this),OrderResult(this),TransactionResult(this),AuthItem(owner,this)
00139 {
00140 }
00141 ~AcmeDataSet();
00142 };
00143
00144
00145
00146
00147
00148
00149 class AcmeServletObject : public config_obj , public DiagItemClass, public WebClientServlet
00150 {
00151
00152     typedef enum {
00153         eWaitStart,
00154         eGetDirs,
00155         eFirstNonce,
00156         eDoAccount,
00157         eDoOrder,
00158         eDoAuthItemDo,
00159         eDoOrderFinalize,
00160         eDoOrderStatus,     //retry count
00161         eDoGetCert,
00162         eDoneSleeping,
00163         eErrorWaitForRetry,
00164         eRestart
00165     } eAcmeServletState_t;
00166
00167 config_string m_ServerRootUrl;
00168 ConfigEndMarker;   // No new data members below this line
00169
00170
00171 ecc_key  AccountKey;
00172 ecc_key  ServerKey;
00173 uint32_t keysize;
00174
00175 //Used by SSL to get keys etc...
00176 puint8_t m_pServerKey;
00177 puint8_t m_pServerCert;
00178
00179
00180 //State variables
00181
00182 time_t m_issued;     //When does cert expire
00183 time_t m_expiry;     //When does cert expire
00184 time_t m_retrydate;  //When shoudl we renew?
00185 NBString m_CertIssuer; //Name of entity who signed cert.
00186 NBString m_LastAction;
00187
00188
00189 AcmeDataSet  * pAcmeSet;
00190
00191 eAcmeServletState_t m_pvt_State;
00192 int m_Retry_Order;
00193 int m_Retry_Status;
00194 int m_Retry_Transaction;
00195 uint32_t m_StatusFlags;
00196
00197
00198 //The erorr state of the world.
00199 NBString err_str;
00200 int err_code;
00201
00202 //Active status of the world
00203 NBString status_str;
00204
```

```
00205
00206 private:
00207
00208 //Access the things we need to generate responses...
00209     NBString GetJwk(); //Java web key
00210     NBString GetThumb(); //Thumb print of account key
00211     bool GetCSR(NBString & s); //Generate a Certificate signing request
00212
00213
00214     void FillInReq(Cert & req);//Fillin names and details on a cert request
00215
00216
00217 //Global pointer so we can get the one and only object for SSL key handeling
00218
00219
00220 void RetryOrder();
00221
00222 void SendForState(eAcmeServletState_t state);
00223 void ProcessForState(eAcmeServletState_t state);
00224 void ProcessForTimeout(eAcmeServletState_t state);
00225
00226
00227
00228 PoolPtr PrepTransaction(const NBString & url, const char* payload="", bool bJwk=false);
00229 void StartTransaction(AcmeServletBuffer & buf,const char * dir_entry, const char* payload="", bool
      bJwk=false);
00230 void StartTransactionUrl(AcmeServletBuffer & buf,const NBString & url, const char* payload="", bool
      bJwk=false);
00231
00232
00233 //Save current active keys to storage
00234 bool SaveKeysToStorage();
00235
00236 //Make and save to storage a selfsigned cert...
00237 bool MakeSaveSelfSignedCert();
00238
00239 //Read the server cert and set the issue and expiry date
00240 //Setup cert on SSL.
00241 void UseCurrentCert();
00242
00243 //DigItemClass virtual for showing state of the world on diag item.
00244 virtual void ServeContent(int fd);
00245
00246 //WebClientServlet and Root servlet virtual func
00247 virtual int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00248
00249 //WebClientServlet  virtual function.
00250 virtual void ActionComplete(eWebClientAction_t action);
00251
00252
00253 bool LoadKeys();
00254 bool MakeKeys();
00255 void CheckCert();
00256
00257 protected:
00258 inline bool FlagIsSet(uint32_t flag){ return ((m_StatusFlags&flag)==flag); };
00259 inline bool FlagIsClear(uint32_t flag){return ((m_StatusFlags&flag)==0); };
00260 inline void SetFlag(uint32_t flag) {m_StatusFlags|=flag;};
00261 inline void ClearFlag(uint32_t flag){m_StatusFlags &=(~flag); };
00262
00263 inline eAcmeServletState_t GetState() {return m_pvt_State;};
00264 inline void SetState(eAcmeServletState_t s){m_pvt_State=s;}
00265
00266
00267
00268
00269 bool bDiag;
00270
00271
00272
00273
00274 public:
00275 //Once the system has loaded valid keys and certificate
00276 //It sleeps and does not do much.
00277 //If you want to change the names, or want to re-run
00278 //the Acme protocol for debugging/diagnostic purposes.
00279 //This function erases all keys
00280 //Erases the certificate
00281 //and restarts the whole process.
00282 void Delete_Everything_Restart();
00283
00284
00285 //Turn on diganostic printf messages to the console.
00286 void SetDiag(bool v){bDiag=v; if(pAcmeSet) pAcmeSet->bDiag=v;}
00287
00288 //Access functions:
00289 //This shows the detailed state of the Acme process.
```

```
00290 NBString GetGlobalStateString();
00291
00292 //This returns the specific state condition of the acmeservlet
00293 NBString GetStateString();
00294 const char * GetStateCC();
00295
00296 //Get the current DNSName the servlet is using.
00297 //NBString GetNames() {return (NBString)m_DnsName; };
00298
00299
00300
00301
00302 //Constructor builds the object...
00303 AcmeServletObject(const char * pUrlDir=DefAcmeUrl):
00304 config_obj(sys,"AcmeService", "The ACME RFC8555 service"),
00305 DiagItemClass("ACMEClient"),
00306 m_ServerRootUrl(pUrlDir,"AcmeServerUrl", "The URL for the acme service directory"),
00307 pAcmeSet(0),
00308 m_pvt_State(eWaitStart),
00309 m_Retry_Order(0),
00310 m_Retry_Status(0),
00311 m_Retry_Transaction(0),
00312 m_StatusFlags(0),
00313 bDiag(false)
00314 {
00315 };
00316
00317
00318 friend AcmeAuthItem;
00319 friend CertGenData *GetDataForCertGen();
00320
00321
00322
00323
00324
00325
00326 };
00327 #endif
```

24.288 aes.h

```
00001 /*NB_REVISION*/
00002
00003 /*
00004  *  FIPS-197 compliant AES implementation
00005  *
00006  *  Copyright (C) 2006-2007  Christophe Devine
00007  *
00008  *  This library is free software; you can redistribute it and/or
00009  *  modify it under the terms of the GNU Lesser General Public
00010  *  License, version 2.1 as published by the Free Software Foundation.
00011  *
00012  *  This library is distributed in the hope that it will be useful,
00013  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00014  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00015  *  Lesser General Public License for more details.
00016  *
00017  *  You should have received a copy of the GNU Lesser General Public
00018  *  License along with this library; if not, write to the Free Software
00019  *  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
00020  *  MA  02110-1301  USA
00021  */
00022
00023 /*
00024  * \file aes.h
00025  */
00026
00027 #ifndef _AES_H
00028 #define _AES_H
00029
00033 typedef struct
00034 {
00035     unsigned long erk[64];
00036     unsigned long drk[64];
00037     int nr;
00038 } aes_context;
00039
00047 void aes_set_key(aes_context *ctx, const unsigned char *key, int keysize);
00048
00056 void aes_encrypt(aes_context *ctx, unsigned char input[16], unsigned char output[16]);
00057
00065 void aes_decrypt(aes_context *ctx, unsigned char input[16], unsigned char output[16]);
00066
00076 void aes_cbc_encrypt(aes_context *ctx, unsigned char iv[16], unsigned char *input, unsigned char
      *output, int len);
```

```
00077
00087 void aes_cbc_decrypt(aes_context *ctx, unsigned char iv[16], unsigned char *input, unsigned char
      *output, int len);
00088
00094 int aes_self_test(int verbose);
00095
00096 #endif /* aes.h */
```

24.289 arp.h File Reference

ARP.

```
#include <buffers.h>
#include <nettypes.h>
#include <stdio.h>
```

Functions

- void ShowArp ()

  Display ARP cache, output will be the stdio serial port.
- void fShowArp (FILE *fp)

  Display ARP cache, output will be sent to the specified file pointer.
- BOOL GetArpMacFromIp (IPADDR4 ip, MACADR &ma)

  Check to see if the specified IP address is in the ARP cache.
- void sendGratuitousArp (int interfaceNumber, IPADDR4 ip)

  Send Gratuitous ARP Request.

24.289.1 Detailed Description

ARP.

24.290 arp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00021 #ifndef _NB_ARP_H
00022 #define _NB_ARP_H
00023
00024 #include <buffers.h>
00025 #include <nettypes.h>
00026 #include <stdio.h>
00027
00028 /*
00029  ********************************************************************************
00030  *
00031  * Runtime Libraries Routine Prototypes
00032  *
00033  ********************************************************************************
00034  */
00035
00041 void ShowArp();
00042
00050 void fShowArp(FILE *fp);
00051
00065 BOOL GetArpMacFromIp(IPADDR4 ip, MACADR &ma);
00066
00078 void sendGratuitousArp(int interfaceNumber, IPADDR4 ip);
00079
00080 /*INTERNAL FUNCTIONS */
00081 void processArp(PoolPtr p, PEFRAME pF);
00082
00083 void AddCheckArp(MACADR *ma, IPADDR4 ip, int ifnum);
00084 void DeleteArpFromIp(IPADDR4 ip);
00085
00086 void RouteOut(PEFRAME pf, IPADDR4 ip, PoolPtr p);
00087 void RouteOutVia(PEFRAME pf, IPADDR4 ipfrom, IPADDR4 ipto, PoolPtr p, int intf);
00088
```

```
00089 void AgeArp();
00090 void AddStaticArp(MACADR *ma, IPADDR4 ip, int ifnum);
00091
00092 void InitializeArp();
00093
00094 BOOL IsMyAddressUsedArpDetect(uint16_t timeout, int interface = 0);
00095
00096 int GetProperInterface4(IPADDR4 dst);
00097
00098 /*****************************************************************************
00099  *
00100  * Data Structures
00101  *
00102  *****************************************************************************/
00103
00104 /*****************************************************************************
00105
00106    Address Resolution Protocol (ARP) Packet Structure
00107
00108       hard_Type            - Hardware type (HTYPE)
00109       prot_Type            - Protocol type (PTYPE)
00110       hard_size            - Hardware length (HLEN)
00111
00112          theMac            - Interface ethernet MAC address
00113    Notes:
00114       Should not change data sizes and always add to the end to avoid breaking
00115       use for interface configuration.
00116
00117  *****************************************************************************/
00118 typedef struct
00119 {
00120     beuint16_t hard_Type;
00121     beuint16_t prot_Type;
00122     uint8_t hard_size;
00123     uint8_t prot_size;
00124     beuint16_t op_code;
00125     MACADR sender_phy;
00126     IPADDR4 sender_Ip;
00127     MACADR target_phy;
00128     IPADDR4 target_Ip;
00129 } __attribute__((packed)) ARP;
00130
00131 typedef ARP *PARP;
00132
00133 #endif
00134
```

24.291 arpinternal.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _ARP_INTERNAL_H
00006 #define _ARP_INTERNAL_H
00007
00008 #include <basictypes.h>
00009 #include <buffers.h>
00010 #include <nettypes.h>
00011
00012 /*-------------------------------------------------------------------------**
00013  ** The following structure defines the                                     **
00014  ** data records in the arp cache.                                         **
00015  ** The fields are:                                                        **
00016  **                                                                        **
00017  ** mac -> The hardware MAC address associated with this record.           **
00018  **              This value is NULL when the entry has not yet been        **
00019  **              processed.                                                **
00020  **                                                                        **
00021  ** ip  -> The IP address associated with this record.                     **
00022  **                                                                        **
00023  ** life_limit ->The seconds count when this entry should be deleted.      **
00024  **                                                                        **
00025  ** pPendingSend ->The buffer pool pointer, this buffer should be sent when the **
00026  **                          mac address is finally resolved.              **
00027  **-------------------------------------------------------------------------*/
00028 #define ARP_STATIC_LIFETIME (0xFFFFFFFF)
00029 typedef struct
00030 {
00031     MACADR mac;
00032     IPADDR4 ip;
00033     uint32_t life_limit;
00034     PoolPtr pPendingSend;
00035     uint16_t pvtData;
00036     uint16_t interface;
```

```
00037     OS_SEM *pPendingSem;
00038     inline uint32_t SetLife(uint32_t newLife)
00039     {
00040         if (life_limit != ARP_STATIC_LIFETIME) { life_limit = newLife; }
00041         return life_limit;
00042     }
00043 } ArpRecord;
00044
00045 BOOL IsNull(ArpRecord *ar);
00046 ArpRecord *FindArp(IPADDR4 ip);
00047 ArpRecord *FindAddArp(IPADDR4 ip);
00048 BOOL MacEqual(MACADR a1, MACADR a2);
00049 void RawSendArp(ArpRecord *pa, IPADDR4 fromaddr, int ifout);
00050
00051 #endif /* ----- #ifndef _ARP_INTERNAL_H_INC  ----- */
```

24.292 atcommand.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef _AT_COMMAND_H
00005 #define _AT_COMMAND_H
00006 #include <stdio.h>
00007
00008 #define AT_COMMAND_REASON_START_AT (1)
00009 #define AT_COMMAND_REASON_EXIT_AT (2)
00010 #define AT_COMMAND_REASON_NORMAL_COMMAND (3)
00011
00012 #define AT_COMMAND_OK (1)
00013 #define AT_COMMAND_EXIT (2)
00014
00015 /* type def for the AT command processing function */
00016 /* Needs to return one of :
00017
00018 AT_COMMAND_OK
00019 AT_COMMAND_EXIT
00020 */
00021 typedef int(ProcessATcommandFunc)(int uartnum, const char *pCommand, FILE *pResponseFile, int reason);
00022
00023 /* Enable AT command on the specific serial port, port must already be popend in interrupt mode. */
00024 void EnableATCommands(int port_number, ProcessATcommandFunc *pcmdf, int task_priority);
00025
00026 void DisableATCommands(int port_number);
00027
00028 #endif
```

24.293 autoip.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  * Doxygen note: AutoIP is an option set in the configuration system for each interface, so no
00007  * functions are used by the application and they do not need to be documented. The previous
00008  * documentation is copied below for historical reference.
00009  */
00010
00011 #ifndef _AUTOIP_H_INC
00012 #define _AUTOIP_H_INC
00013
00014 #include <buffers.h>
00015 #include <nettimer.h>
00016
00017 #define PROBE_WAIT 1
00018 #define PROBE_NUM 3
00019 #define PROBE_MIN 1
00020 #define PROBE_MAX 2
00021 #define ANNOUNCE_WAIT 2
00022 #define ANNOUNCE_NUM 2
00023 #define ANNOUNCE_INTERVAL 2
00024 #define MAX_CONFLICTS 10
00025 #define RATE_LIMIT_INTERVAL 60
00026 #define DEFEND_INTERVAL 10
00027
00028 #define BASE_AUTO_IP 0xA9FE0100
00029
00030 #define AUTO_IP_INIT 1
00031 #define AUTO_IP_NEGOTIATING 2
00032 #define AUTO_IP_CONFIGURED 3
00033 #define AUTO_IP_STOPPED 4
```

```
00034 #define AUTO_IP_STARTARP 5
00035 #define AUTO_IP_ARPCHECKING 6
00036 #define AUTO_IP_ANNOUNCING 7
00037 #define AUTO_IP_CONFLICT 8
00038 #define AUTO_IP_DISABLED 9
00039
00040 #define NOSCHEDULE_TICK 0
00041
00042 class InterfaceBlock;
00043
00044 class AutoIPClient : public TimeOutElement
00045 {
00046    private:
00047     InterfaceBlock *pIfb;
00048     int AutoIPState;
00049     IPADDR4 last_try;
00050
00051     unsigned long nextTick;
00052     int probeCount;
00053     int announceCount;
00054     int conflicts;
00055     bool collision;
00056
00057     AutoIPClient *nextIPC;
00058
00059     void addressCollision();
00060     IPADDR4 generateNewIPAddr();
00061     void negotiate();
00062     void startArp();
00063     void arpCheck();
00064     void announce();
00065     void tickDelay(uint16_t delay);
00066
00067     virtual void TimeElementEvent();
00068
00069    public:
00070     AutoIPClient(InterfaceBlock *ib);
00071     ~AutoIPClient();
00072     void start();
00073     void stop();
00074     void disable();
00075     void enable();
00076     void restart();
00077     int getState();
00078     friend void processArp(PoolPtr p, PEFRAME pF);
00079     IPADDR4 cur_val();
00080     void LinkNotify(bool bLink);
00081 };
00082
00083 AutoIPClient *AddAutoIPInterface(int InterfaceNumber = 0);
00084
00085 #endif /* ----- #ifndef _AUTOIP_H_INC  ----- */
```

24.294 base64.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _BASE64_H_
00006 #define _BASE64_H_
00007
00008 #include <basictypes.h>
00009 #include <ctype.h>
00010 #include <iointernal.h>
00011
00028 int Base64Decode(const char *input, uint8_t *outputbuf, const char *pEndMarker = NULL);
00029
00046 int Base64UrlDecode(const char *input, uint8_t *outputbuf, const char *pEndMarker = NULL);
00047
00064 int Base64Encode(uint8_t *input, int inlen, char *outputbuf);
00065
00082 int Base64UrlEncode(uint8_t *input, int inlen, char *outputbuf);
00083
00101 int Base64StreamEncode(int fd, uint8_t *input, int inlen, char *outBuf, int maxOutLen);
00102
00120 int Base64UrlStreamEncode(int fd, uint8_t *input, int inlen, char *outBuf, int maxOutLen);
00121
00122 class b64ctx
00123 {
00124    private:
00125     int bits;
00126     uint32_t accum;
00127     char *outBuf;
00128     const char *encodestr;
```

```
00129     int maxOutLen;
00130     int m_myFd;
00131     int m_baseFd;
00132
00133     int write(const char *in, int inlen);
00134     int close();
00135     friend int b64io_write(int fd, const char *in, int nbytes);
00136     friend int b64io_close(int fd);
00137
00138   public:
00139     b64ctx(char *_outBuf, int _maxOutLen, bool UrlEncode = false);
00140     void init(char *_outBuf, int _maxOutLen);
00141     int flush();
00142
00143     int GetFD(IoExpandStruct &io, int baseFd);
00144 };
00145
00146 #ifdef TEST_BASE64
00147 /* Test string */
00148 const char *base64msg =
00149     "TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFz\
00150 b24sIGJ1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCB\
00151 pcyBhIGx1c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbi\
00152 B0aGUgY29udGludWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBle\
00153 GNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=";
00154 #endif /* #ifdef TEST_BASE64 */
00155
00156 #endif /* #ifndef _BASE64_H_ */
```

24.295 bb_i2c.h

```
00001 #ifndef  __BB_I2C_H
00002 #define  __BB_I2C_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 #include <predef.h>
00008 #include <nbrtos.h>
00009 #include <pins.h>
00010
00011 /*
00012  * I2C WireIntf objects instantiated by the system.
00013  *
00014  * Provides access to the all the I2C peripherals on the processor. The I2C peripherals available will
      vary by platform.
00015  * For example, Wire.read(); will exectute a read opeeration on I2C module 0.
00016  */
00017
00018 #ifndef WIRE_DEFAULT_IINSTACE
00019 #define WIRE_DEFAULT_INSTANCE Wire0
00020 #endif
00021 #define Wire WIRE_DEFAULT_INSTANCE
00022
00023
00024 /*-------------------------------------------------------------------------------------------------
00025  * I2C Class
00026  *-------------------------------------------------------------------------------------------------*/
00027
00039 class BBI2C {
00040 public:
00041     enum Result_t {
00042         I2C_RES_ACK,
00043         I2C_RES_NACK,
00044         I2C_RES_ARB_LST,
00045         I2C_RES_BUSY,
00046         I2C_RES_ARG,
00047         I2C_RES_IN_PROGRESS,
00048         I2C_RES_ERR
00049     };
00050
00051 private:
00052     enum HwCmd_t {
00053         CMD_TXD          = 0,
00054         CMD_RXN          = 1,
00055         CMD_STOP         = 2,
00056         CMD_DISCARDN     = 3,
00057         CMD_START        = 4,
00058         CMD_START_NAK    = 5,
00059         CMD_START_HS     = 6,
00060         CMD_START_HS_NAK = 7,
00061     };
00062     enum TxnStat {
00063         TXN_RDY,
00064         TXN_SETUP,
```

```
00065         TXN_BUS_BUSY,
00066         TXN_RESTART_FOR_READ,
00067         TXN_BEGIN_RX,
00068         TXN_RX_IN_PROGRESS,
00069         TXN_RX_QUEUED,
00070         TXN_RX_STOPPING,
00071         TXN_BEGIN_TX,
00072         TXN_TX_IN_PROGRESS,
00073         TXN_TX_QUEUED,
00074         TXN_IN_PROGRESS,
00075         TXN_STOPPING,
00076         TXN_WAITING
00077     };
00078 #ifdef I2C_DEBUG
00079     struct IsrLogEntry {
00080         uint32_t isrNum;
00081         TxnStat oldState;
00082         TxnStat newState;
00083         const uint8_t *buf;
00084         uint32_t len;
00085         uint32_t lenToQueue;
00086         uint32_t tick;
00087         uint32_t tickFrac;
00088         int line;
00089 
00090         void Log(const I2C &i2c, TxnStat state, int line);
00091         void Dump();
00092     };
00093     friend struct IsrLogEntry;
00094     struct IsrLog {
00095         IsrLogEntry entry[ISR_LOG_LEN];
00096         uint32_t head;
00097         uint32_t tail;
00098         IsrLog();
00099         void Log(const I2C &i2c, TxnStat state, int line);
00100         void Dump();
00101     };
00102     friend struct IsrLog;
00103 #endif
00104     PinIO scl, sda;
00105 
00106     uint32_t delayCount;
00107     uint8_t  *dataBuf;
00108     uint32_t dataLen;
00109     uint32_t rxLenToQueue;
00110     uint16_t maxDelay_ms;
00111     uint16_t delayCycles_ms;// how many delay counts per milisecond
00112     uint16_t currPinLowCount;// how many milisecond counts we've been low for
00113     uint16_t reqPinLowCount;// how many miliseconds counts we need
00114     TxnStat  txnState;
00115     Result_t txnResult;
00116 
00117     OS_SEM txnSem;
00118 
00119     uint32_t isrCnt;
00120 #ifdef I2C_DEBUG
00121     IsrLog   log;
00122 #endif
00123 
00124 
00125     uint8_t  iadrAddressSize;   // Number of address register bytes, 0 - 3
00126     void isr();
00127     int isr_rx(uint32_t sr);
00128     int isr_tx(uint32_t sr);
00129 #ifdef I2C_DEBUG
00130     void Log(TxnStat state, int line);
00131     void DumpLog();
00132 #endif
00133 
00134     Result_t start(uint8_t deviceAddr);
00135     Result_t restart(uint8_t deviceAddr);
00136 
00137     Result_t stop();
00138     Result_t write8(uint8_t dat);
00139     Result_t read8(uint8_t &dat, bool bLast);
00140     Result_t queueRx();
00141 
00142 
00143 public:
00150     BBI2C(PinIO scl, PinIO sda);
00151 
00152     // Copy constructor
00153     BBI2C(const BBI2C & rhs)
00154         : scl(rhs.scl), sda(rhs.sda), txnState(rhs.txnState)
00155     {}
00156 
00163     void setup(uint32_t maxBusSpeed);
```

```
00164
00169     void resetBus();
00170
00185     inline void setNumAddressBytes(uint8_t numAddressBytes = 1) { iadrAddressSize = numAddressBytes; }
00186
00196     virtual Result_t writeReg8(uint8_t devAddr, uint32_t reg, uint8_t data);
00197     virtual Result_t writeReg16(uint8_t devAddr, uint32_t reg, uint16_t data, int byteOrder);
00198     virtual Result_t writeReg32(uint8_t devAddr, uint32_t reg, uint32_t data, int byteOrder);
00199
00209     virtual Result_t readReg8(uint8_t devAddr, uint32_t reg, uint8_t &data);
00210     virtual Result_t readReg16(uint8_t devAddr, uint32_t reg, uint16_t &data, int byteOrder);
00211     virtual Result_t readReg32(uint8_t devAddr, uint32_t reg, uint32_t &data, int byteOrder);
00212
00227     virtual Result_t writeRegN(uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen);
00228
00239     virtual Result_t readRegN(uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen);
00240
00241     // Static implementations of the C++ class that can be used if you prefer a C style interface
00242     static Result_t writeReg8(int module, uint8_t devAddr, uint32_t reg, uint8_t dat);
00243     static Result_t readReg8(int module, uint8_t devAddr, uint32_t reg, uint8_t &dat);
00244     static Result_t writeRegN(int module, uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen);
00245     static Result_t readRegN(int module, uint8_t devAddr, uint32_t reg, uint8_t *buf, uint32_t blen);
00246
00247     // These functions are for internal use only.
00248     void dump(uint32_t line);
00255     void SetMaxBusDelay(uint16_t miliseconds);
00256     uint32_t GetBusSpeed() { return 132000000/delayCount; }
00257
00258     friend void LPI2C1_ISR();
00259     friend void LPI2C2_ISR();
00260     friend void LPI2C3_ISR();
00261     friend void LPI2C4_ISR();
00262
00263     friend class WireIntf;
00264
00265 };  // end class I2C
00266
00267 #endif   /* ----- #ifndef __BB_I2C_H  ----- */
```

24.296 buffers.h File Reference

NetBurner Buffers API.

```
#include <constants.h>
#include <basictypes.h>
#include <nbrtos.h>
```

Functions

- uint16_t GetFreeCount ()

  Returns the number of free buffers in the system. Buffers are used for both serial and network interfaces.

- void ShowBuffer (PoolPtr p)

  Prints a pool buffer to stdout.

24.296.1 Detailed Description

NetBurner Buffers API.

24.297 buffers.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00017 /* This file provides the definitions for handling Buffers */
00018
00019 #include <constants.h>   // Added 3-22-12, dciliske
00020
00021 #ifndef _NB_BUFFERS_H
00022 #define _NB_BUFFERS_H
00023
00024 #ifdef _DEBUG
```

```
00025 #define BUFFER_DIAG
00026 #endif
00027
00028 #include <basictypes.h>
00029 #include <nbrtos.h>
00030
00031 struct pool_buffer; /*Forward declaration*/
00032 typedef struct pool_buffer *PoolPtr;
00033 typedef volatile PoolPtr VPoolPtr;
00034
00035 /*The buffer states that define the ownership of the buffer */
00036 #define BO_UNUSED 0   /*The buffer is free*/
00037 #define BO_SOFTWARE 1 /*The buffer belongs to some software function.*/
00038 #define BO_PRX 2      /*The buffer is to be used for the RX function.*/
00039 #define BO_PTXF 3     /*The buffer is to be TXed and then freeded*/
00040 #define BO_PTXS 4     /*The buffer is to be TXed and then set to S/W owns.*/
00041
00042 /*The buffer state flags */
00043 #define BS_PHY_BCAST 1    /*The underlying packet is to/from a physical layer broadcast address.*/
00044 #define BS_IP_BCAST 2     /*The underlying packet is to/from a layer 3 broadcast address.*/
00045 #define BS_IP_LOCAL_NET 4 /*The underlying packet is to/from the local subnet.*/
00046 #define BS_TCP_PUSH 8     /*The push flag was set on RX.*/
00047 #define BS_PHY_802_3 0x10 /*The network wants 802_3 encapsulation*/
00048 #define BS_VLAN 0x20      /*The network wants 802_3 encapsulation*/
00049 #define BS_DBG_LOG 0x80   /*This is a debug log buffer. Do *not* log operations on it.*/
00050
00051 #ifdef BUFFER_DIAG
00052 #define BUFFER_DIAG_WORD_COUNT 4
00053 #else
00054 #define BUFFER_DIAG_WORD_COUNT 0
00055 #endif
00056 #ifdef NET_SHIFT16
00057 #define BUF_PREDATA_SENTINAL_LEN (8)
00058 #define BUF_PREDATA_PADLEN 8
00059 #define BUF_POSTDATA_PADLEN 3
00060 #define BUF_POSTDATA_SENTINAL_LEN (11 - BUFFER_DIAG_WORD_COUNT)
00061 #else
00062 #define BUF_PREDATA_SENTINAL_LEN (8)
00063 #define BUF_PREDATA_PADLEN 6
00064 #define BUF_POSTDATA_PADLEN 3
00065 #define BUF_POSTDATA_SENTINAL_LEN (12 - BUFFER_DIAG_WORD_COUNT)
00066 #endif
00067
00068 struct pool_buffer : public OS_FIFO_EL
00069 {
00070     //      puint8_t pBufQueuePointer;          /* For the OS FIFO list stuff */
00071     uint32_t PreSentinal[BUF_PREDATA_SENTINAL_LEN];
00072
00073     VPoolPtr pPoolNext; /*A pointer to the next buffer in pool*/
00074     VPoolPtr pPoolPrev; /*A pointer to the prev buffer in pool*/
00075     vuint32_t dwTime;
00076     vuint32_t dwTimeFraction;
00077     vuint16_t usedsize;
00078     uint8_t bBuffer_state; /*Unused,PRx,PtxF,PtxS,RxC*/
00079     uint8_t bBufferFlag;
00080     uint8_t bUsageCount;
00081     uint8_t bInterfaceNumber;
00082     uint8_t bAlignmentPad[BUF_PREDATA_PADLEN];
00083     uint8_t pData[ETHER_BUFFER_SIZE]; /*The buffer data*/
00084     uint8_t bPostAlignmentPad[BUF_POSTDATA_PADLEN];
00085     uint32_t PostSentinal[BUF_POSTDATA_SENTINAL_LEN];
00086
00087 #ifdef BUFFER_DIAG
00088     PCSTR m_fAlloc;
00089     uint32_t m_fline;
00090     uint32_t m_fill;
00091     uint32_t m_fill2;
00092 #endif /* BUFFER_DIAG */
00093 };
00094
00095 inline uint8_t OSPoolFifoPost(OS_FIFO *pFifo, PoolPtr pToPost)
00096 {
00097     return pFifo->Post(pToPost);
00098 }
00099
00100 inline uint8_t OSPoolFifoPostFirst(OS_FIFO *pFifo, PoolPtr pToPost)
00101 {
00102     return pFifo->PostFirst(pToPost);
00103 }
00104
00105 inline PoolPtr OSPoolFifoPend(OS_FIFO *pFifo, uint16_t timeout)
00106 {
00107     return static_cast<PoolPtr>(pFifo->Pend(timeout));
00108 }
00109
00110 inline PoolPtr OSPoolFifoPend(OS_FIFO *pFifo, TickTimeout timeout)
00111 {
```

```
00112     return static_cast<PoolPtr>(pFifo->Pend(timeout));
00113 }
00114
00115
00116 inline PoolPtr OSPoolFifoPendNoWait(OS_FIFO *pFifo)
00117 {
00118     return static_cast<PoolPtr>(pFifo->PendNoWait());
00119 }
00120
00121 /*Buffer operations*/
00122
00123 #ifdef BUFFER_DIAG
00124 void ShowBuffers();
00125 void ShowBuffers_Web(int sockfd);
00126 PoolPtr GetBufferX(PCSTR file, int line); /*Alocates a Software buffer and returns the buffer
      number.*/
00127 #define GetBuffer() GetBufferX(__FILE__, __LINE__)
00128 #ifdef FAST_BUFFERS
00129 PoolPtr GetFastBufferX(PCSTR file, int line); /*Alocates a Software buffer and returns the buffer
      number.*/
00130 #define GetFastBuffer() GetFastBufferX(__FILE__, __LINE__)
00131 #else /* #ifdef FAST_BUFFERS */
00132 #define GetFastBuffer() GetBufferX(__FILE__, __LINE__)
00133 #endif                                          /* #ifdef FAST_BUFFERS */
00134 void FreeBufferX(PoolPtr nbuf, PCSTR file, int line); /*Frees a buffer and places it in the unused
      state.*/
00135 #define FreeBuffer(x) FreeBufferX(x, __FILE__, __LINE__)
00136
00137 void ChangeOwnerX(PoolPtr nbuf, PCSTR file, int line); /*Frees a buffer and places it in the unused
      state.*/
00138 #define ChangeOwner(x) ChangeOwnerX(x, __FILE__, __LINE__)
00139 #else
00140 PoolPtr GetBuffer();           /*Alocates a Software buffer and returns the buffer number.*/
00141 #ifdef FAST_BUFFERS
00142 PoolPtr GetFastBuffer();       /*Alocates a Software buffer (potentially from the fast pool) pool and
      returns the buffer number.*/
00143 #else /* #ifdef FAST_BUFFERS */
00144 #define GetFastBuffer() GetBuffer()
00145 #endif /* #ifdef FAST_BUFFERS */
00146 void FreeBuffer(PoolPtr nbuf); /*Frees a buffer and places it in the unused state.*/
00147 #define ChangeOwner(x)
00148 #endif
00149
00150 #ifdef BUFFER_DIAG_LOG
00151 #include <syslog.h>
00152 extern uint32_t bufLogID;
00153 #define LOG_DEST AsciiToIp("10.1.1.3")
00154 #define SET_LOG_DEST                    \
00155     {                                   \
00156         extern IPADDR SysLogAddress; \
00157         SysLogAddress = LOG_DEST;       \
00158     }
00159 #define LOG_INTF 2
00160 #define BuffLog_GetFmt "%0.8lu)  GET    - P: %p, L: %0.4d, F: %s\r\n"
00161 #define BuffLog_FreeFmt "%0.8lu)  FREE   - P: %p, L: %0.4d, F: %s\r\n"
00162 #define BuffLog_GetLF(p, l, f) SysLogVia(LOG_INTF, BuffLog_GetFmt, bufLogID++, p, l, f)
00163 #define BuffLog_FreeLF(p, l, f) SysLogVia(LOG_INTF, BuffLog_FreeFmt, bufLogID++, p, l, f)
00164 #define BuffLog_Get(p) BuffLog_GetLF(p, __LINE__, __FILE__)
00165 #define BuffLog_Free(p) BuffLog_FreeLF(p, __LINE__, __FILE__)
00166 #define BuffLog_Write(p) SysLogVia(LOG_INTF, "WRITE  - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
      __FILE__)
00167 #define BuffLog_Read(p) SysLogVia(LOG_INTF, "READ   - P: %p, L: %0.4d, F: %s\r\n", p, __LINE__,
      __FILE__)
00168 #else
00169 #define BuffLog_Get(p)
00170 #define BuffLog_Free(p)
00171 #define BuffLog_Write(p)
00172 #define BuffLog_Read(p)
00173 #endif
00174 /*Frees a linked list of buffers and places them in the unused state.*/
00175 void FreeBufferList(PoolPtr nbuf);
00176
00177 void IncUsageCount(PoolPtr pp);
00178
00203 uint16_t GetFreeCount();
00204
00205 void InitBuffers(); /* Initializes the buffer system*/
00206
00211 void ShowBuffer(PoolPtr p);
00212
00213 class buffer_list
00214 {
00215    public:
00216     PoolPtr m_Head;
00217     PoolPtr m_Tail;
00218     uint16_t m_wElements;
00219     void InsertHead(PoolPtr buffer);
```

```
00220     void InsertTail(PoolPtr buffer);
00221     void InsertBefore(PoolPtr buf2insert, PoolPtr b4buffer);
00222     void InsertAfter(PoolPtr buf2insert, PoolPtr after_buffer);
00223     void Remove(PoolPtr buffer);
00224     PoolPtr RemoveHead();
00225     PoolPtr RemoveTail();
00226     buffer_list()
00227     {
00228         m_Head = 0;
00229         m_Tail = 0;
00230         m_wElements = 0;
00231     }
00232     uint16_t GetCount() { return m_wElements; };
00233 };
00234
00235 /*
00236 This defines a class for storing an array of bytes in a list of buffers.
00237 It is used for I/O buffers
00238 */
00239 class fifo_buffer_storage
00240 {
00241    private:
00242     PoolPtr m_Head;
00243     PoolPtr m_Tail;
00244     OS_CRIT m_critical_section;
00245     int m_BytesStored;
00246     uint32_t m_startOffset;
00247     uint32_t m_maxChunkLen;
00248     uint8_t m_Segments_Stored;
00249     uint8_t m_MaxSegments; /*Stores the max number of segments allowed fore this buffer*/
00250     bool m_checksumWrites;
00251     uint8_t m_buffer_pool;
00252
00253     // storage access/update functions to be used by BufPtr.
00254     // DO NOT CALL WITHOUT LOCKING STRUCT FIRST
00255     uint8_t *GetWritePtr(uint32_t *remLen);
00256     void WriteDone(uint32_t bytesCopied);
00257
00258     uint8_t *GetReadPtr(uint32_t *remLen);
00259     void ReadDone(uint32_t bytesCopied);
00260
00261    public:
00262     /* Accessor for internal struct buffer pointers. NOT IRQ SAFE */
00263     class WriteBufPtr
00264     {
00265         WriteBufPtr();
00266         fifo_buffer_storage &m_fifo;
00267         uint8_t *pBuf;
00268         uint32_t bufLen;
00269         uint32_t copied;
00270
00271        public:
00272         WriteBufPtr(fifo_buffer_storage &fifo) : m_fifo(fifo), copied(0)
00273         {
00274             m_fifo.m_critical_section.Enter();
00275             pBuf = m_fifo.GetWritePtr(&bufLen);
00276         }
00277         ~WriteBufPtr()
00278         {
00279             m_fifo.WriteDone(copied);
00280             m_fifo.m_critical_section.Leave();
00281         }
00282
00283         inline uint8_t *buf() { return pBuf + copied; }
00284         inline uint32_t len() { return bufLen - copied; }
00285         void ByteCopyDone(uint32_t bytesCopied) { copied += (bytesCopied <= (bufLen - copied)) ?
      bytesCopied : (bufLen - copied); }
00286     };
00287     /* Accessor for internal struct buffer pointers. NOT IRQ SAFE */
00288     class ReadBufPtr
00289     {
00290         ReadBufPtr();
00291         fifo_buffer_storage &m_fifo;
00292         uint8_t *pBuf;
00293         uint32_t bufLen;
00294         uint32_t copied;
00295
00296        public:
00297         ReadBufPtr(fifo_buffer_storage &fifo) : m_fifo(fifo), copied(0)
00298         {
00299             m_fifo.m_critical_section.Enter();
00300             pBuf = m_fifo.GetReadPtr(&bufLen);
00301         }
00302         ~ReadBufPtr()
00303         {
00304             m_fifo.ReadDone(copied);
00305             m_fifo.m_critical_section.Leave();
```

```
00306         }
00307
00308         inline uint8_t *buf() { return pBuf + copied; }
00309         inline uint32_t len() { return bufLen - copied; }
00310         void ByteCopyDone(uint32_t bytesCopied) { copied += (bytesCopied <= (bufLen - copied)) ?
      bytesCopied : (bufLen - copied); }
00311     };
00312     class PeekIterator
00313     {
00314         PeekIterator();
00315         fifo_buffer_storage &m_fifo;
00316         PoolPtr pCurrPP;
00317         uint8_t *peekPtr;
00318         bool locked;
00319
00320         PeekIterator(const PeekIterator &rhs)
00321             : m_fifo(rhs.m_fifo), pCurrPP(rhs.pCurrPP), peekPtr(rhs.peekPtr), locked(false)
00322         {
00323             if (pCurrPP)
00324             {
00325                 m_fifo.m_critical_section.Enter();
00326                 locked = true;
00327             }
00328         }
00329
00330        public:
00331         PeekIterator(fifo_buffer_storage &fifo, bool begin = true) : m_fifo(fifo), pCurrPP(NULL),
      peekPtr(NULL), locked(false)
00332         {
00333             if (begin)
00334             {
00335                 m_fifo.m_critical_section.Enter();
00336                 locked = true;
00337                 pCurrPP = m_fifo.m_Head;
00338                 if (pCurrPP) { peekPtr = pCurrPP->pAsBytePtr; }
00339             }
00340         }
00341         ~PeekIterator() { if (locked) m_fifo.m_critical_section.Leave(); }
00342
00343         PeekIterator &operator++();     // preincrement
00344         PeekIterator operator++(int);   // postincrement
00345         uint8_t operator*();            // dereference
00346         int32_t operator-(const PeekIterator &rhs);
00347         PeekIterator &operator+=(int);
00348
00349         operator uint16_t() const;
00350         operator uint32_t() const;
00351
00352         inline bool operator==(const PeekIterator &rhs)
00353         {
00354             return ((&m_fifo == &(rhs.m_fifo)) && (pCurrPP == rhs.pCurrPP) && (peekPtr ==
      rhs.peekPtr));
00355         }
00356         bool operator!=(const PeekIterator &rhs) { return !(*this == rhs); }
00357     };
00358
00359     friend class WriteBufPtr;
00360     friend class ReadBufPtr;
00361     friend class PeekIterator;
00362
00363     uint32_t LongSpaceAvail();
00364
00365     /*The number of available bytes of storage*/
00366     uint16_t SpaceAvail();
00367     /*The number of bytes used.*/
00368     uint16_t SpaceUsed();
00369
00370     /*Read data from the buffer*/
00371     /*REturns the number of bytes read*/
00372     int ReadData(puint8_t pCopyTo, int max_bytes);
00373
00374     /* Peek at the next data byte */
00375     uint8_t PeekData(int idx = -1);
00376
00377     int ReadDatawSum(puint8_t pCopyTo, int max_bytes, uint32_t &csum);
00378
00379     // Returns number of bytes skipped
00380     int SkipData(int skip);
00381
00382     int ReadTerminatedData(puint8_t pCopyTo, int max_bytes, uint8_t term_char);
00383
00384     /* Advanced functions for no-copy buffer usage */
00385     int PushBuffer(PoolPtr pp, int dataLen, int startOffset);
00386     PoolPtr PullBuffer();
00387     PoolPtr PullBuffer(uint32_t &csum);
00388     inline PoolPtr PeekBuffer() { return m_Head; }
00389     void SetFifoLock(bool locked);
```

```
00390
00391     /*Write Data to the buffer*/
00392     /*Returns the number of bytes read*/
00393     int WriteData(puint8_t pCopyFrom, int num_bytes);
00394
00395     void RemoveLast();   // Remove last  char
00396
00397     /*Peek Iterator methods for looking at queued data*/
00398     PeekIterator Peek_Begin();
00399     PeekIterator Peek_End();
00400
00401     /*Returns true if empty*/
00402     BOOL Empty();
00403
00404     /*Returns True if Full*/
00405     BOOL Full();
00406
00407     /*Build one */
00408     fifo_buffer_storage(uint8_t max_buffers = 0, uint8_t use_fromisr = 1, uint8_t use_fast_buffs = 1);
00409     ~fifo_buffer_storage();
00410
00411     /*Reset frees the storage*/
00412     void Reset(uint8_t max_buffers);
00413
00414     void SetMaxBuffers(uint8_t max_buffers);
00415     inline void SetMaxChunkLen(uint32_t maxStorage)
00416     {
00417         m_maxChunkLen = (maxStorage > (ETHER_BUFFER_SIZE - m_startOffset)) ? (ETHER_BUFFER_SIZE -
      m_startOffset) : maxStorage;
00418     }
00419     inline void SetStartOffset(uint32_t offset)
00420     {
00421         m_startOffset = offset;
00422         SetMaxChunkLen(m_maxChunkLen);
00423     }
00424     inline uint32_t GetMaxBuffers()  { return m_MaxSegments; }
00425     inline uint32_t GetMaxChunkLen() { return m_maxChunkLen; }
00426     inline uint32_t GetStartOffset() { return m_startOffset; }
00427     inline void SetWriteChecksum(bool summingOn) { m_checksumWrites = summingOn; }
00428     inline uint32_t GetMinimumFullSize(int maxSegments = -1)
00429         {
00430             if (maxSegments < 0)
00431                 maxSegments = m_MaxSegments;
00432             return (m_maxChunkLen * (maxSegments-1))+1;
00433         }
00434
00435     /*Init initializes it to zero*/
00436     void Init(uint8_t max_buffers, uint8_t use_fromisr = 1, uint8_t use_fast_buffs = 1);
00437 } __attribute__((packed));
00438
00439 class SMPoolPtr
00440 {
00441     PoolPtr m_pp;
00442
00443   public:
00444     SMPoolPtr()
00445     {
00446         m_pp = GetBuffer();
00447         BuffLog_Get(m_pp);
00448     }
00449     SMPoolPtr(const PoolPtr pp) { m_pp = pp; }
00450     SMPoolPtr(const SMPoolPtr &pp) { m_pp = pp.m_pp; }
00451     ~SMPoolPtr()
00452     {
00453         if (m_pp)
00454         {
00455             FreeBuffer(m_pp);
00456             BuffLog_Free(m_pp);
00457         }
00458     }
00459
00460     inline const SMPoolPtr &operator=(const PoolPtr pp)
00461     {
00462         m_pp = pp;
00463         return *this;
00464     }
00465     inline const SMPoolPtr &operator=(const SMPoolPtr &pp)
00466     {
00467         m_pp = pp.m_pp;
00468         return *this;
00469     }
00470     inline pool_buffer &operator*() { return *m_pp; }
00471     inline PoolPtr &operator->() { return m_pp; }
00472     explicit operator PoolPtr() { return m_pp; }
00473 };
00474
00475 #endif
```

```
00476
```

24.298 cc_attrs.h

```
00001 #ifndef __CC_ATTRS_H
00002 #define __CC_ATTRS_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <predef.h>
00007
00008 #undef __noinline
00009 #undef __section
00010
00011 #define __packed __attribute__((__packed__))
00012 #define __align(x) __attribute__((aligned(x)))
00013 #define __noinline __attribute__((noinline))
00014 #define __section(x) __attribute__((section(x)))
00015
00016
00017 #endif /* ----- #ifndef __CC_ATTRS_H  ----- */
```

24.299 command.h File Reference

NetBurner Command Processor.

```
#include <basictypes.h>
#include <stdio.h>
#include <ssh/NetBurner/UserAuthManager.h>
```

Macros

- #define **CMD_OK** (0)

  Function was excecuted properly.
- #define **CMD_CLOSE** (1)

  Connection closed.
- #define **CMD_FAIL** (2)

  General function failed.
- #define **CMD_AUTH_FAIL** (3)

  Authorization failed.
- #define **CMD_TO_MANY_FDS** (4)

  Too many file descriptors currently in use.
- #define **CMD_FAILED_SSH_KEY_GEN** (5)

  Unable to correctly generate key needed for SSH connections.
- #define **CMD_DIS_CAUSE_TIMEOUT** (1)

  Connection timed out.
- #define **CMD_DIS_CAUSE_CLOSED** (2)

  Connection closed.
- #define **CMD_DIS_SOCKET_CLOSED** (3)

  Socket closed. Don't send a response for this case.
- #define **CMD_DIS_AUTH_FAILED** (4)

  Authorization failed. Don't send a response for this case.

Enumerations

- enum ListenOn : int8_t { eListenOnTcp = 1 , eListenOnSsh = 2 }

  The various remote channels that the command processor can listen to.

Functions

- int CmdStartCommandProcessor (int priority)

  Start the command processor.
- int CmdAddCommandFd (int fd, bool require_auth, bool time_out_conn, bool local_echo=true)

  Add an established FD connection to the list of fd's managed by the command processor.
- int CmdRemoveCommandFd (int fd)

  Remove an established FD (either a TCP session, a serial connection, or an SSH session).
- int CmdListenOnTcpPort (uint16_t port, int do_telnet_processing, int max_connections)

  Start listening for a connection over TCP.
- int CmdListenQuietOnTcpPort (uint16_t port, int do_telnet_processing, int max_connections)

  Start listening for a connection over TCP, but without the siggnon or password.
- int CmdListenOnSshPort (uint16_t port, int max_connections)

  Start listening for a connection over SSH.
- int CmdListenQuietOnSshPort (uint16_t port, int max_connections)

  Start listening for a connection over SSH, but without the siggnon or password.
- int CmdStopListeningOnTcpPort (uint16_t port)

  Stop Listening for connections on the specified port. Also closes all open connections that were based on that port.
- int CmdStopListeningOnSshPort (uint16_t port)

  Stop Listening for connections on the specified port. Also closes all open connections that were based on that port.
- void SendToAll (const char *buffer, int len, bool include_serial_ports)

  Send a message to all connected sockets, excluding "Listening sockets".

Variables

- int(* CmdAuthenticateFunc)(const char *name, const char *passwd)

  External Authentication function CALLBACK for TCP connections, used to verify username and password. If this function pointer is not NULL then each new Telnet session will be asked to authenticate.
- int(* CmdAuthenticateSshFunc)(const char *name, const char *authVal, AuthType authType)

  External Authentication function CALLBACK for SSH connections, used to verify username and password. If this function pointer is not NULL then each new SSH session will be asked to authenticate. If this function pointer is NULL, it tries to use CmdAuthenticateFunc instead.
- int(* CmdCmd_func)(const char *command, FILE *fRespondto, void *pData)

  The command processing callback function for handling string commands.
- int(* CmdChar_func)(char command, FILE *fRespondto, void *pData)

  The command processing callback function for handling single character commands. If this is implemented does not do echo or line editing this is the responsibility of the application programmer.
- void *(* CmdConnect_func)(FILE *fRespondto)

  Connect callback function. If this function is not NULL, then the system will call this function every time a new session is started.
- void(* CmdPrompt_func)(FILE *fRespondto, void *pData)

  Prompt callback function. If this function is not NULL, then the system will call this function every time a new prompt line needs to be displayed.
- void(* CmdDisConnect_func)(FILE *fRespondto, int cause, void *pData)

  Dis-Connect callback function, if this function is not NULL then the system will call this function every time a session is terminated.
- int CmdIdleTimeout
- const char * **Cmdlogin_prompt**

  If this is defined, then it will be sent to the socket on connection before Authentication is tried.

24.299.1 Detailed Description

NetBurner Command Processor.

24.300 command.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00017 #ifndef _NB_COMMAND_H
00018 #define _NB_COMMAND_H
00019
00020 #include <basictypes.h>
00021 #include <stdio.h>
00022
00023 #if defined(NB_SSH_SUPPORTED)
00024 // SSH Required Resources
00025 #include <ssh/NetBurner/UserAuthManager.h>
00026 #endif
00027
00033 #define CMD_OK (0)
00034 #define CMD_CLOSE (1)
00035 #define CMD_FAIL (2)
00036 #define CMD_AUTH_FAIL (3)
00037 #define CMD_TO_MANY_FDS (4)
00038 #define CMD_FAILED_SSH_KEY_GEN (5)
00048 enum ListenOn : int8_t
00049 {
00050     eListenOnTcp = 1,
00051     eListenOnSsh = 2,
00052 };
00065 extern int (*CmdAuthenticateFunc)(const char *name, const char *passwd);
00066
00067 #if defined(NB_SSH_SUPPORTED)
00080 extern int (*CmdAuthenticateSshFunc)(const char *name, const char *authVal, AuthType authType);
00081 #endif
00082
00097 extern int (*CmdCmd_func)(const char *command, FILE *fRespondto, void *pData);
00098
00115 extern int (*CmdChar_func)(char command, FILE *fRespondto, void *pData);
00116
00130 extern void *(*CmdConnect_func)(FILE *fRespondto);
00131
00142 extern void (*CmdPrompt_func)(FILE *fRespondto, void *pData);
00143
00149 #define CMD_DIS_CAUSE_TIMEOUT (1)
00150 #define CMD_DIS_CAUSE_CLOSED (2)
00151 #define CMD_DIS_SOCKET_CLOSED (3)
00152 #define CMD_DIS_AUTH_FAILED (4)
00166 extern void (*CmdDisConnect_func)(FILE *fRespondto, int cause, void *pData);
00167
00168 extern int CmdIdleTimeout;
00170
00171 extern const char
00172     *Cmdlogin_prompt;
00173
00182 int CmdStartCommandProcessor(int priority);
00183
00184 /* Stop the running process and close all open sockets/fd's */
00185 void CmdStopCommandProcessor();
00186
00199 int CmdAddCommandFd(int fd, bool require_auth, bool time_out_conn, bool local_echo = true);
00200
00209 int CmdRemoveCommandFd(int fd);
00210
00221 int CmdListenOnTcpPort(uint16_t port, int do_telnet_processing, int max_connections);
00222
00233 int CmdListenQuietOnTcpPort(uint16_t port, int do_telnet_processing, int max_connections);
00234
00235 #if defined(NB_SSH_SUPPORTED)
00246 int CmdListenOnSshPort(uint16_t port, int max_connections);
00247
00258 int CmdListenQuietOnSshPort(uint16_t port, int max_connections);
00259 #endif
00260
00269 int CmdStopListeningOnTcpPort(uint16_t port);
00270
00271 #if defined(NB_SSH_SUPPORTED)
00280 int CmdStopListeningOnSshPort(uint16_t port);
00281 #endif
00282
00290 void SendToAll(const char *buffer, int len, bool include_serial_ports);
00291
00292 #endif /* _NB_COMMAND_H */
00293
```

24.301 config_netobj.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef CONFIG_NETOBJ_H
00006 #define CONFIG_NETOBJ_H
00007 #include <config_obj.h>
00008
00009 class config_localname : public config_string
00010 {
00011     public:
00022         config_localname(config_obj &owner, NBString def_val, const char *name, const char *desc =
      NULL)
00023             : config_string(owner, def_val, name, desc){};
00024
00025         config_localname(config_localname &&l)
00026             : config_string((config_string&&)l)
00027         { }
00028
00038         config_localname(NBString def_val, const char *name, const char *desc = NULL) :
      config_string(def_val, name, desc){};
00039
00050         config_localname(config_obj &owner, const char *def_val, const char *name, const char *desc =
      NULL)
00051             : config_string(owner, def_val, name, desc){};
00052
00062         config_localname(const char *def_val, const char *name, const char *desc = NULL) :
      config_string(def_val, name, desc){};
00063
00064         virtual void CommitTestedValue(uint32_t permission_mask);
00065
00066 };
00067
00068
00069 class I4Record;
00070
00071 class CUR_IPADDR4 : public config_value
00072 {
00073    public:
00074     IPADDR4 i4;
00075     ConfigEndMarker;
00076     CUR_IPADDR4(const char *name, const char *desc = NULL) : config_value(name, desc)
00077     {
00078         SetFlag(fConfigReadOnly | fConfigNoSave);
00079         i4 = 0;
00080     }
00081
00082     CUR_IPADDR4(CUR_IPADDR4 &&I)
00083         : config_value((config_value&&)I), i4(I.i4)
00084     {
00085
00086     }
00087
00088     virtual void GetTextValue(NBString &s) { s.siprintf("\"%hI\"", i4); };
00089
00090     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs) { return eUnchanged; }
00091     virtual void CommitTestedValue(uint32_t permission_mask){};
00092
00093     CUR_IPADDR4 &operator=(const CUR_IPADDR4 &v)
00094     {
00095         i4 = v.i4;
00096         return *this;
00097     };
00098     CUR_IPADDR4 &operator=(const IPADDR4 &v)
00099     {
00100         i4 = v;
00101         return *this;
00102     };
00103     CUR_IPADDR4 &operator=(const uint32_t v)
00104     {
00105         i4 = v;
00106         return *this;
00107     };
00108     virtual void ExtendedSchema(int fd, int indent, bool pretty) { DoSchemaLine(fd, "format", "ipv4",
      indent, pretty); };
00109     virtual void GetTypeValue(NBString &s) { s = "string"; };
00110     operator IPADDR4() { return i4; };
00111
00112     inline bool IsNull() const { return i4.IsNull(); };
00113     inline bool NotNull() const { return i4.NotNull(); };
00114     inline void SetNull() { return i4.SetNull(); };
00115     inline bool IsLoopBack() const { return i4.IsLoopBack(); };
00116     inline bool IsMultiCast() const { return i4.IsMultiCast(); };
00117     inline bool IsGlobalBroadCast() const { return i4.IsGlobalBroadCast(); };
00118     inline bool IsAutoIP() { return i4.IsAutoIP(); };
```

```
00119 };
00120
00121 inline bool operator==(const IPADDR4 i, const CUR_IPADDR4 &j)
00122 {
00123     return i == j.i4;
00124 }
00125 inline bool operator!=(const IPADDR4 i, const CUR_IPADDR4 &j)
00126 {
00127     return i != j.i4;
00128 }
00129 inline bool operator>(const IPADDR4 i, const CUR_IPADDR4 &j)
00130 {
00131     return i > j.i4;
00132 }
00133 inline bool operator<(const IPADDR4 i, const CUR_IPADDR4 &j)
00134 {
00135     return i > j.i4;
00136 }
00137
00138 inline bool operator==(const CUR_IPADDR4 & i, const IPADDR4 j)
00139 {
00140     return i.i4 == j;
00141 }
00142 inline bool operator!=(const CUR_IPADDR4 & i, const IPADDR4 j)
00143 {
00144     return i.i4 != j;
00145 }
00146 inline bool operator>(const CUR_IPADDR4 & i, const IPADDR4 j)
00147 {
00148     return i.i4 > j;
00149 }
00150 inline bool operator<(const CUR_IPADDR4 & i, const IPADDR4 j)
00151 {
00152     return i.i4 > j;
00153 }
00154
00155 inline bool operator==(const uint32_t i, const CUR_IPADDR4 &j)
00156 {
00157     return i == j.i4;
00158 }
00159 inline bool operator!=(const uint32_t i, const CUR_IPADDR4 &j)
00160 {
00161     return i != j.i4;
00162 }
00163 inline bool operator>(const uint32_t i, const CUR_IPADDR4 &j)
00164 {
00165     return i > j.i4;
00166 }
00167 inline bool operator<(const uint32_t i, const CUR_IPADDR4 &j)
00168 {
00169     return i < j.i4;
00170 }
00171
00172 inline bool operator==(const CUR_IPADDR4 &i, const uint32_t j)
00173 {
00174     return i.i4 == j;
00175 }
00176 inline bool operator!=(const CUR_IPADDR4 &i, const uint32_t j)
00177 {
00178     return i.i4 != j;
00179 }
00180 inline bool operator>(const CUR_IPADDR4 &i, const uint32_t j)
00181 {
00182     return i.i4 > j;
00183 }
00184 inline bool operator<(const CUR_IPADDR4 &i, const uint32_t j)
00185 {
00186     return i.i4 < j;
00187 }
00188
00189 /* Some predefined records */
00190 class I4Record : public config_obj
00191 {
00192     void *m_pBlock_if;
00193
00194    public:
00195     config_chooser mode{"Mode", "DHCP", "DHCP,DHCP w Fallback,Static,Disabled",
00196                         "DHCP,DHCP with backup static address,Static preconfigured address,Disabled"};
00197     config_IPADDR4 addr{"0.0.0.0", "StaticAddr", "Configured IP Address"};
00198     config_IPADDR4 mask{"0.0.0.0", "StaticMask", "Configured IP Mask"};
00199     config_IPADDR4 gate{"0.0.0.0", "StaticGate", "Configured IP Gateway"};
00200     config_IPADDR4 dns1{"0.0.0.0", "StaticDNS1", "Configured IP DNS(1)"};
00201     config_IPADDR4 dns2{"0.0.0.0", "StaticDNS2", "Configured IP DNS(2)"};
00202
00203     config_bool autoip{true, "AutoIPEn", "Enable Auto IP"};
00204
00205     //These are the only values used activly by the system
```

```
00206     CUR_IPADDR4 cur_addr{"ActiveAddr", "Current IPV4 address in use"};
00207     CUR_IPADDR4 cur_mask{"ActiveMask", "Current IPV4 mask in use"};
00208     CUR_IPADDR4 cur_gate{"ActiveGate", "Current IPV4 gateway in use"};
00209     CUR_IPADDR4 cur_dns1{"ActiveDNS1", "Current IPV4 dns(1) in use"};
00210     CUR_IPADDR4 cur_dns2{"ActiveDNS2", "Current IPV4 dns(2) in use"};
00211     CUR_IPADDR4 cur_auto{"AutoIPAddr", "Current IPV4 auto address in use"};
00212     config_bool mac_dns {true,"EnableMacmDNS","Enable nburnxxxxxx.local using mac address"};
00213     config_localname  local_name{"","LocalName","The local name to resolve with mDNS (.local
      assumed)"};
00214
00215     ConfigEndMarker;
00216
00217     I4Record(const char *name, const char *desc = NULL) : config_obj(name, desc) { m_pBlock_if = 0; };
00218
00219     I4Record(I4Record &&IR);
00220
00221     void Attach(void *pb) { m_pBlock_if = pb; }
00222     void GetReportItem(NBString &s, int item);
00223 };
00224
00225 #ifdef IPV6
00226 class I6Record;
00227
00228 class Dynamic_IPADDR : public config_value
00229 {
00230     int m_nItem;
00231     I6Record *m_pI6;
00232     ConfigEndMarker;
00233
00234    public:
00235     virtual void GetTextValue(NBString &s);
00236
00237     Dynamic_IPADDR(const char *name, int item, I6Record *pI6, const char *desc = NULL) :
      config_value(name, desc)
00238     {
00239         m_nItem = item;
00240         m_pI6 = pI6;
00241         SetFlag(fConfigReadOnly | fConfigNoSave);
00242     }
00243     Dynamic_IPADDR(Dynamic_IPADDR &&di);
00244
00245     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs) { return eUnchanged; }
00246     virtual void CommitTestedValue(uint32_t permission_mask){};
00247     void AddToJsonOutObject(ParsedJsonDataSet &jo, const char *name);
00248     virtual void ExtendedSchema(int fd, int indent, bool pretty) { DoSchemaLine(fd, "format", "ipv6",
      indent, pretty); };
00249     virtual void GetTypeValue(NBString &s) { s = "array"; };
00250 };
00251
00252 class I6Record : public config_obj
00253 {
00254     int m_nIf;
00255
00256    public:
00257     config_chooser mode{"Mode", "DHCP", "DHCP,DHCP w Fallback,Static,Disabled",
00258                         "DHCP,DHCP with backup static address,Static preconfigured address,Disabled"};
00259     config_IPADDR addr{"0::0", "StaticAddr", "Configured IPV6 address"};
00260     config_IPADDR dns1{"0::0", "StaticDNS1", "Configured IPV6 DNS(1)"};
00261     config_IPADDR dns2{"0::0", "StaticDNS2", "Configured IPV6 DNS(2)"};
00262
00263     Dynamic_IPADDR address_list{"ActiveAddr", 0, this, "Active currently in user IPV6 addres(s)"};
00264     Dynamic_IPADDR dns_list{"ActiveDNS", 1, this, "Active currently in user IPV6 DNS(s)"};
00265     Dynamic_IPADDR route_list{"ActiveRoute", 2, this, "Active currently in user IPV6 routes"};
00266
00267     ConfigEndMarker;
00268
00269     I6Record(const char *name, const char *desc = NULL) : config_obj(name, desc) { m_nIf = -1; };
00270     I6Record(I6Record &&i6);
00271     void Attach(int nif) { m_nIf = nif; };
00272     void ReportGetTextValue(NBString &s, int item);
00273     int GetnIf() { return m_nIf; };
00274 };
00275 #endif
00276
00277 #endif
```

24.302 config_obj.h File Reference

Configuration object header file.

```
#include <buffers.h>
#include <nbstring.h>
#include <string.h>
```

```
#include <utils.h>
#include <plat_cfg_types.h>
```

Classes

- class config_obj

  Base class used to create configuration objects.
- class config_value

  Base class used to create a configuration value.
- class config_uint

  Unsigned 32-bit Integer Configuration Variable.
- class config_int

  Signed 32-bit Integer Configuration Variable.
- class config_double

  Double Float Configuration Variable.
- class config_bool

  Boolean Configuration Variable.
- class config_string

  String Configuration Variable.
- class config_pass

  Password string Configuration Variable.
- class config_IPADDR4

  Configuration Variable for IPADDR4 (IPv4) object types.
- class config_IPADDR

  Configuration Variable for IPADDR (IPv6) object type.
- class config_MACADR

  Configuration Variable for MACADR object type.
- class config_chooser

  Chooser Configuration Variable - Select From a List of Items.

Functions

- void SaveConfigToStorage ()

  Save configuration to flash storage.

Variables

- const uint32_t **fConfigValueLeaf** = 0x01

  Value is a leaf.
- const uint32_t **fConfigReadOnly** = 0x02

  Variable is read-only.
- const uint32_t **fConfigModified** = 0x04

  Variable has been modified, but not yet saved.
- const uint32_t **fConfigHidden** = 0x08

  Not visible to configuration web server display.
- const uint32_t **fConfigNoSave** = 0x10

  Do not save to flash memory when save functions are called.
- const uint32_t **fConfigNoObscure** = 0x20

  Do not obscure the value.
- const uint32_t **fConfigNeedReboot** = 0x40

  System reboot required for changes to take effect.

- const uint32_t **fConfigNoReload** = 0x80

  Disable reloading value from flash during a call to ReloadFromFlash.
- const uint32_t **fConfigDetached** = 0x100

  Disable reloading value from flash during a call to ReloadFromFlash.

24.302.1 Detailed Description

Configuration object header file.

24.303 config_obj.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00039 #ifndef CONFIG_OBJ_H
00040 #define CONFIG_OBJ_H
00041
00042 #include <buffers.h>
00043 #include <nbstring.h>
00044 #include <string.h>
00045 #include <utils.h>
00046
00047 void ShowTree();
00048
00049 enum ConfigTestResult
00050 {
00051     eUnchanged,
00052     eOk,
00053     eBad
00054 };
00055
00063 const uint32_t fConfigValueLeaf = 0x01;
00064 const uint32_t fConfigReadOnly = 0x02;
00065 const uint32_t fConfigModified = 0x04;
00066 const uint32_t fConfigHidden = 0x08;
00067 const uint32_t fConfigNoSave = 0x10;
00068 const uint32_t fConfigNoObscure = 0x20;
00069 const uint32_t fConfigNeedReboot = 0x40;
00070 const uint32_t fConfigNoReload = 0x80;
00071 const uint32_t fConfigDetached = 0x100;
00074 /* Config Mask Values */
00075 const uint32_t PermitFlashFromStorage = (0x80000000);
00076
00077 class config_leaf;
00078 class config_obj;
00079 class ParsedJsonDataSet;
00080 class RootParseStateInfo;
00081
00082 typedef void(LeafCallBack)(config_leaf *p, void *pextra);
00083
00084 /*
00085  * Configuration leaf class.
00086  * Used to manage the configuration tree, internal use only.
00087  */
00088 class config_leaf
00089 {
00090     // We explicitly *do not allow* copy construction, as the only likely usage
00091     //  would be passing to variadic functions *which will not know this is non-POD*
00092     config_leaf(config_leaf &rhs) = delete;
00093   protected:
00094     config_leaf *FindChild(const char *&cp);
00095     void RootParse(RootParseStateInfo &rpsi);
00096     static void FixTree(config_leaf* root);
00097
00098
00099     virtual void RemoveFromRootList();
00100
00101   public:
00102     void FdPrintTree(int fd, config_leaf *pl, int n, bool pretty, uint32_t inhibit_mask =
      fConfigHidden, bool bRawValue = false);
00103     void FdPrintSchema(int fd, config_leaf *pl, int n, bool pretty, uint32_t inhibit_mask =
      fConfigHidden);
00104     void SchemaOptions(int fd, int indent, bool pretty);
00105     void ForEachLeaf(LeafCallBack *pc, void *pextra, bool siblings = false);
00106     bool ParseBuffer(fifo_buffer_storage &rxbuf, uint32_t permissions, const char *where);
00107     bool ParseBlob(uint8_t *pdata, int len, uint32_t permissions);
00108     bool ParseFd(int fd, uint32_t permissions, config_leaf * pParseRoot = NULL);
00109     config_obj *FindParent() { return pParent; };
```

```
00110     static config_leaf *FindConfigLeaf(const unsigned char *name, config_leaf *pBranch = NULL);
00111
00112     //Call if a config object is not static
00113     void FixNonStaticObject();
00114
00115     void RemoveFromConfigTree();
00116
00117     const char *pName;
00118     const char *pDescription;
00119     config_leaf *pNextSibling;
00120     config_obj *pParent;
00121     config_leaf *pChildren;
00122     config_leaf *pGList;
00123     static config_leaf *pRootList;
00124     static config_leaf *pDetachList;
00125
00126     uint32_t m_Flags;
00127     void DoSchemaLine(int fd, const char *name, const char *value, int indent, bool pretty, bool
      quoted = true);
00128
00129    public:
00130     bool NameMatch(const char *cp);
00131     virtual const char *GetSortNameValue() { return pName; };
00132     virtual void GetDescriptionValue(NBString &s) { s = pDescription; };
00133     virtual void GetNameValue(NBString &s) { s = pName; };
00134     virtual void GetTextValue(NBString &s) { s = pName; };
00135     virtual void GetRawValue(NBString &s) { return GetTextValue(s); };
00136     virtual void GetTypeValue(NBString &s) { s = "unknown"; };
00137     virtual void ExtendedSchema(int fd, int indent, bool pretty){};
00138     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs) { return eOk; };
00139     // This is named so error messages make sense to user
00140     // Old name was GetExtent
00141     virtual int Missing_ConfigEndMarker(void *&startp) = 0;
00142
00143     void ClearModified();
00144
00145     bool MatchId(ParsedJsonDataSet &pjs);
00146
00147     virtual void CommitTestedValue(uint32_t permission_mask){};
00148
00149     virtual ~config_leaf();
00150
00151     config_leaf(bool bDetached=false)
00152     {
00153         pNextSibling = NULL;
00154         pChildren = NULL;
00155         m_Flags = 0;
00156         pParent = NULL;
00157         if (bDetached)
00158         {
00159            pGList=pDetachList;
00160            pDetachList=this;
00161            m_Flags|=fConfigDetached;
00162         }
00163         else
00164         {
00165             pGList = pRootList;
00166             pRootList = this;
00167         }
00168     }
00169     config_leaf(config_leaf &&l);
00170
00171     static void DiagShow();
00172     static void FixTree();
00173     void FindUnknownParent();
00174
00175     void ShowOne();
00176
00177     int compare(config_leaf *pl)
00178     {
00179         if (pl == NULL) return -1;
00180         return strcmp(GetSortNameValue(), pl->GetSortNameValue());
00181     };
00182
00183     void GetFullName(NBString &s);
00184     void GetBranchName(NBString &s, config_leaf *branchRoot);
00185     void RenderToFd(int fd, bool pretty = false, uint32_t mask = fConfigHidden, bool bRawValue =
      false);
00186     void RenderSchemaToFd(int fd, bool pretty = false, uint32_t mask = fConfigHidden);
00187     uint32_t GetFlags() { return m_Flags; }
00188     void SetFlag(uint32_t flag) { m_Flags |= flag; };
00189     void SetBranchFlag(uint32_t flag);
00190     void ClrFlag(uint32_t flag) { m_Flags &= (~flag); };
00191     void ReloadFromFlash();
00192
00193     friend void HtmlLeafRender(int fd, config_leaf *pl, int eMode, int len, const char *extra);
00194 };
```

```
00195
00256 class config_obj : public config_leaf
00257 {
00258    protected:
00259     config_obj(bool bDetached=false):config_leaf(bDetached){}
00260     config_obj *pMasterObjectLink;
00261     static config_obj *pObjList;
00262
00263     virtual void RemoveFromRootList();
00264    public:
00265     void InstallUnderMe(config_leaf &ltoadd);
00266     void RemoveFromChildren(config_leaf &ltoremove);
00267
00280     config_obj(config_obj &owner, const char *name, const char *desc)
00281     {
00282         pName = name;
00283         pDescription = desc;
00284         pChildren = NULL;
00285         pMasterObjectLink = pObjList;
00286         pObjList = this;
00287         pParent = &owner;
00288     }
00289     config_obj(config_obj &&o);
00290
00303     config_obj(const char *name, const char *desc)
00304     {
00305         pName = name;
00306         pDescription = desc;
00307         pChildren = NULL;
00308         pParent = NULL;
00309         pMasterObjectLink = pObjList;
00310         pObjList = this;
00311     }
00312
00313
00314
00315     bool DoIContain(config_leaf *pl);
00316
00317     /*
00318      * These two functions, along with GetTextValue(), can be used to create a custom object
00319      * class, including the responsibility for the JSON serialization.
00320      *
00321      * GetTextValue():  Must return a string with the object encoded in it in JSON format
00322      *
00323      * TestNewValue():  Takes a parsed JSON object and extracts the values from that tree.
00324      *                  This is where parameters can be tested for validity, and if not valid,
00325      *                  the entire set is marked as invalid.
00326      *
00327      * CommitTestedValue(): Commit the values extracted by TestNewValue()
00328      *
00329      * This enables functionality such as parsing a set of values to determine if the SET is
00330      * valid, rather than just an individual value.
00331      */
00332     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00333     virtual void CommitTestedValue(uint32_t permission_mask);
00334
00335     // This is named so error messages make sense to user, old name was GetExtent
00336     virtual int Missing_ConfigEndMarker(void *&startp)
00337     {
00338         startp = this;
00339         return sizeof(*this);
00340     };
00341
00349     virtual void GetTextValue(NBString &s);
00350
00358     virtual void GetTypeValue(NBString &s) { s = "object"; };
00359
00360     friend class config_leaf;
00361 };
00362
00363 /*
00364  * Configuration Root object class
00365  */
00366 class root_obj : public config_obj
00367 {
00368    public:
00369     root_obj()
00370     { pName = "Config";
00371       pDescription = "Root of config tree";
00372     }
00373
00374     // This is named so error messages make sense to user, old name was GetExtent
00375     virtual int Missing_ConfigEndMarker(void *&startp)
00376     {
00377         startp = this;
00378         return sizeof(*this);
00379     };
```

```
00380 };
00381
00382 class detached_root_obj : public config_obj
00383 {
00384    public:
00385     detached_root_obj():config_obj(true)
00386     { pName = "detached";
00387       pDescription = "Root of detached tree";
00388     }
00389
00390     // This is named so error messages make sense to user, old name was GetExtent
00391     virtual int Missing_ConfigEndMarker(void *&startp)
00392     {
00393         startp = this;
00394         return sizeof(*this);
00395     };
00396 };
00397
00398
00399
00400
00401 extern root_obj root;
00402 extern detached_root_obj detached;
00403
00425 class config_value : public config_leaf
00426 {
00427    public:
00428     virtual void GetTextValue(NBString &s) = 0;
00429
00430    protected:
00443     config_value(config_obj &owner, const char *name, const char *desc)
00444     {
00445         pName = name;
00446         pDescription = desc;
00447         owner.InstallUnderMe(*this);
00448         pChildren = NULL;
00449         m_Flags |= fConfigValueLeaf;
00450     }
00451
00464     config_value(const char *name, const char *desc)
00465     {
00466         pName = name;
00467         pDescription = desc;
00468         pParent = NULL;
00469         pChildren = NULL;
00470         m_Flags |= fConfigValueLeaf;
00471     }
00472
00473     config_value(config_value &&v);
00474 };
00475
00483 class config_uint : public config_value
00484 {
00485    protected:
00486     uint32_t val;
00487     uint32_t pend_val;
00488
00489    public:
00497     virtual void GetTextValue(NBString &s) { s.siprintf("%u", val); };
00498
00507     config_uint(config_obj &owner, uint32_t def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
00508     {
00509         val = def_val;
00510         pend_val = val;
00511     }
00512
00520     config_uint(uint32_t def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
00521     {
00522         val = def_val;
00523         pend_val = val;
00524     }
00525
00526     config_uint(config_uint &&u);
00527
00528     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00529     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
00530
00531     // Assignment operators
00532
00544     operator uint32_t() const { return val; };
00545
00551     config_uint &operator=(const uint32_t i)
00552     {
00553         pend_val = val = i;
00554         return *this;
```

```
00555     };
00556
00562     config_uint &operator=(const config_uint &ci)
00563     {
00564         pend_val = val = ci.val;
00565         return *this;
00566     };
00567
00568     // This is named so error messages make sense to user. Old name was GetExtent
00569     virtual int Missing_ConfigEndMarker(void *&startp)
00570     {
00571         startp = this;
00572         return sizeof(*this);
00573     };
00574
00582     virtual void GetTypeValue(NBString &s) { s = "integer"; };
00583 };
00584
00585 class config_hex_uint : public config_uint
00586 {
00587
00588     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00589
00590 public:
00599     config_hex_uint(config_obj &owner, uint32_t def_val, const char *name, const char *desc = NULL) :
      config_uint(owner,def_val, name, desc)
00600     {
00601     }
00602
00610     config_hex_uint(uint32_t def_val, const char *name, const char *desc = NULL) :
      config_uint(def_val,name, desc)
00611     {
00612     }
00613
00614     virtual void GetTextValue(NBString &s) { s.siprintf("\"%08X\"", val); };
00615     virtual void GetTypeValue(NBString &s) { s = "hexuint"; };
00616
00617 };
00618
00619
00620
00628 class config_int : public config_value
00629 {
00630   protected:
00631     int val;
00632     int pend_val;
00633
00634   public:
00642     virtual void GetTextValue(NBString &s) { s.siprintf("%d", val); };
00643
00652     config_int(config_obj &owner, int def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
00653     {
00654         val = def_val;
00655         pend_val = val;
00656     }
00657
00665     config_int(int def_val, const char *name, const char *desc = NULL) : config_value(name, desc)
00666     {
00667         val = def_val;
00668         pend_val = val;
00669     }
00670
00671     config_int(config_int &&i);
00672
00673     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00674
00675     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
00676
00677     // Assignment operators
00678
00690     operator int() const { return val; };
00691
00697     config_int &operator=(const int i)
00698     {
00699         pend_val = val = i;
00700         return *this;
00701     };
00702
00709     config_int &operator=(const config_int &ci)
00710     {
00711         pend_val = val = ci.val;
00712         return *this;
00713     };
00714
00715     // This is named so error messages make sense to user, old name was GetExtent
00716     virtual int Missing_ConfigEndMarker(void *&startp)
```

```
00717     {
00718         startp = this;
00719         return sizeof(*this);
00720     };
00721
00729     virtual void GetTypeValue(NBString &s) { s = "integer"; };
00730 };
00731
00739 class config_double : public config_value
00740 {
00741     double val;
00742     double pend_val;
00743
00744   public:
00752     virtual void GetTextValue(NBString &s) { s.sprintf("%g", val); };
00753
00762     config_double(config_obj &owner, double def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
00763     {
00764         val = def_val;
00765         pend_val = val;
00766     }
00767
00775     config_double(double def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
00776     {
00777         val = def_val;
00778         pend_val = val;
00779     }
00780
00781     config_double(config_double &&d);
00782
00783     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00784     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
00785
00786     // Assignment operators
00787
00799     operator int() const { return val; };
00800
00812     operator float() const { return val; };
00813
00825     operator double() const { return val; };
00826
00832     config_double &operator=(const double d)
00833     {
00834         pend_val = val = d;
00835         return *this;
00836     };
00837
00843     config_double &operator=(const config_double &ci)
00844     {
00845         pend_val = val = ci.val;
00846         return *this;
00847     };
00848
00849     // This is named so error messages make sense to user. old name was GetExtent
00850     virtual int Missing_ConfigEndMarker(void *&startp)
00851     {
00852         startp = this;
00853         return sizeof(*this);
00854     };
00855
00863     virtual void GetTypeValue(NBString &s) { s = "float"; };
00864 };
00865
00866 /*
00867  * Class used for system status reports, for internal use only
00868  */
00869 class config_report : public config_value
00870 {
00871   protected:
00872     const char *m_value;
00873
00874   public:
00875     config_report(config_obj &owner, const char *value, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
00876     {
00877         m_Flags = fConfigReadOnly | fConfigNoSave;
00878         m_value = value;
00879     }
00880     config_report(const char *value, const char *name, const char *desc = NULL) : config_value(name,
      desc)
00881     {
00882         m_Flags = fConfigReadOnly | fConfigNoSave;
00883         m_value = value;
00884     }
00885
```

```
00886     config_report(config_report &&r);
00887
00888     virtual void GetTextValue(NBString &s);
00889     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00890     virtual void CommitTestedValue(uint32_t permission_mask);
00891     // This is named so error messages make sense to user
00892     // Old name was GetExtent
00893     virtual int Missing_ConfigEndMarker(void *&startp)
00894     {
00895         startp = this;
00896         return sizeof(*this);
00897     };
00898     virtual void GetTypeValue(NBString &s) { s = "string"; };
00899
00900     const char *c_str() { return m_value; };
00901     void ModifyValue(const char *nv) { m_value = nv; };
00902 };
00903
00911 class config_bool : public config_value
00912 {
00913    protected:
00914     bool val;
00915     bool pend_val;
00916
00917    public:
00925     virtual void GetTextValue(NBString &s)
00926     {
00927         if (val)
00928             s = "true";
00929         else
00930             s = "false";
00931     };
00932
00941     config_bool(config_obj &owner, bool def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
00942     {
00943         val = def_val;
00944         pend_val = val;
00945     }
00946
00954     config_bool(bool def_val, const char *name, const char *desc = NULL) : config_value(name, desc)
00955     {
00956         val = def_val;
00957         pend_val = val;
00958     }
00959     config_bool(config_bool &&b);
00960
00961     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
00962     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
00963
00964     // Assignemt operators
00965
00977     operator bool() const { return val; };
00978
00984     config_bool &operator=(const bool v)
00985     {
00986         pend_val = val = v;
00987         return *this;
00988     };
00989
00995     config_bool &operator=(const config_bool &cb)
00996     {
00997         pend_val = val = cb.val;
00998         return *this;
00999     };
01000
01006     config_bool &operator=(const int i)
01007     {
01008         pend_val = val = (i != 0);
01009         return *this;
01010     };
01011
01012     // This is named so error messages make sense to user, old name was GetExtent
01013     virtual int Missing_ConfigEndMarker(void *&startp)
01014     {
01015         startp = this;
01016         return sizeof(*this);
01017     };
01018
01024     virtual void GetTypeValue(NBString &s) { s = "boolean"; };
01025 };
01026
01034 class config_string : public config_value
01035 {
01036    protected:
01037     NBString val;
01038     NBString pend_val;
```

```
01039     NBString enum_list;
01040
01041   public:
01049     virtual void GetTextValue(NBString &s);
01050 //    {
01051 //        s = "\"";
01052 //        s += val;
01053 //        s += "\"";
01054 //    };
01055
01056     config_string(config_string &&s);
01057
01068     config_string(config_obj &owner, NBString def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01069     {
01070         val = def_val;
01071         pend_val = val;
01072     }
01073
01083     config_string(NBString def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01084     {
01085         pName = name;
01086         val = def_val;
01087         pend_val = val;
01088     }
01089
01100     config_string(config_obj &owner, const char *def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01101     {
01102         val = def_val;
01103         pend_val = val;
01104     }
01105
01115     config_string(const char *def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01116     {
01117         val = def_val;
01118         pend_val = val;
01119     }
01120
01121     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
01122     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
01123
01130     void SetEnumList(NBString s) { enum_list = s; };
01131
01132     // Assignment Operators
01133
01139     operator NBString() const { return val; };
01140
01146     config_string &operator=(const char *p)
01147     {
01148         pend_val = val = p;
01149         return *this;
01150     };
01151
01157     config_string &operator=(const NBString &s)
01158     {
01159         pend_val = val = s;
01160         return *this;
01161     };
01162
01168     config_string &operator=(const config_string &ci)
01169     {
01170         pend_val = val = ci.val;
01171         return *this;
01172     };
01173
01174     // This is named so error messages make sense to user, old name was GetExtent
01175     virtual int Missing_ConfigEndMarker(void *&startp)
01176     {
01177         startp = this;
01178         return sizeof(*this);
01179     };
01180
01186     inline const char *c_str() const { return val.c_str(); };
01187
01193     inline size_t length() const { return val.length(); };
01194
01202     inline const char &operator[](size_t pos) const { return val[pos]; };
01203
01211     virtual void GetTypeValue(NBString &s) { s = "string"; };
01212
01213     virtual void ExtendedSchema(int fd, int indent, bool pretty);
01214
01215     /* *
01216      * @brief Perform a string interpolation and place the finished interpolation
```

```
01217      *  in the Destination string
01218      *
01219      * @param dest  Destination string
01220      *
01221      * @returns Whether the interpolation was successful
01222      */
01223     bool Interpolate(NBString &dest)
01224     {
01225         return val.Interpolate(dest);
01226     }
01227
01228     friend class config_pass;
01229     friend class config_chooser;
01230 };
01231
01241 class config_pass : public config_string
01242 {
01243    public:
01254     config_pass(config_obj &owner, NBString def_val, const char *name, const char *desc = NULL)
01255         : config_string(owner, def_val, name, desc){};
01256
01266     config_pass(NBString def_val, const char *name, const char *desc = NULL) : config_string(def_val,
      name, desc){};
01267
01278     config_pass(config_obj &owner, const char *def_val, const char *name, const char *desc = NULL)
01279         : config_string(owner, def_val, name, desc){};
01280
01290     config_pass(const char *def_val, const char *name, const char *desc = NULL) :
      config_string(def_val, name, desc){};
01291
01292     config_pass(config_pass &&s);
01293
01301     virtual void GetTextValue(NBString &s);
01302
01310     virtual void GetRawValue(NBString &s);
01311
01312     virtual void CommitTestedValue(uint32_t permission_mask);
01313
01314     // Add help pop-ups for web page display
01315     virtual void ExtendedSchema(int fd, int indent, bool pretty)
01316     {
01317         config_string::ExtendedSchema(fd, indent, pretty);
01318         DoSchemaLine(fd, "format", "password", indent, pretty);
01319     };
01320
01326     operator NBString() const { return val; };
01327
01333     config_pass &operator=(const char *p)
01334     {
01335         pend_val = val = p;
01336         return *this;
01337     };
01338
01344     config_pass &operator=(const NBString &s)
01345     {
01346         pend_val = val = s;
01347         return *this;
01348     };
01349
01355     config_pass &operator=(const config_string &ci)
01356     {
01357         pend_val = val = ci.val;
01358         return *this;
01359     };
01360
01366     config_pass &operator=(const config_pass &ci)
01367     {
01368         pend_val = val = ci.val;
01369         return *this;
01370     };
01371 };
01372
01381 class config_IPADDR4 : public config_value
01382 {
01383     IPADDR4 val;
01384     IPADDR4 pend_val;
01385
01386    public:
01395     virtual void GetTextValue(NBString &s) { s.siprintf("\"%hI\"", val); };
01396
01407     config_IPADDR4(config_obj &owner, IPADDR4 def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01408     {
01409         val = def_val;
01410         pend_val = val;
01411     }
01412
```

```
01422     config_IPADDR4(IPADDR4 def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01423     {
01424         val = def_val;
01425         pend_val = val;
01426     }
01427
01428     config_IPADDR4(config_IPADDR4 &&I);
01429
01440     config_IPADDR4(config_obj &owner, const char *def_val, const char *name, const char *desc = NULL)
      : config_value(owner, name, desc)
01441     {
01442         IPADDR4 i4;
01443         i4.SetFromAscii(def_val);
01444         val = i4;
01445         pend_val = val;
01446     }
01447
01457     config_IPADDR4(const char *def_val, const char *name, const char *desc = NULL) :
      config_value(name, desc)
01458     {
01459         IPADDR4 i4;
01460         i4.SetFromAscii(def_val);
01461         val = i4;
01462         pend_val = val;
01463     }
01464
01465     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
01466     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
01467
01468     // Assignment operators
01469
01475     operator IPADDR4() const { return val; };
01476
01484     inline bool IsNull() const { return val.IsNull(); };
01485
01493     inline bool NotNull() const { return !(val.IsNull()); };
01494
01500     inline void SetNull()
01501     {
01502         val.SetNull();
01503         pend_val = val;
01504     };
01505
01511     config_IPADDR4 &operator=(const IPADDR4 &i4)
01512     {
01513         pend_val = val = i4;
01514         return *this;
01515     };
01516
01522     config_IPADDR4 &operator=(const config_IPADDR4 &ci)
01523     {
01524         pend_val = val = ci.val;
01525         return *this;
01526     };
01527
01528     // This is named so error messages make sense to user, old name was GetExtent
01529     virtual int Missing_ConfigEndMarker(void *&startp)
01530     {
01531         startp = this;
01532         return sizeof(*this);
01533     };
01534
01542     virtual void GetTypeValue(NBString &s) { s = "string"; };
01543
01544     // Add web page help
01545     virtual void ExtendedSchema(int fd, int indent, bool pretty) { DoSchemaLine(fd, "format", "ipv4",
      indent, pretty); };
01546 };
01547
01548 #ifdef IPV6
01557 class config_IPADDR : public config_value
01558 {
01559     IPADDR val;
01560     IPADDR pend_val;
01561
01562   public:
01570     virtual void GetTextValue(NBString &s) { s.siprintf("\"%I\"", val); };
01571
01582     config_IPADDR(config_obj &owner, IPADDR def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01583     {
01584         val = def_val;
01585         pend_val = val;
01586     }
01587
01588     config_IPADDR(config_IPADDR &&I);
```

```
01589
01599     config_IPADDR(IPADDR def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01600     {
01601         val = def_val;
01602         pend_val = val;
01603     }
01604
01615     config_IPADDR(config_obj &owner, const char *def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01616     {
01617         IPADDR i6;
01618         i6.SetFromAscii(def_val);
01619         val = i6;
01620         pend_val = val;
01621     }
01622
01632     config_IPADDR(const char *def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01633     {
01634         IPADDR i6;
01635         i6.SetFromAscii(def_val);
01636         val = i6;
01637         pend_val = val;
01638     }
01639
01640     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
01641     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
01642
01648     operator IPADDR() const { return val; };
01649
01657     bool IsNull() const { return val.IsNull(); }
01658
01666     bool NotNull() const { return !(val.IsNull()); }
01667
01671     void SetNull()
01672     {
01673         val.SetNull();
01674         pend_val = val;
01675     }
01676
01682     config_IPADDR &operator=(const IPADDR &i6)
01683     {
01684         pend_val = val = i6;
01685         return *this;
01686     };
01687
01693     config_IPADDR &operator=(const config_IPADDR &ci)
01694     {
01695         pend_val = val = ci.val;
01696         return *this;
01697     };
01698
01699     // This is named so error messages make sense to user, old name was GetExtent
01700     virtual int Missing_ConfigEndMarker(void *&startp)
01701     {
01702         startp = this;
01703         return sizeof(*this);
01704     };
01705
01713     virtual void GetTypeValue(NBString &s) { s = "string"; };
01714
01715     // Add web page help
01716     virtual void ExtendedSchema(int fd, int indent, bool pretty) { DoSchemaLine(fd, "format", "ipv6",
      indent, pretty); };
01717 };
01718 #endif
01719
01727 class config_MACADR : public config_value
01728 {
01729     MACADR val;
01730     MACADR pend_val;
01731
01732   public:
01741     virtual void GetTextValue(NBString &s)
01742     {
01743         s.siprintf("\"%02X:%02X:%02X:%02X:%02X:%02X\"", val.phywadr[0] » 8, val.phywadr[0] & 0xFF,
      val.phywadr[1] » 8,
01744                    val.phywadr[1] & 0xFF, val.phywadr[2] » 8, val.phywadr[2] & 0xFF);
01745     };
01746
01757     config_MACADR(config_obj &owner, MACADR def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01758     {
01759         val = def_val;
01760         pend_val = val;
01761     }
```

```
01762
01763     config_MACADR(config_MACADR &&m);
01764
01774     config_MACADR(MACADR def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01775     {
01776         val = def_val;
01777         pend_val = val;
01778     }
01779
01791     config_MACADR(config_obj &owner, const char *def_val, const char *name, const char *desc = NULL) :
      config_value(owner, name, desc)
01792     {
01793         MACADR ma;
01794         ma = AsciiToMac(def_val);
01795
01796         val = ma;
01797         pend_val = val;
01798     }
01799
01810     config_MACADR(const char *def_val, const char *name, const char *desc = NULL) : config_value(name,
      desc)
01811     {
01812         MACADR ma;
01813         ma = AsciiToMac(def_val);
01814         val = ma;
01815         pend_val = val;
01816     }
01817
01818     // Assignment Operators
01819
01825     operator MACADR() const { return val; };
01826
01832     config_MACADR &operator=(const config_MACADR &ci)
01833     {
01834         pend_val = val = ci.val;
01835         return *this;
01836     };
01837
01843     config_MACADR &operator=(const MACADR &ci)
01844     {
01845         pend_val = val = ci;
01846         return *this;
01847     };
01848
01849     MACADR operator+(uint32_t rhs)
01850     { return val + rhs; }
01851
01852     // This is named so error messages make sense to user, old name was GetExtent
01853     virtual int Missing_ConfigEndMarker(void *&startp)
01854     {
01855         startp = this;
01856         return sizeof(*this);
01857     };
01858
01859     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
01860     virtual void CommitTestedValue(uint32_t permission_mask) { val = pend_val; }
01861
01869     virtual void GetTypeValue(NBString &s) { s = "string"; };
01870 };
01871
01872 // This is named so error messages make sense to user, old name was GetExtent
01873 #define ConfigEndMarker                                 \
01874     virtual int Missing_ConfigEndMarker(void *&startp) \
01875     {                                                   \
01876         startp = this;                                  \
01877         return sizeof(*this);                           \
01878     };
01879
01892 class config_chooser : public config_obj
01893 {
01894     config_string value{"", "Value"};
01895     config_string choices{"", "Choices"};
01896     ConfigEndMarker;
01897
01898     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
01899
01900   public:
01910     config_chooser(config_obj &owner, const char *name, const char *in_value, const char *in_choices,
      const char *desc = NULL)
01911         : config_obj(owner, name, desc)
01912     {
01913         value = in_value;                                   // Current choice
01914         choices = in_choices;                               // List of choices
01915         choices.m_Flags |= fConfigReadOnly | fConfigNoSave;   // Confiuration flags
01916         value.SetEnumList(choices);                         // Create enum list from string
      containing all choices
```

```
01917     };
01918
01919     config_chooser(config_chooser &&c);
01920
01929     config_chooser(const char *name, const char *in_value, const char *in_choices, const char *desc =
      NULL) : config_obj(name, desc)
01930     {
01931         value = in_value;
01932         choices = in_choices;
01933         choices.m_Flags |= fConfigReadOnly | fConfigNoSave;
01934         value.SetEnumList(choices);
01935     };
01936
01945     bool IsSelected(const char *choice) { return (choice == value); };
01946
01955     bool IsSelected(const NBString &s) { return (s == value); };
01956
01970     bool IsInChoices(const char *str, size_t strLen)
01971     {
01972         const char *sChoices = choices.c_str();
01973         size_t listLen = choices.length();
01974         size_t index = 0;
01975
01976         if (str == nullptr) { return false; }
01977         if (listLen == 0) { return false; }
01978
01979         while (index < listLen)
01980         {
01981             size_t curVarStart = index;
01982             size_t curVarLength;
01983
01984             // determine the length of the current element in the list of choices
01985             while (sChoices[index] != 0 && sChoices[index] != ',')
01986             {
01987                 // index until the end of the current element in the list of choices
01988                 index++;
01989             }
01990             curVarLength = index - curVarStart;
01991
01992             if (strncmp(str, &sChoices[curVarStart], (curVarLength > strLen) ? curVarLength : strLen)
      == 0)
01993             {
01994                 return true;   // found a match
01995             }
01996             index++;   // increment past ','
01997         }
01998
01999         return false;
02000     }
02001
02015     bool IsInChoices(const NBString &str, size_t strLen)
02016     {
02017         const char *sStr = str.c_str();
02018         const char *sChoices = choices.c_str();
02019         size_t length = choices.length();
02020         size_t index = 0;
02021
02022         if (sStr == nullptr) { return false; }
02023         if (length == 0) { return false; }
02024
02025         while (index < length)
02026         {
02027             size_t curVarStart = index;
02028             size_t curVarLength;
02029
02030             // determine the length of the current element in the list of choices
02031             while (sChoices[index] != 0 && sChoices[index] != ',')
02032             {
02033                 // index until the end of the current element in the list of choices
02034                 index++;
02035             }
02036             curVarLength = index - curVarStart;
02037
02038             if (strncmp(sStr, &sChoices[curVarStart], curVarLength > strLen ? curVarLength : strLen)
      == 0)
02039             {
02040                 return true;   // found a match
02041             }
02042             index++;   // increment past ','
02043         }
02044
02045         return false;
02046     }
02047
02053     const config_string &GetChoices() { return choices; }
02054
02062     const config_string &SetChoices(const char *in_choices)
```

```
02063     {
02064         choices = in_choices;
02065         value.SetEnumList(choices);
02066 
02067         return choices;
02068     }
02069 
02070      /* @brief Set the list of choices
02071      *
02072      * @param in_choices The list of option choices
02073      *
02074      * @returns A config_string object containing the list of choices
02075      */
02076     const config_string &SetChoices(const NBString &in_choices)
02077     {
02078         choices = in_choices;
02079         value.SetEnumList(choices);
02080 
02081         return choices;
02082     }
02083 
02084 
02090     operator NBString() const { return (NBString)value; };
02091 
02097     config_chooser &operator=(const char *p)
02098     {
02099         value = p;
02100         return *this;
02101     };
02102 
02108     config_chooser &operator=(const NBString &s)
02109     {
02110         value = s;
02111         return *this;
02112     };
02113 
02119     config_chooser &operator=(const config_chooser &ci)
02120     {
02121         value = ci.value;
02122         return *this;
02123     };
02124 
02125     virtual void GetTypeValue(NBString &s) { s = "object"; };
02126 };
02127 
02128 class reboot_obj : public config_bool
02129 {
02130     virtual const char *GetSortNameValue() { return "ZZZc"; };
02131 
02132    public:
02133     reboot_obj() : config_bool(root, false, "Reboot", "Cause system reboot on save"){};
02134     void clear()
02135     {
02136         val = false;
02137         pend_val = false;
02138     };
02139 
02140     // This is named so error messages make sense to user, old name was GetExtent
02141     virtual int Missing_ConfigEndMarker(void *&startp)
02142     {
02143         startp = this;
02144         return sizeof(*this);
02145     };
02146 };
02147 
02148 class version_obj : public config_int
02149 {
02150     bool bNeverSet;
02151     virtual const char *GetSortNameValue() { return "ZZZb"; };
02152 
02153    public:
02154     version_obj() : config_int(root, 0, "Version", "Version serial number") { bNeverSet = true; };
02155 
02156     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
02157     void inc()
02158     {
02159         val++;
02160         pend_val = val;
02161     };
02162     // This is named so error messages make sense to user
02163     // Old name was GetExtent
02164     virtual int Missing_ConfigEndMarker(void *&startp)
02165     {
02166         startp = this;
02167         return sizeof(*this);
02168     };
02169 };
```

```
02170
02171 class empty_config_obj : public config_obj
02172 {
02173     ConfigEndMarker;
02174
02175   public:
02176     empty_config_obj(const char *name, const char *desc = NULL) : config_obj(name, desc){};
02177     empty_config_obj(config_obj &owner, const char *name, const char *desc = NULL) : config_obj(owner,
      name, desc){};
02178     empty_config_obj(empty_config_obj &&e);
02179 };
02180
02181
02182
02183
02184
02185 // This class is intended for recovery applications or for devices supporting multiple
02186 //  application profiles to preserve unused config tree branches across configuration
02187 //  updates
02188 //  Ex:
02189 //      Application A has a config object at 'AppData.AppA', and Application B
02190 //      has a config object at 'AppData.AppB'. Normally, if Application A were
02191 //      to update any configuration (whether Application or System related), this
02192 //      would wipe out any AppB configuration as it is not known of by Application A.
02193 //      Using a config_preserver_obj, any configuration tree data below it's registration
02194 //      will be persisted across updates unless explicitly wiped.
02195 class config_preserver_obj : public config_obj
02196 {
02197     ConfigEndMarker;
02198     ParsedJsonDataSet &pendingTreeData;
02199     ParsedJsonDataSet &treeData;
02200
02201   public:
02202     config_preserver_obj(const char *name, const char *desc = NULL);
02203     config_preserver_obj(config_obj &owner, const char *name, const char *desc = NULL);
02204     config_preserver_obj(config_preserver_obj &&po);
02205
02206
02207     virtual ConfigTestResult TestNewValue(ParsedJsonDataSet &pjs);
02208     virtual void CommitTestedValue(uint32_t permission_mask);
02209     virtual void GetTextValue(NBString &s);
02210     virtual void GetRawValue(NBString &s);
02211 };
02212
02213 extern const char *AppName;
02214 extern const char PlatformName[];
02215
02216 class SysRecord : public config_obj
02217 {
02218   public:
02219     config_report system_platform{PlatformName, "Platform", "Hardware Platform"};
02220     config_report system_app{AppName, "Application", "Application name"};
02221     ConfigEndMarker;
02222
02223     SysRecord(const char *name, const char *desc = NULL) : config_obj(name, desc){};
02224     SysRecord(config_obj &owner, const char *name, const char *desc = NULL) : config_obj(owner, name,
      desc){};
02225     SysRecord(SysRecord &&sr);
02226 };
02227
02228 extern SysRecord sys;
02229 extern empty_config_obj netif;
02230
02231 extern reboot_obj rebooter;
02232 extern version_obj config_cur_version;
02233
02234 extern const int plat_def_baud;
02235 extern const int plat_def_delay;
02236 extern const int plat_def_quiet;
02237 extern const int plat_def_watchdog_enabled;
02238
02239 #include <plat_cfg_types.h>
02240 #ifdef PRESERVE_APP_DATA
02241 extern config_preserver_obj appdata;
02242 #else
02243 extern empty_config_obj appdata;
02244 #endif
02245
02250 void SaveConfigToStorage();
02251
02252 class MonitorRecord : public config_obj
02253 {
02254   public:
02255     config_int Baud{plat_def_baud, "BootBaud"};
02256     config_uart Uart{plat_def_com, "BootUart"};
02257     config_int BootDelay{plat_def_delay, "BootDelay"};
02258     config_bool Quiet{plat_def_quiet, "BootQuiet"};
```

```
02259     config_chooser sercfg_action{"SerialConfig", "DuringBoot",
      "DuringBoot,AlwaysEnabled,PauseAfterBoot,Disabled"};
02260     config_string abortbootcmd{"A", "Abort"};
02261     config_pass system_user{"", "User"};
02262     config_pass system_pass{"", "Password"};
02263     ConfigEndMarker;
02264
02265     MonitorRecord(const char *name) : config_obj(name, "Boot monitor record")
02266     {
02267         sercfg_action.SetFlag(fConfigNeedReboot);
02268         Baud.SetFlag(fConfigNeedReboot);
02269         Uart.SetFlag(fConfigNeedReboot);
02270     };
02271     MonitorRecord(config_obj &owner, const char *name) : config_obj(owner, name, "Boot Monitor
      Record")
02272     {
02273         sercfg_action.SetFlag(fConfigNeedReboot);
02274         Baud.SetFlag(fConfigNeedReboot);
02275         Uart.SetFlag(fConfigNeedReboot);
02276     };
02277     MonitorRecord(MonitorRecord &&mr);
02278 };
02279 extern MonitorRecord monitor_config;
02280 #endif
02281
```

24.304 config_server.h File Reference

Configuration Server.

`#include <servlets.h>`

Functions

- void EnableConfigMirror ()

  Enable the configuration mirror.
- void SaveConfigToStorage ()

  Write all pending data to flash memory.
- size_t ConfigSize ()

  Returns the number of bytes currently in use by configuration flash.
- size_t ConfigMaxSize ()

  Returns the number of bytes available in configuration flash.

24.304.1 Detailed Description

Configuration Server.

24.305 config_server.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00025 #ifndef __CONFIG_SERVER_H
00026 #define __CONFIG_SERVER_H
00027
00028 #include <servlets.h>
00029
00030 typedef void(diagserver)(int sock, const char *url);
00031 extern diagserver *pDiagServer;
00032
00033 /*
00034  * Macro used for styling the local discover page
00035  * .styleDiv    - Style for the DIVs around the table and button, responsible for element placement
00036  * .discDiv     - Style for DIV around the table, responsible for border and spacing from top of page
00037  * .discTab     - Style for table
00038  * .devTh       - Style for text separating devices in table
00039  * .devTd       - Style for all the table TD elements
00040  * .tabTh       - Style for column headers in table
00041  * .btnCnf      - Style for button INPUT element
```

```
00042  */
00043 #define DiscoverStyle
      \
00044     "<style>"
      \
00045     ".styleDiv{width:70%; min-width:480px; max-width:900px; font-family:sans-serif; margin-left:auto;
      margin-right:auto;}" \
00046     ".discDiv{border-width:1px; border-style:solid; border-color:#595454; border-radius:5px;
      margin-top:30px;}"            \
00047     ".discTab{width:100%; color:#595454; border-spacing:0px;}"
      \
00048     ".devTh{font-weight:bold; border-bottom:2px solid #ddd; padding:8px; border-top:1px solid #595454;
      color:#333;}"       \
00049     ".devTd{border-top:1px solid #ddd; padding:8px; color:#333;}"
      \
00050     ".tabTh{padding:8px; background-color:#595454; color:#fff; text-align:left;}"
      \
00051     ".btnCnf{width:125px; height:40px; background-color:#595454; color:#fff; border-radius:5px;
      margin-top:10px;}"          \
00052     "</style>"
00053
00054 void StartConfigServer(int prio);
00055 void ConfigInit();
00056 int FindValidConfig();   // returns fd for config blob reader
00057 void AddConfigServlet(servlet *s);
00058 void ResumeSerialConfig(int fd_cfg = CurrentStdioFD(1), bool persist = true);
00059
00071 void EnableConfigMirror();
00072
00073 // Weak reference returns true if pass is ok, passed string with USER:PASS
00074 bool ConfigAuthenticate(const char *up);
00075
00076 // Weak refernce to set up config servlet. Allows one to override
00077 void ConfigSetServlets();
00078
00079 // Configuration servlet stuff
00080 void AddConfigServlet(servlet *s);
00081
00082 // Exposed for expanding WebConfigServlet
00083 void ConfigIntFillInFD(fd_set &r_set, fd_set &w_set, fd_set &write_fds);
00084 void ConfigIntProcessSelectResult(fd_set &r_set, fd_set &w_set, fd_set &error_fds);
00085
00091 void SaveConfigToStorage();
00092
00093 class WebConfigServlet : public servlet
00094 {
00095     int m_cfg_listen;
00096     int m_port;
00097
00098   public:
00099     WebConfigServlet(int port = 20034);
00100     virtual int AddToSelectSet(fd_set &read_fds, fd_set &write_fds, fd_set &error_fds);
00101     virtual void ProcessSelectResult(fd_set &read_fds, fd_set &write_fds, fd_set &error_fds);
00102 };
00103
00109 size_t ConfigSize();
00110
00116 size_t ConfigMaxSize();
00117
00118 #endif /* ----- #ifndef __CONFIG_SERVER_H  ----- */
00119
```

24.306 config_time.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef CONFIG_TIMET_H
00006 #define CONFIG_TIMET_H
00007
00008 #include <config_obj.h>
00009 #include <sys/time.h>
00010 #include <sys/types.h>
00011 #include <time.h>
00012
00013
00014
00015 class config_time_t : public config_string
00016 {
00017 protected:
00018     time_t tv;
00019     bool buser_sees_gmt;
00020
00021 public:
```

```
00022 static NBString TimetoNBString(time_t tv,bool bUtc=true);
00023 static time_t ParseTime(const char * cp,bool bUtc=true);
00024 static time_t ParseTime(NBString & ns,bool bUtc=true) {return config_time_t::ParseTime(ns.c_str(),
      bUtc); }
00025
00037     config_time_t(config_obj &owner, time_t def_val, const char *name, const char *desc = NULL, bool
      bUserSeesGmt=false) : config_string(owner,config_time_t::TimetoNBString(def_val), name, desc)
00038     {
00039      tv=def_val;
00040      buser_sees_gmt=bUserSeesGmt;
00041     }
00042
00043
00044     config_time_t(config_obj &owner, const char * def_val, const char *name, const char *desc =
      NULL,bool bUserSeesGmt=false) : config_string(owner,def_val, name, desc)
00045     {
00046         tv=ParseTime(def_val);
00047         buser_sees_gmt=bUserSeesGmt;
00048     }
00049
00050     config_time_t(const char * def_val, const char *name, const char *desc = NULL,bool
      bUserSeesGmt=false) : config_string(def_val, name, desc)
00051     {
00052         tv=ParseTime(def_val);
00053         buser_sees_gmt=bUserSeesGmt;
00054     }
00055
00064     config_time_t(time_t def_val, const char *name, const char *desc = NULL) :
      config_string(config_time_t::TimetoNBString(def_val),name, desc)
00065     {
00066         tv=def_val;
00067     }
00068
00069 virtual void CommitTestedValue(uint32_t permission_mask)
      {config_string::CommitTestedValue(permission_mask); tv=config_time_t::ParseTime(val); };
00070
00071 virtual void GetTypeValue(NBString &s) { s = "time_t"; };
00072
00078     operator time_t() const { return tv; };
00079
00080
00081
00087     config_time_t &operator=(time_t t)
00088     {
00089         tv=t;
00090         pend_val = val = config_time_t::TimetoNBString(tv);
00091         return *this;
00092     };
00093
00099     config_time_t &operator=(const config_time_t &ci)
00100     {
00101         pend_val = val = ci.val;
00102         tv=ci.tv;
00103         return *this;
00104     };
00105
00106     NBString GetAsGmtString() {return val; }
00107
00108     NBString GetAsLocalString() {return TimetoNBString(tv,false); }
00109
00110
00111
00112
00113     // This is named so error messages make sense to user, old name was GetExtent
00114     virtual int Missing_ConfigEndMarker(void *&startp)
00115     {
00116         startp = this;
00117         return sizeof(*this);
00118     };
00119
00120
00121 void RenderValue(int fd, int len, const char *extra);
00122 void RenderInput(int fd, int len, const char *extra);
00123 bool ProcessValue(const char * pValue);
00124
00125 };
00126
00127 #endif
```

24.307 constants.h File Reference

NetBurner System Constants.

```
#include <predef.h>
```

24.307.1 Detailed Description

NetBurner System Constants.

24.308 constants.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00010 #include <predef.h>
00011
00012 #ifndef _CONSTANTS_H
00013 #define _CONSTANTS_H
00014
00015 #define TICK_IRQ_LEVEL (5) /* System clock IRQ level */
00016 #define SERIAL_IRQ_LEVEL (3)
00017 #define SERIAL_VECTOR_BASE (64)
00018
00019 /*
00020
00021 PLEASE READ THIS BEFORE MODIFYING TICKS_PER_SECOND
00022
00023 Before you change this value understand what changing it does.
00024 Making it faster only slows things down. It does not speed up task switches,
00025 If only changes the granularity of time delays and timeouts.
00026 Task switches happen much much faster than the tick interval. They happen as
00027 soon as one task blocks or an interrupt /or task causes a higher priority task
00028 to be unblocked. The Time tick has NOTHING to do with task switches.
00029
00030 If you change it to the maximum 200 you have increased the RTOS overhead by a factor
00031 of 10 and have actually slowed your system.
00032 */
00033
00034 #define TICKS_PER_SECOND (20) /* System clock tick */
00035
00036 /*
00037  ****************************************************************************
00038  * OS Maximum Interrupt Level
00039  *
00040  *  OS_MAX_IRQ_MASK       - Optional, Maximum IRQ level that the RTOS is
00041  *                          allowed to mask. RTOS objects may not be used
00042  *                          in ISRs above this level.
00043  ****************************************************************************
00044  */
00045 /* #define OS_MAX_IRQ       (3) */
00046
00047 /* Ethernet buffer defines */
00048 #define ETHER_BUFFER_SIZE 1548
00049 #define ETH_MAX_PAYLOAD (1500)
00050 #define ETH_MAX_SIZE (1522)
00051 #define ETH_MIN_SIZE (46)
00052 #define IP_HEADER_SIZE (20)
00053 #define UDP_HEADER_SIZE (8)
00054 #define MAX_UDPDATA (ETH_MAX_PAYLOAD - (IP_HEADER_SIZE + UDP_HEADER_SIZE))
00055
00056 #define SERIAL_TX_BUFFERS (2) /* ETHERN_BUFFER_SIZE = bytes of serial TX fifo */
00057 #define SERIAL_RX_BUFFERS (2) /* ETHERN_BUFFER_SIZE = bytes of serial RX fifo */
00058 #define stdin_buffer_size (200)
00059
00060 #define OS_MAX_TASKS 32 /* Max number of system tasks */
00061
00062 #define OS_MAX_PRIOS 64 /* Maximum number of system priorities */
00063
00064 /*
00065  ****************************************************************************
00066  *
00067  * System task priorities
00068  *
00069  *    Restating the NBRTOS RTOS Library document (NBRtosLibrary.pdf)
00070  *       Lowest priority is OS_MAX_PRIOS-1, 1 is the highest.
00071  *       There can only be one task at each priority level.
00072  *       Idle task is created at priority 63.
00073  *
00074  *    UserMain is created at priority 10. It is recommended and is supported by
00075  *       examples and default projects that a call to
00076  *          OSChangePrio( MAIN_PRIO );
00077  *       be made to lower the priority to the range recommended for the main
```

```
00078  *       application.
00079  *
00080  *    Factory applications use priorities 46 through 56.
00081  *
00082  *    Netburner Runtime library support and driver tasks use 36 through 45.
00083  *
00084  *    Care should be taken in use of priorities it can affect reliability and
00085  *       performance.
00086  *
00087  ******************************************************************************
00088  */
00089
00090 /* Recommend UserMain priority */
00091 #define MAIN_PRIO (50)
00092
00093 /* Runtime library driver and support task priorities */
00094 #define USB_HW_PRIO (45)
00095 #define PPP_PRIO (44)
00096 #define SECURITY_TASK_PRIO (43)
00097 #define WIFI_STATION_TASK_PRIO (42)
00098 #define WIFI_TASK_PRIO (41)
00099 #define CONFIG_SERVER_PRIO (40)
00100 #define HTTP_PRIO (39)
00101 #define ETHER_SEND_PRIO (38)
00102
00103 /* Features */
00104 /* SSH server must be lower than main for progress displays */
00105 #define SSH_TASK_PRIORITY (56)
00106
00107 /*
00108  ******************************************************************************
00109  *
00110  * Stack size definitions
00111  *
00112  *    SSH requires larger stacks for session key generation see predef.h
00113  *
00114  ******************************************************************************
00115  */
00116 #if defined NB_SSH_SUPPORTED || defined NB_SSL_SUPPORTED
00117 #define MAIN_TASK_STK_SIZE (3072)
00118 #define IP_STK_SIZE (2048)
00119 #define TCP_STK_SIZE (3072)
00120 #define HTTP_STK_SIZE (3072)
00121 #define IDLE_STK_SIZE (2048)
00122 #define ETHER_SEND_STK_SIZE (2048)
00123 #define PPP_STK_SIZE (2048)
00124 #define USB_HW_STK_SIZE (2048)
00125 #define USER_TASK_STK_SIZE (3072)
00126 #else /* #ifdef NB_SSH_SUPPORTED */
00127 #define MAIN_TASK_STK_SIZE (2048)
00128 #define IP_STK_SIZE (2048)
00129 #define TCP_STK_SIZE (2048)
00130 #define HTTP_STK_SIZE (2048)
00131 #define IDLE_STK_SIZE (2048)
00132 #define ETHER_SEND_STK_SIZE (2048)
00133 #define PPP_STK_SIZE (2048)
00134 #define USB_HW_STK_SIZE (2048)
00135 #define USER_TASK_STK_SIZE (2048)
00136 #endif /* #ifdef NB_SSH_SUPPORTED */
00137
00138 /* TCP definitions */
00139 #define DEFAULT_TCP4_MSS (512)
00140 #define DEFAULT_TCP6_MSS (1200)
00141 #define DEFAULT_TCP_RTTVAR \
00142     ((TICKS_PER_SECOND * 3) / 4)            /*See RFC 1122 for a 50msec tick 60 ticks=3 sec 4*15=60
      (The 4 comes from stevens Vol1-300) */
00143 #define TCP_CONN_TO (75 * TICKS_PER_SECOND) /* 75 seconds Min  */
00144 #define TCP_ACK_TICK_DLY (TICKS_PER_SECOND / 5) /* 200 msec delayed ACK timer */
00145 #define DEFAULT_INITAL_RTO (TICKS_PER_SECOND * 3)
00146 #define TCP_MAX_RTO (64 * TICKS_PER_SECOND)
00147 #define TCP_MIN_RTO (TICKS_PER_SECOND / 2)
00148 #define TCP_2MSL_WAIT (60 * TICKS_PER_SECOND)
00149 #define MAX_TCP_RETRY (12)
00150 #define TCP_WRITE_TIMEOUT (TICKS_PER_SECOND * 10)
00151 #define TCP_BUFFER_SEGMENTS (5) /* Store 4 segments max in tx and rx buffers  allows fast retransmit
      when packets lost*/
00152
00153 #define MAX_MULTICAST_GROUPS (32)
00154
00155 #define HTTP_TIMEOUT (TICKS_PER_SECOND * 10) /* 10 idle Seconds and a partially received request is
      abandoned */
00156 #define HTTP_READ_TIME_LIMIT (30)            /* Seconds to allow reading to avoid denial of service*/
00157 #define HTTP_RX_BUFFERSIZE (10000)
00158 #define MAX_HTTP_PENDING_SOCKETS (5)     // Number of sockets allowed to be pending on listening
      socket, performance will degrade < 3
00159 #define MAX_HTTP_CONNECTED_SOCKETS (5)   // Number of sockets allowed to be connected to http server
      simultaneously
```

```
00160 #define MAX_HTTP_POST_VAR_NAME_SIZE  (256)  // Max HTTP POST variable name length
00161 #define MAX_HTTP_POST_VAR_VALUE_SIZE (256)  // Max HTTP POST variable value length
00162
00163 // FDs are preallocated
00164 // STDOUT
00165 // STDIN
00166 // STDERR
00167 //           <-SERIAL_SOCKET_OFFSET
00168 // Serial 0
00169 // Serial 1
00170 //           <-TCP_SOCKET_OFFSET
00171 // TCP FD's
00172 //           <-EXTRA_IO_OFFSET
00173 // Extra FDS
00174
00175 #define SERIAL_SOCKET_OFFSET (3)
00176 #define TCP_SOCKET_OFFSET (5)
00177 #define TCP_SOCKET_STRUCTS (128)
00178 #define EXTRA_IO_OFFSET (TCP_SOCKET_OFFSET + TCP_SOCKET_STRUCTS)
00179 #define EXTRA_FDS (122)
00180 #define TOTAL_FDS (TCP_SOCKET_OFFSET + TCP_SOCKET_STRUCTS + EXTRA_FDS)
00181 #define FDSET_ELEMENTS ((TOTAL_FDS) / 32)
00182
00183 #define TASK_TABLE_SIZE 2
00184
00185 #define MAX_IP_ERRS 3
00186
00187 #define BUFFER_POOL_SIZE (256) /* was 64 in last release, we increased buffer segments to handle
      higher throughput events*/
00188 #define POOL_BUFFER_SIZE (sizeof(pool_buffer))   // Size of each buffer - 1712 bytes by default
00189 #define UDP_DISPATCH_SIZE (15)
00190 #define UDP_MIN_BUFFER_THRESHOLD (10)
00191 #define ARP_ENTRY_SIZE (256)
00192
00193 #define UDP_NETBURNERID_PORT (0x4E42) /* NB */
00194 #define UDP_DHCP_SERVER_PORT (67)
00195 #define UDP_DHCP_CLIENT_PORT (68)
00196 #define UDP_MDNS_PORT (5353)
00197
00198 #define TFTP_RX_PORT (1414)
00199
00200 #define LINK_STATUS_CHECK_INTERVAL (2 * TICKS_PER_SECOND)
00201
00202 #define FTPD_SOCKET_TIMEOUT (5 * 60 * TICKS_PER_SECOND)
00203
00204 #ifndef _DEBUG
00205 #define ENABLE_SRAM_SYS
00206 #endif
00207
00208 /* If ENABLE _SRAM_SYS is TRUE, then the processor's on-chip SRAM will be
00209     used for fast network buffering and OS tasks as defined below:
00210 */
00211 #ifdef ENABLE_SRAM_SYS
00212 #define FAST_SYSTEM_VARIABLES
00213
00214 // Uncommented system tasks will be stored in SRAM, otherwise SDRAM will be used.
00215 //#define FAST_IDLE_STACK
00216 #define FAST_MAIN_STACK
00217 #define FAST_ETHERNET_VARIABLES
00218 #define FAST_ETHERNET_STACK
00219 #define FAST_BUFFERS_VARIABLES
00220 #define FAST_BUFFERS
00221 #define FAST_IP_VARIABLES
00222 #define FAST_IP_STACK
00223 #define FAST_TCP_VARIABLES
00224 #define FAST_TCP_STACK
00225 #define FAST_USB_VARIABLES
00226 #define FAST_USB_STACK
00227 //#define FAST_HTTP_STACK
00228 //#define FAST_FTP_STACK
00229 //#define FAST_WIFI_STACK
00230 //#define FAST_PPP_STACK
00231 //#define FAST_COMMAND_STACK
00232
00233 /* If these defines are enabled, any user variables or tasks declared with
00234     FAST_USR_STK or FAST_USR_VAR will be stored in SRAM.
00235 */
00236 #define FAST_USER_VARIABLES
00237 #define FAST_USER_STACK
00238
00239 #define FAST_TLS_VARIABLES
00240
00241 #endif
00242
00243 #ifdef FAST_SYSTEM_VARIABLES
00244 #define FAST_SYS_VAR __attribute__((section("SYS_VAR_SECT")))
00245 #define FAST_SYS_VAR_REL __attribute__((section("SYS_VAR_SECT_REL")))
```

```
00246 #define FAST_SYS_VAR_REL_STR __attribute__((section("SYS_VAR_SECT_REL_STR")))
00247 #else
00248 #define FAST_SYS_VAR
00249 #define FAST_SYS_VAR_REL
00250 #define FAST_SYS_VAR_REL_STR
00251 #endif
00252 #ifdef FAST_IDLE_STACK
00253 #define FAST_IDLE_STK __attribute__((section("IDLE_STK_SECT")))
00254 #else
00255 #define FAST_IDLE_STK
00256 #endif
00257 #ifdef FAST_MAIN_STACK
00258 #define FAST_MAIN_STK __attribute__((section("MAIN_STK_SECT")))
00259 #else
00260 #define FAST_MAIN_STK
00261 #endif
00262 #ifdef FAST_USER_STACK
00263 #define FAST_USER_STK __attribute__((section("USER_STK_SECT")))
00264 #else
00265 #define FAST_USER_STK
00266 #endif
00267 #ifdef FAST_USER_VARIABLES
00268 #define FAST_USER_VAR __attribute__((section("USER_VAR_SECT")))
00269 #else
00270 #define FAST_USER_VAR
00271 #endif
00272 #ifdef FAST_ETHERNET_VARIABLES
00273 #define FAST_ETHER_VAR __attribute__((section("ETHER_VAR_SECT")))
00274 #define FAST_ETHER_VAR_REL __attribute__((section("ETHER_VAR_SECT_REL")))
00275 #else
00276 #define FAST_ETHER_VAR
00277 #define FAST_ETHER_VAR_REL
00278 #endif
00279 #ifdef FAST_ETHERNET_STACK
00280 #define FAST_ETHER_STK __attribute__((section("ETHER_STK_SECT")))
00281 #else
00282 #define FAST_ETHER_STK
00283 #endif
00284 #ifdef FAST_IP_VARIABLES
00285 #define FAST_IP_VAR __attribute__((section("IP_VAR_SECT")))
00286 #define FAST_IP_VAR_REL __attribute__((section("IP_VAR_SECT_REL")))
00287 #else
00288 #define FAST_IP_VAR
00289 #define FAST_IP_VAR_REL
00290 #endif
00291 #ifdef FAST_IP_STACK
00292 #define FAST_IP_STK __attribute__((section("IP_STK_SECT")))
00293 #else
00294 #define FAST_IP_STK
00295 #endif
00296 #ifdef FAST_TCP_VARIABLES
00297 #define FAST_TCP_VAR __attribute__((section("TCP_VAR_SECT")))
00298 #define FAST_TCP_VAR_REL __attribute__((section("TCP_VAR_SECT_REL")))
00299 #else
00300 #define FAST_TCP_VAR
00301 #define FAST_TCP_VAR_REL
00302 #endif
00303 #ifdef FAST_TCP_STACK
00304 #define FAST_TCP_STK __attribute__((section("TCP_STK_SECT")))
00305 #else
00306 #define FAST_TCP_STK
00307 #endif
00308 #ifdef FAST_HTTP_STACK
00309 #define FAST_HTTP_STK __attribute__((section("HTTP_STK_SECT")))
00310 #else
00311 #define FAST_HTTP_STK
00312 #endif
00313 #ifdef FAST_FTP_STACK
00314 #define FAST_FTP_STK __attribute__((section("FTP_STK_SECT")))
00315 #else
00316 #define FAST_FTP_STK
00317 #endif
00318 #ifdef FAST_WIFI_STACK
00319 #define FAST_WIFI_STK __attribute__((section("WIFI_STK_SECT")))
00320 #else
00321 #define FAST_WIFI_STK
00322 #endif
00323 #ifdef FAST_PPP_STACK
00324 #define FAST_PPP_STK __attribute__((section("PPP_STK_SECT")))
00325 #else
00326 #define FAST_PPP_STK
00327 #endif
00328 #ifdef FAST_COMMAND_STACK
00329 #define FAST_COMMAND_STK __attribute__((section("COMMAND_STK_SECT")))
00330 #else
00331 #define FAST_COMMAND_STK
00332 #endif
```

```
00333 #ifdef FAST_BUFFERS_VARIABLES
00334 #define FAST_BUFF_VAR __attribute__((section("BUFFERS_VAR_SECT")))
00335 #define FAST_BUFF_VAR_REL __attribute__((section("BUFFERS_VAR_SECT_REL")))
00336 #else
00337 #define FAST_BUFF_VAR
00338 #define FAST_BUFF_VAR_REL
00339 #endif
00340 #ifdef FAST_TLS_VARIABLES
00341 #define FAST_TLS_VAR __attribute__((section("TLS_VAR_SECT")))
00342 #define FAST_TLS_VAR_REL __attribute__((section("TLS_VAR_SECT_REL")))
00343 #else
00344 #define FAST_TLS_VAR
00345 #define FAST_TLS_VAR_REL
00346 #endif
00347 #ifdef FAST_USB_VARIABLES
00348 #define FAST_USB_VAR __attribute__((section("USB_VAR_SECT")))
00349 #define FAST_USB_VAR_REL __attribute__((section("USB_VAR_SECT_REL")))
00350 #else
00351 #define FAST_USB_VAR
00352 #define FAST_USB_VAR_REL
00353 #endif
00354 #ifdef FAST_USB_STACK
00355 #define FAST_USB_STK __attribute__((section("USB_STK_SECT")))
00356 #else
00357 #define FAST_USB_STK
00358 #endif
00359
00360 #define DO_NOT_CACHE __attribute__((section("NO_CACHE_SECT")))
00361
00362 #define FIRST_UNUSED_TIMER (-1)
00363
00364 /* Config server stuff */
00365 #define NumberOfConfigWebChannels 12
00366 #define ConfigActionIdleTimeout (TICKS_PER_SECOND * 30)
00367 #define MAX_HDR_SIZE (256) /* Size of web client recieve single header maximum*/
00368
00369
00370 #define NUM_DNS_CACHE (16)
00371
00372 #endif /* #ifndef _CONSTANTS_H */
```

24.309 convert.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _CONVERT_H_
00006 #define _CONVERT_H_
00007 #include <basictypes.h>
00008
00009 /*
00010  ******************************************************************************
00011  *
00012  * Definitions
00013  *
00014  ******************************************************************************
00015  */
00016 /* Minimum data buffer length in bytes */
00017 #define CONVERT_BUFFER_LENGTH_MAX ((4 * 1024) - 1)
00018
00019 /* Converted PEM data type */
00020 #define CONVERT_NONE (0)
00021 #define CONVERT_RSA_PRIVATE_KEY (1)
00022 #define CONVERT_DSA_PRIVATE_KEY (2)
00023 #define CONVERT_CERTIFICATE (3)
00024 #define CONVERT_ECDSA_PRIVATE_KEY (4)
00025
00026 /* PEM data headers and footers */
00027 #define CONVERT_CERTIFICATE_HEADER "-----BEGIN CERTIFICATE-----"
00028 #define CONVERT_RSA_PRIVATE_KEY_HEADER "-----BEGIN RSA PRIVATE KEY-----"
00029 #define CONVERT_DSA_PRIVATE_KEY_HEADER "-----BEGIN DSA PRIVATE KEY-----"
00030 #define CONVERT_ECDSA_PRIVATE_KEY_HEADER "-----BEGIN EC PRIVATE KEY-----"
00031 #define CONVERT_CERTIFICATE_FOOTER "-----END CERTIFICATE-----"
00032 #define CONVERT_RSA_PRIVATE_KEY_FOOTER "-----END RSA PRIVATE KEY-----"
00033 #define CONVERT_DSA_PRIVATE_KEY_FOOTER "-----END DSA PRIVATE KEY-----"
00034 #define CONVERT_ECDSA_PRIVATE_KEY_FOOTER "-----END EC PRIVATE KEY-----"
00035
00036 /*
00037  ******************************************************************************
00038
00039    Convert data from Privacy-enhanced Electronic Mail (PEM) format to binary.
00040
00041    Parameters:
00042       dataPtr                - Buffer for converted data
```

```
00043       pemEncodedData          - PEM encoded data
00044       dataSize                - Buffer size in bytes
00045       convertedDataLength     - Set to length of converted data
00046
00047    Return:
00048       CONVERT_NONE            - Malformed data or calling parameters
00049       CONVERT_CERTIFICATE
00050       CONVERT_RSA_PRIVATE_KEY
00051       CONVERT_DSA_PRIVATE_KEY
00052
00053    Notes:
00054       dataSize should be at least CONVERT_BUFFER_LENGTH_MAX and NULL
00055            terminated.
00056       Limited to file formats used for RSA & DSA keys and X.509 certificates.
00057
00058  ***************************************************************************
00059  */
00060 int ConvertPEMFormattedData(puint8_t dataPtr,
00061                             const char *pemEncodedData,
00062                             uint32_t dataSize,
00063                             puint32_t convertedDataLength,
00064                             char **nextPtr = NULL);
00065
00066 /*
00067  ***************************************************************************
00068
00069    Convert multibyte host format data to/from little endian
00070
00071    Parameters:
00072       data                    - Multibyte data to convert
00073
00074    Return:
00075       Data in little-endian format
00076
00077    Notes:
00078       none
00079
00080  ***************************************************************************
00081  */
00082 uint16_t convertLittleEndianWord(uint16_t hData);
00083 uint32_t convertLittleEndianDword(uint32_t hData);
00084
00085 #endif /* #ifdef _CONVERT_H_ */
```

24.310 counters.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <predef.h>
00006 #include <basictypes.h>
00007 /* The data counters kept by the TCP/IP system */
00008 #ifndef _NB_COUNTER_H
00009 #define _NB_COUNTER_H
00010 extern volatile uint32_t frames_tx;           //   "tx:%u","tx:%d",
00011 extern volatile uint32_t frames_rx;           //   "rx:%u","rx:%d",
00012 extern volatile uint32_t frames_tx_err;       //   "tx_err:%u","tx_err:%d",
00013 extern volatile uint32_t frames_rx_err;       //   "rx_err:%u","rx_err:%d",
00014 extern volatile uint32_t frames_rx_discard;   //   "rx_err:%u","rx_err:%d",
00015 extern volatile uint32_t frames_rx_arp;       //   "rx_arp:%u","rx_arp:%d",
00016 extern volatile uint32_t frames_tx_arp;       //   "tx_arp:%u","tx_arp:%d",
00017 extern volatile uint32_t frames_tx_udp;       //   "tx_udp:%u","tx_udp:%d",
00018 extern volatile uint32_t frames_rx_udp;       //   "rx_udp:%u","rx_udp:%d",
00019 extern volatile uint32_t frames_tx_tcp;       //   "tx_tcp:%u","tx_tcp:%d",
00020 extern volatile uint32_t frames_rx_tcp;       //   "rx_tcp:%u","rx_tcp:%d",
00021 extern volatile uint32_t frames_tx_icmp;      //   "tx_icmp:%u","tx_icmp:%d",
00022 extern volatile uint32_t frames_rx_icmp;      //   "rx_icmp:%u","rx_icmp:%d",
00023 extern volatile uint32_t frames_ip_errors;    //   "ip_err:%u","ip_err:%d",
00024 extern volatile uint32_t frames_ip_discard;   //   "ip_discard:%u","ip_discard:%d",
00025 extern volatile uint32_t frames_udp_errors;   //  "udp_err:%u","udp_err:%d",
00026 extern volatile uint32_t frames_tcp_errors;   //  "tcp_err:%u","tcp_err:%d",
00027 extern volatile uint32_t enet_last_errhw;     //   "last_err:%u""last_err:%d"
00028 extern volatile uint32_t enet_last_errlw;     //   "last_err:%u""last_err:%d"
00029 extern volatile uint32_t ip_last_err;         //   "last_err:%u""last_err:%d"
00030 extern volatile uint32_t enet_isr;            // "enet_isr:%u""enet_isr:%d"
00031 extern volatile uint32_t frames_rx_unknown;   //"frames_rx_unknown:%u""frames_rx_unknown%d"
00032 extern volatile uint32_t frames_rx_ppp_errors;
00033 extern volatile uint32_t frames_rx_fragment;
00034
00035 #ifdef FEC_ISR_ERROR_COUNTERS
00036 extern uint32_t TxIsrError;
00037 extern uint32_t TxIsrLateCollisions;
00038 extern uint32_t TxIsrRetryLimit;
00039 extern uint32_t TxIsrUnderrun;
```

```
00040 #endif   // FEC_ISR_ERROR_COUNTERS
00041
00042 #endif
```

24.311 dbgmon.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DBG_MON_H
00006 #define _DBG_MON_H
00007 #ifdef __cplusplus
00008
00009 int OpenDBGSerial(int portnum, unsigned int baudrate);
00010
00011 #endif
00012 #endif
```

24.312 debugalloc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DEBUG_ALLOC_H_
00006 #define _DEBUG_ALLOC_H_
00007
00008 #include <stddef.h>
00009 /*
00010  ****************************************************************************
00011  *
00012  * Definitions
00013  *
00014  ****************************************************************************
00015  */
00016 /*
00017  * Memory Allocation Debugging
00018  *    Should be defined if needed, unnecessary burden to release code
00019  *
00020  */
00021 /* #define NB_DEBUG_ALLOC_SUPPORTED   ( 1 ) */
00022
00023 /*
00024  * Memory Allocation Features
00025  *
00026  */
00027 /* All activities are displayed with iprintf */
00028 /* #define NB_DEBUG_ALLOC_VERBOSE         ( 1 ) */
00029
00030 /* Log all activities as well */
00031 /* #define NB_DEBUG_ALLOC_LOG_ALL         ( 1 ) */
00032
00033 /*
00034  * Entry log size
00035  *    Size of allocLogEntry per entry
00036  *    Additional two for ...LOG_ALL
00037  *    Overflow is reported but the process continues
00038  */
00039 #define NB_DEBUG_ALLOC_LOG_SIZE (2048)
00040
00041 /* Guardian size in bytes before and after returned allocated memory */
00042 #define NB_DEBUG_ALLOC_GUARD_SIZE (64)
00043
00044 /* Guardian value filled and checked */
00045 #define NB_DEBUG_ALLOC_GUARD_VALUE (0xA5)
00046
00047 #ifdef __cplusplus
00048 extern "C"
00049 {
00050 #endif
00051     /*
00052      ****************************************************************************
00053      *
00054      * Debug malloc, calloc, realloc and free "C" Library Interface
00055      *
00056      ****************************************************************************
00057      */
00058
00059     /*
00060      ****************************************************************************
00061
```

```
00062        Debug logger and optionally guardians:
00063           malloc
00064           calloc
00065           realloc
00066           free
00067
00068        Parameters:
00069           byteCount            - Memory needed in bytes
00070           elementCount         - Elements of byteCount bytes (calloc)
00071           ptr                  - Previously allocated memory
00072           caller               - Function calling, best choice __FUNCTION__
00073           line                 - Line number of call, best choice __LINE__
00074
00075        Return:
00076           > 0 Address of allocated memory, 0 problems
00077
00078        Notes:
00079           calloc zeros memory allocated
00080           realloc can extend or truncate memory, contents are the same as ptr
00081           realloc if problems does not change or deallocate memory
00082           Will not free a pointer not logged
00083
00084      ***************************************************************************
00085      */
00086     void *mallocDebug(size_t byteCount, const char *caller, int line);
00087     void *callocDebug(size_t elementCount, size_t byteCount, const char *caller, int line);
00088     void *reallocDebug(void *ptr, size_t byteCount, const char *caller, int line);
00089     void freeDebug(void *ptr, const char *caller, int line);
00090
00091     /*
00092      ***************************************************************************
00093
00094        Display using iprintf log and all allocated log
00095
00096        Parameters:
00097           None
00098
00099        Return:
00100           None
00101
00102        Notes:
00103           None
00104
00105      ***************************************************************************
00106      */
00107     void printAllocDebugLog(void);
00108     void printAllocDebugLogAll(void);
00109
00110 /*
00111  ***************************************************************************
00112  *
00113  * Definitions
00114  *
00115  ***************************************************************************
00116  */
00117 #ifdef NB_DEBUG_ALLOC_SUPPORTED
00118 #define NBMALLOC(bYtEcOuNt) mallocDebug(bYtEcOuNt, __FUNCTION__, __LINE__);
00119 #define NBCALLOC(eLeMeNtCoUnT, bYtEcOuNt) callocDebug(eLeMeNtCoUnT, bYtEcOuNt, __FUNCTION__,
    __LINE__);
00120 #define NBREALLOC(pTr, bYtEcOuNt) reallocDebug(pTr, bYtEcOuNt, __FUNCTION__, __LINE__);
00121 #define NBFREE(pTr) freeDebug(pTr, __FUNCTION__, __LINE__);
00122 #else /* #ifdef NB_DEBUG_ALLOC_SUPPORTED */
00123 #define NBMALLOC(bYtEcOuNt) malloc(bYtEcOuNt);
00124 #define NBCALLOC(eLeMeNtCoUnT, bYtEcOuNt) calloc(eLeMeNtCoUnT, bYtEcOuNt);
00125 #define NBREALLOC(pTr, bYtEcOuNt) realloc(pTr, bYtEcOuNt);
00126 #define NBFREE(pTr) free(pTr);
00127 #endif /* #ifdef NB_DEBUG_ALLOC_SUPPORTED */
00128
00129 #ifdef __cplusplus
00130 };
00131 #endif
00132 #endif /* #ifndef _DEBUG_ALLOC_H_ */
```

24.313 debugiprintf.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DEBUG_IPRINTF_H_
00006 #define _DEBUG_IPRINTF_H_
00007 #include <stdio.h>
00008 /*
00009  ***************************************************************************
00010  *
```

```
00011  * Definitions
00012  *
00013  *************************************************************************
00014  */
00015 /*
00016  * To include feature define NB_DEBUG_IPRINTF_CONTROL as uint32_t for choices
00017  *    Example:
00018  *       Declare once
00019  *          uint32_t WifiDebug = ( NB_DEBUG_IPRINTF_INIT | NB_DEBUG_IPRINTF_ERROR );
00020  *       Define before "include"ing this file
00021  *          #define NB_DEBUG_IPRINTF_CONTROL   WifiDebug
00022  *
00023  */
00024 /*
00025  * Debug iprintf choices
00026  *    User defined bits 9 through 27
00027  *
00028  */
00029 #define NB_DEBUG_OFF (0x00000000)
00030 #define NB_DEBUG_ERROR (0x00000001)
00031 #define NB_DEBUG_INIT (0x00000002)
00032 #define NB_DEBUG_INT (0x00000004)
00033 #define NB_DEBUG_RX (0x00000008)
00034 #define NB_DEBUG_TX (0x00000010)
00035 #define NB_DEBUG_USER_BEG (0x00010000)
00036 #define NB_DEBUG_USER_END (0x08000000)
00037 #define NB_DEBUG_TRACE (0x80000100)
00038 #define NB_DEBUG_ALL (0xFFFFFFFF)
00039
00040 /*
00041  *************************************************************************
00042  *
00043  * NB_DEBUG_IPRINTF
00044  *
00045  *************************************************************************
00046  */
00047 #ifdef NB_DEBUG_IPRINTF_CONTROL
00048
00049 #include <utils.h>
00050 extern uint32_t NB_DEBUG_IPRINTF_CONTROL;
00051 #define NB_DEBUG_IPRINTF(choice, ...)                                                                    \
00052     {                                                                                                    \
00053         if ((NB_DEBUG_IPRINTF_CONTROL & choice) == choice)                                               \
00054         {                                                                                                \
00055             (void)iprintf("NB Debug %s, line %d, at %ld\r\n", __FUNCTION__, __LINE__, TimeTick); \
00056             iprintf(__VA_ARGS__);                                                                        \
00057             iprintf("\r\n");                                                                             \
00058         }                                                                                                \
00059     }
00060
00061 #else /* #ifdef NB_DEBUG_IPRINTF_CONTROL */
00062
00063 #define NB_DEBUG_IPRINTF(choice, ...) \
00064     {                                 \
00065         (void)0;                      \
00066     }
00067
00068 #endif /* #ifdef NB_DEBUG_IPRINTF_CONTROL */
00069
00070 #endif /* #ifndef _DEBUG_IPRINTF_H_ */
```

24.314 debugprintblock.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DEBUG_PRINT_BLOCK_H_
00006 #define _DEBUG_PRINT_BLOCK_H_
00007 #include <stdio.h>
00008
00009 /*
00010  * Display of data in blocks
00011  *    Should be defined if needed, unnecessary burden to release code
00012  *
00013  */
00014 /*
00015  *************************************************************************
00016  *
00017  * Definitions
00018  *
00019  *************************************************************************
00020  */
00021
00022 #ifdef __cplusplus
```

```
00023 extern "C"
00024 {
00025 #endif
00026     /*
00027      ***************************************************************************
00028      *
00029      * Print block "C" Library Interface
00030      *
00031      ***************************************************************************
00032      */
00033
00034     /*
00035      ***************************************************************************
00036
00037        Print Block
00038           Print block of data in comma delimited format using iprintf
00039
00040        Parameters:
00041           typePtr              - Memory needed in bytes
00042           dataSize             - Bytes in each data item
00043           blockPtr             - Buffer with block of data
00044           byteCount            - Byte count
00045
00046        Return:
00047           None
00048
00049        Notes:
00050           None
00051      ***************************************************************************
00052      */
00053     void debugPrintBlock(const char *typePtr, size_t dataSize, unsigned char *blockPtr, unsigned int
      byteCount);
00054
00055 #ifdef __cplusplus
00056 };
00057 #endif
00058 #endif /* #ifndef _DEBUG_PRINT_BLOCK_H_ */
```

24.315 defer.h

```
00001 #ifndef __DEFER_H
00002 #define __DEFER_H
00003 /* As posted by Oded Lazar, modified from StackOverflow user R. Martinho Fernandes, */
00004 /*  and originally published by Andrei Alexandrescu and Petru Marginean in Dr. Dobbs
00005  *  December 2000 */
00006 /* https://oded.blog/2017/10/05/go-defer-in-cpp/ */
00007 /* https://stackoverflow.com/questions/10270328/the-simplest-and-neatest-c11-scopeguard/ */
00008 /* http://www.drdobbs.com/cpp/generic-change-the-way-you-write-excepti/184403758 */
00009 /*NB_REVISION*/
00010
00011 /*NB_COPYRIGHT*/
00012
00013 #include <functional>
00014 #include <utility>
00015
00016 #define DEFER_CONCAT_(a, b) a##b
00017 #define DEFER_CONCAT(a, b) DEFER_CONCAT_(a, b)
00018
00019 #define DEFER_CALL(fn) _ScopeGuard DEFER_CONCAT(__defer__, __LINE__) = [&]() { fn; };
00020
00021 class _ScopeGuard
00022 {
00023    public:
00024     template<class Callable>
00025     _ScopeGuard(Callable &&fn) : fn_(std::forward<Callable>(fn))
00026     {
00027     }
00028
00029     _ScopeGuard(_ScopeGuard &&other) : fn_(std::move(other.fn_)) { other.fn_ = nullptr; }
00030
00031     ~_ScopeGuard()
00032     {
00033         // must not throw
00034         if (fn_) fn_();
00035     }
00036
00037     _ScopeGuard(const _ScopeGuard &) = delete;
00038     void operator=(const _ScopeGuard &) = delete;
00039
00040    private:
00041     std::function<void()> fn_;
00042 };
00043
00044 #define IF_REENTERED(fn)                                                \
00045     static int DEFER_CONCAT(__reenter_guard_depth__, __LINE__) = 0; \
```

```
00046     _ReentrancyGuard DEFER_CONCAT(__reenter_guard__, __LINE__)(DEFER_CONCAT(__reenter_guard_depth__,
      __LINE__), [&]() { fn; });
00047
00048 class _ReentrancyGuard
00049 {
00050    public:
00051     template<class Callable>
00052     _ReentrancyGuard(int &entered, Callable &&fn) : depth(entered)
00053     {
00054         if (depth) { fn(); }
00055         depth++;
00056     }
00057
00058     _ReentrancyGuard(_ReentrancyGuard &&other) : depth(other.depth) {}
00059
00060     ~_ReentrancyGuard() { depth--; }
00061
00062     _ReentrancyGuard(const _ReentrancyGuard &) = delete;
00063     void operator=(const _ReentrancyGuard &) = delete;
00064
00065    private:
00066     int &depth;
00067 };
00068
00069 #endif /* ----- #ifndef __DEFER_H  ----- */
```

24.316 device.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _DEVICE_H
00006 #define _DEVICE_H
00007 #include <nbrtos.h>
00008 /*
00009  ********************************************************************************
00010  *
00011  * Constants
00012  *
00013  ********************************************************************************
00014  */
00015 /* Fixed interrupt sources */
00016 #define DEVICE_INTERRUPT_MAXIMUM (8)
00017
00018 /*
00019  ********************************************************************************
00020  *
00021  * Enumerations
00022  *
00023  ********************************************************************************
00024  */
00025 /*
00026    DeviceSetupIsr interrupt trigger
00027
00028       InterruptTriggerNone              - No interrupt level
00029       InterruptTriggerSensitive         - Trigger-sensitive interrupt
00030       InterruptTriggerRisingEdge        - Rising edge triggered interrupt
00031       InterruptTriggerFallingEdge       - Falling edge triggered interrupt
00032       InterruptTriggerBothEdges         - Both rising and falling edge triggered
00033
00034 */
00035 typedef enum _DeviceInterruptTrigger
00036 {
00037     InterruptTriggerNone,
00038     InterruptTriggerSensitive,
00039     InterruptTriggerRisingEdge,
00040     InterruptTriggerFallingEdge,
00041     InterruptTriggerBothEdge
00042
00043 } DeviceInterruptTrigger;
00044
00045 /*
00046  ********************************************************************************
00047  *
00048  * C++ definitions
00049  *
00050  ********************************************************************************
00051  */
00052
00053 /*
00054  ********************************************************************************
00055  *
00056  * Routine Prototypes
00057  *
```

```
00058  ********************************************************************************
00059  */
00060 /* ISR function */
00061 typedef void(DeviceIsrFn)();
00062 typedef DeviceIsrFn *DeviceIsrFnPtr;
00063
00064 /*
00065  ********************************************************************************
00066
00067    Verify availability and support for device.
00068
00069
00070       Parameters:
00071          irq         - Fixed level interrupt source
00072
00073
00074       Return:
00075          TRUE - Support and available on module, FALSE option not supported
00076
00077        Note:
00078          Only verifies irq is valid
00079
00080  ********************************************************************************
00081  */
00082 BOOL DeviceIsValid(int irq);
00083
00084 /*
00085  ********************************************************************************-
00086
00087    Set interrupt controller and attach function to interrupt
00088
00089       Parameters:
00090          irq               - Fixed level interrupt source
00091          trigger           - Triggering edge level
00092          isrPtr            - ISR function
00093
00094       Return:
00095          TRUE - Set, FALSE - Not valid interrupt source
00096
00097       Notes:
00098          None
00099
00100  ********************************************************************************-
00101  */
00102 BOOL DeviceSetupIsr(int irq, DeviceInterruptTrigger trigger, DeviceIsrFnPtr isrPtr);
00103
00104 /*
00105  ********************************************************************************-
00106
00107    Acknowledge interrupt
00108
00109       Parameters:
00110          irq               - Fixed level interrupt source
00111
00112       Return:
00113          None
00114
00115       Notes:
00116          None
00117
00118  ********************************************************************************-
00119  */
00120 void DeviceAckInterrupt(int irq);
00121
00122 /*
00123  ********************************************************************************-
00124
00125    Disable interrupt
00126
00127       Parameters:
00128          irq               - Fixed level interrupt source
00129
00130       Return:
00131          None
00132
00133       Notes:
00134          None
00135
00136  ********************************************************************************-
00137  */
00138 void DeviceDisableInterrupt(int irq);
00139
00140 /*
00141  ********************************************************************************-
00142
00143    Enable interrupt
00144
```

```
00145       Parameters:
00146          irq               - Fixed level interrupt source
00147
00148       Return:
00149          None
00150
00151       Notes:
00152          None
00153
00154  ***************************************************************************-
00155  */
00156 void DeviceEnableInterrupt(int irq);
00157
00158 /*
00159  ***************************************************************************
00160  *
00161  * Runtime Libraries Routine Prototypes
00162  *
00163  ***************************************************************************
00164  */
00165
00166 /*
00167  ***************************************************************************
00168
00169    Acquire QSPI semaphore for sharing access to SPI devices.
00170
00171    Parameters:
00172       None
00173
00174    Return:
00175       Semaphore, NULL for non-existent device
00176
00177    Notes:
00178       None
00179
00180  ***************************************************************************
00181  */
00182 OS_SEM *DeviceGetQspiSem(void);
00183
00184 /*
00185  ***************************************************************************
00186
00187    Acquire I2C semaphore for sharing access to I2C devices.
00188
00189    Parameters:
00190       None
00191
00192    Return:
00193       Semaphore, NULL for non-existent device
00194
00195    Notes:
00196       None
00197
00198  ***************************************************************************
00199  */
00200 OS_SEM *DeviceGetI2cSem(void);
00201
00202 #endif /* _DEVICE_H */
```

24.317 dhcpclient.h File Reference

NetBurner IPv4 DHCP Client Header File.

```
#include <buffers.h>
#include <nbrtos.h>
#include <nettimer.h>
#include <nettypes.h>
```

Classes

- class DhcpObject

  DHCP client class.

Macros

- #define **SDHCP_NOTSTARTED** 0

The System has not been initialized.

- #define **SDHCP_DISCOVER** 1

  The system is discovering the DHCP servers.
- #define **SDHCP_OFFER** 2

  The system has responded to an OFFER.
- #define **SDHCP_ACK** 3

  The System has Acknowledged the OFFER.
- #define **SDHCP_INIT** 4

  The System is reinitalzing.
- #define **SDHCP_CMPL** 5

  The System has obtained a valid DHCP lease.
- #define **SDHCP_RENEW** 6

  The System is in the process of renewing.
- #define **SDHCP_REBIND** 7

  The System has failed the Renew and is trying to Rebind.
- #define **SDHCP_RELEASE** 8

  The System is trying to release the Lease.
- #define **SBOOTP_TRANSMITTING** 9

  Trying BOOTP.
- #define **SBOOTP_DONE** 10

  BOOTP complete.
- #define **SDHCP_FAILED** 11

  DHCP attempt failed - could not obtain a DHCP lease.

Functions

- int32_t GetIntefaceDHCPState (int interface=0)

  Returns current state of the DHCP lease, with optional interface parameter.
- int32_t WaitForDHCPInterface (int interface=0, uint16_t TicksToWait=10 ∗TICKS_PER_SECOND)

  Wait until a DHCP lease is obtained, or the timeout occurs.

24.317.1 Detailed Description

NetBurner IPv4 DHCP Client Header File.

24.318 dhcpclient.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00023 #ifndef _NB_DHCP_CLIENT_H
00024 #define _NB_DHCP_CLIENT_H
00025
00026 #include <buffers.h>
00027 #include <nbrtos.h>
00028 #include <nettimer.h>
00029 #include <nettypes.h>
00030
00031 // Functions necessary to use DHCP in the simplest form
00032 #define DHCP_OK (0)
00033 #define DHCP_NOTNEEDED (1)
00034 #define DHCP_FALLBACK (2)
00035 #define DHCP_FAILED (-1)
00036
00037 #define DHCP_LINK_DOWN_RESTART_TICKS (5 * TICKS_PER_SECOND)
00038
00042 //----- DHCP states -----
00043 #define SDHCP_NOTSTARTED 0
00044 #define SDHCP_DISCOVER 1
00045 #define SDHCP_OFFER 2
```

```
00046 #define SDHCP_ACK 3
00047 #define SDHCP_INIT 4
00048 #define SDHCP_CMPL 5
00049 #define SDHCP_RENEW 6
00050 #define SDHCP_REBIND 7
00051 #define SDHCP_RELEASE 8
00052 #define SBOOTP_TRANSMITTING 9
00053 #define SBOOTP_DONE 10
00054 #define SDHCP_FAILED 11
00057 class InterfaceBlock;   // Forward declaration
00058 class DHCPMessage;      // Forward declaration
00059
00065 class DhcpObject : public TimeOutElement
00066 {
00067    private:
00068     int DhcpState;          // Current state of DHCP engine
00069     int DhcpRetryCount;     // Number of transmit retries.
00070     int DhcpTimeCounter;    // Number of seconds til retry
00071     uint32_t DHCPLastXid;   // Last XID
00072     int DhcpBackoffTimer;
00073     volatile int DhcpPendingPacket;
00074     uint32_t DhcpLeasePendingStart;   // IP address lease start time in seconds
00075     uint32_t DhcpLeaseDuration;       // Relative duration of lease in seconds
00076
00077     void CreateDhcpRenewMsg(DHCPMessage &NewMsg);
00078     void CreateDhcpReleaseMsg(DHCPMessage &NewMsg);
00079     BOOLEAN BootpConfig(DHCPMessage &Msg);
00080     BOOLEAN DhcpConfig(DHCPMessage &Msg);
00081     void UpdateIPRuntimeVars(BOOLEAN release);
00082     void ReTransmitPacket(uint32_t xid = 0);
00083     void CreateBootpMsg(DHCPMessage &NewMsg);
00084     uint32_t GetNextEventIntervalSecs();
00085
00086     BOOL bProcessThisDhcp;
00087     DhcpObject *m_next_in_list;
00088     uint32_t m_LinkChangeTick;
00089
00090     virtual void TimeElementEvent();
00091
00092    public:
00093     IPADDR4 DhcpClientIP;      // Allocated IP address
00094     IPADDR4 DhcpClientMask;    // Allocated subnet mask
00095     IPADDR4 DhcpServerIP;      // Server ID
00096     IPADDR4 DhcpRelayIP;       // Relay Agent
00097     IPADDR4 DhcpRouterIP;      // Gateway IP
00098     IPADDR4 DhcpDNSIP;         // DNS IP
00099     IPADDR4 DhcpDNSIP2;        // 2nd DNS IP
00100     uint32_t DhcpLeaseStart;   // IP address lease start time in seconds
00101     uint32_t DhcpLeaseTime;    // IP address lease time in seconds
00102     uint32_t DhcpRenewTime;    // Time to Renewing state in seconds
00103     uint32_t DhcpRebindTime;   // Time to Rebinding state in seconds
00104     InterfaceBlock *pIfb;      // Interface to use
00105     OS_SEM NotifySem;
00106
00119     void StartDHCP();
00120
00121     // Start the BOOTP process
00122     void StartBOOTP();
00123
00124     // After BOOTP has failed call this to stop BOOTP processing
00125     void StopBOOTP();
00126
00136     void StopDHCP();
00137
00147     void RestartDHCP();
00148
00161     bool bDoFallBack();
00162
00172     void RenewDHCP();
00173
00183     void RebindDHCP();
00184
00193     BOOL ValidDhcpLease();
00194
00203     uint32_t GetRemainingDhcpLeaseTime();
00204
00213     uint32_t GetDhcpRenewTime() { return DhcpRenewTime; }
00214
00223     uint32_t GetDhcpRebindTime() { return DhcpRebindTime; }
00224
00233     uint32_t GetDhcpExpirationTime() { return DhcpLeaseTime; }
00234
00243     int32_t GetDHCPState();
00244
00245     //---- function prototypes -----
00246     void PrintDhcpState(int state);
00247
```

```
00248     void DHCPTimer();
00249     void DHCPPacket(PoolPtr p);
00250
00251     static void staticDHCPPacket(PoolPtr p);
00252
00253     void StopProcessing();
00254
00255     DhcpObject(InterfaceBlock *ib);
00256     ~DhcpObject();
00257
00258     void (*leaseExpiredCallback)(DhcpObject *dhcpClient);
00259     void (*DHCPFailedCallback)(DhcpObject *dhcpClient);
00260     void (*leaseObtainedCallback)(DhcpObject *dhcpClient);
00261
00262     void LinkNotify(bool link);
00263
00264     friend void PrintDhcpStatus(DhcpObject &dob);
00265     friend void PrintDhcpState(int state);
00266
00267 };   // End of DHCP Object
00268
00279 int32_t GetIntefaceDHCPState(int interface = 0);
00280
00294 int32_t WaitForDHCPInterface(int interface = 0, uint16_t TicksToWait = 10 * TICKS_PER_SECOND);
00295
00296 #endif
```

24.319 dhcpd.h File Reference

NetBurner DHCP Server.

```
#include <predef.h>
#include <constants.h>
#include <buffers.h>
#include <nettypes.h>
```

Functions

- bool AddStandardDHCPServer (int intf=0, IPADDR4 startAddr=IPADDR4::NullIP())

  Starts a standard allocator DHCP server.

24.319.1 Detailed Description

NetBurner DHCP Server.
The DHCPD library provides a DHCP server daemon such that the Netburner module can provide leases to other devices on the same network. The standard reference implementation also implements a discovery mechanism which will only enable the server if no other server is present when initiated.

24.320 dhcpd.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00020 #ifndef __DHCPD_H
00021 #define __DHCPD_H
00022
00023 // NB Definitions
00024 #include <predef.h>
00025
00026 // NB Constants
00027 #include <constants.h>
00028
00029 // NB Libs
00030 #include <buffers.h>
00031 #include <nettypes.h>
00032
00033 #define DHCP_SERV_MAX_INTF (4)
00034 #define DHCP_OFFER_DURATION (2 * TICKS_PER_SECOND)
00035 #define DHCP_SERV_MAX_HOSTNAME_LEN (32)
00036
```

```
00037 #define LEASE_POOL_SIZE 150
00038 #define DHCPD_STARTING_ADDRESS 0xC0A80184
00039
00040 namespace DHCP
00041 {
00042 typedef enum LeaseState
00043 {
00044     LEASE_OPEN = 0x0,
00045     LEASE_OFFERED = 0x1,
00046     LEASE_TAKEN = 0x2,
00047     LEASE_STATIC = 0x3,
00048     ARP_CONFLICT = 0x4,
00049 } LeaseState_t;
00050
00051 struct DhcpLeaseRequest
00052 {
00053     IPADDR4 ip;                                     // the IP Address to be offered
00054     MACADR mac;                                     // source mac address of the request
00055     uint32_t duration;                              // The duration of the lease, in seconds
00056     uint32_t xid;                                   // transaction id of the DHCP message
00057     char hostname[DHCP_SERV_MAX_HOSTNAME_LEN + 1];  // buffer for client hostname
00058 };
00059
00060 struct DhcpLeaseData
00061 {
00062     IPADDR4 ip;                                     // the IP Address to be offered
00063     MACADR mac;                                     // source mac address of the request
00064     uint32_t expiration;                            // The tick that the lease expires on
00065     char hostname[DHCP_SERV_MAX_HOSTNAME_LEN + 1];  // buffer for client hostname
00066 };
00067
00068 struct DhcpInfo
00069 {
00070     IPADDR4 netmask;           // Netmask for the lease offered
00071     IPADDR4 gateway;           // gateway to be offered
00072     IPADDR4 dns_1;             // primary DNS Server
00073     IPADDR4 dns_2;             // secondary DNS Server
00074     IPADDR4 logServ;           // Syslog Server
00075     IPADDR4 smtpServ;          // SMTP Server
00076     IPADDR4 ntpServ;           // NTP Server
00077     const char *domain_name;   // Domain name for hosts
00078     char *hostname;            // Hostname
00079     char *tftp_name;           // tftp server name/dotted ip, null terminated
00080     char *bootfile;            // tftp boot file, null terminated
00081     uint16_t bonusLength;      // Length of any additional options, pre written
00082     uint8_t *bonusOpts;        // additional options, pre written
00083 };
00084
00085 // Lease Allocator is the base class/interface for lease allocators for the DHCP server
00086 class LeaseAllocator
00087 {
00088     LeaseAllocator *m_pNext;
00089
00090   public:
00091     LeaseAllocator();
00092     ~LeaseAllocator();
00093
00094     inline LeaseAllocator *SetNextAllocator(LeaseAllocator *nextAlloc)
00095     {
00096         m_pNext = nextAlloc;
00097         return m_pNext;
00098     }
00099     inline LeaseAllocator *GetNextAllocator() { return m_pNext; }
00100
00101     virtual uint32_t GetLeaseTime() = 0;
00102     virtual bool OfferLease(DhcpLeaseRequest *pLease, int intfNum) = 0;
00103     virtual bool RequestLease(DhcpLeaseRequest *pLease, int intfNum) = 0;
00104     virtual bool ReleaseLease(DhcpLeaseRequest *pLease, int intfNum) = 0;
00105     virtual bool LeaseValid(DhcpLeaseRequest *pLease, int intfNum) = 0;
00106     virtual bool GetDhcpInfo(DhcpInfo &infoBlock, MACADR &client_mac, int intfNum) = 0;
00107     virtual bool AddInterface(int intfNum) = 0;
00108     virtual void RemoveInterface(int intfNum) = 0;
00109     virtual bool GetLeaseData(DhcpLeaseData *data) = 0;
00110 };
00111
00112 // SingleAllocator, an incredibly stripped down allocator that fails to conform to many
00113 // standard behaviors, but shows the basics of what needs to be implemented
00114 class SingleAllocator : public LeaseAllocator
00115 {
00116     IPADDR4 m_theIP;
00117
00118   protected:
00119     uint32_t m_leaseDuration;   // Lease duration to hand out, in seconds
00120     DhcpInfo m_configInfo;
00121
00122   public:
00123     SingleAllocator(IPADDR4 ip) : LeaseAllocator(), m_theIP(ip), m_leaseDuration(3600) {}
```

```
00124     ~SingleAllocator();
00125
00126     inline void ChangeAllocAddr(IPADDR4 ip) { m_theIP = ip; }
00127
00128     inline virtual bool OfferLease(DhcpLeaseRequest *pLease, int intfNum)
00129     {
00130         pLease->ip = m_theIP;
00131         pLease->duration = m_leaseDuration;
00132         return true;
00133     }
00134     inline virtual bool RequestLease(DhcpLeaseRequest *pLease, int intfNum)
00135     {
00136         if (pLease->ip != m_theIP)
00137         {
00138             pLease->ip = 0x00000000;
00139             return false;
00140         }
00141         pLease->ip = m_theIP;
00142         pLease->duration = m_leaseDuration;
00143         return true;
00144     }
00145     inline virtual bool ReleaseLease(DhcpLeaseRequest *pLease, int intfNum) { return pLease->ip ==
      m_theIP; }
00146     inline virtual bool LeaseValid(DhcpLeaseRequest *pLease, int intfNum)
00147     {
00148         pLease->duration = m_leaseDuration;
00149         return pLease->ip == m_theIP;
00150     }
00151     virtual bool SetStaticLease(DhcpLeaseRequest *pLease) { return false; }
00152     virtual uint32_t GetLeaseTime() { return m_leaseDuration; }
00153
00154     void SetLeaseTime(uint32_t hours, uint32_t minutes = 0, uint32_t seconds = 0);
00155     virtual bool GetDhcpInfo(DhcpInfo &infoBlock, MACADR &client_mac, int intfNum);
00156     virtual void UpdateDhcpInfo(const DhcpInfo *infoBlock) { return; }
00157     virtual bool AddInterface(int intfNum) { return true; }
00158     virtual void RemoveInterface(int intfNum) { return; }
00159     virtual bool IsRegisteredInterface(int intfNum) { return true; }
00160     virtual bool GetLeaseData(DhcpLeaseData *data) { return false; }
00161 };
00162
00163 // BlockAllocator, A basic allocator that handles multiple leases in a contiguous IP block
00164 class BlockAllocator : public LeaseAllocator
00165 {
00166     IPADDR4 m_startIP;
00167     const int m_leaseCount;
00168     DhcpLeaseData *const m_leaseBlock;
00169     bool m_staticLeaseExist;
00170     uint32_t m_lastIndex;
00171     int m_validIntf[DHCP_SERV_MAX_INTF];
00172
00173     bool IsValidIntf(int intfNum);
00174     uint32_t m_leaseDuration;   // Lease duration to hand out, in seconds
00175     DhcpInfo m_configInfo;
00176
00177    public:
00178     BlockAllocator(const IPADDR4 startIP, const int leaseCount, DhcpLeaseData *const leaseBlock);
00179     ~BlockAllocator();
00180     void SetLeaseTime(uint32_t hours, uint32_t minutes = 0, uint32_t seconds = 0);
00181
00182     virtual uint32_t GetLeaseTime() { return m_leaseDuration; }
00183     virtual bool OfferLease(DhcpLeaseRequest *pLease, int intfNum);
00184     virtual bool RequestLease(DhcpLeaseRequest *pLease, int intfNum);
00185     virtual bool ReleaseLease(DhcpLeaseRequest *pLease, int intfNum);
00186     virtual bool LeaseValid(DhcpLeaseRequest *pLease, int intfNum);
00187     virtual bool SetStaticLease(DhcpLeaseRequest *pLease);
00188     virtual bool GetDhcpInfo(DhcpInfo &infoBlock, MACADR &client_mac, int intfNum);
00189     virtual void UpdateDhcpInfo(const DhcpInfo *infoBlock);
00190     virtual bool AddInterface(int intfNum);
00191     virtual void RemoveInterface(int intfNum);
00192     inline virtual bool IsRegisteredInterface(int intfNum)
00193     {
00194         for (int i = 0; i < DHCP_SERV_MAX_INTF; i++)
00195         {
00196             if (m_validIntf[i] == intfNum) { return true; }
00197         }
00198         return false;
00199     }
00200     virtual bool GetLeaseData(DhcpLeaseData *data);
00201
00202     void ResetLeases();
00203     inline void SetStartIP(const IPADDR4 newStartIP)
00204     {
00205         m_startIP = newStartIP;
00206         ResetLeases();
00207     }
00208 };
00209
```

```
00210 // MacPrefixAllocator is derived from BlockAllocator, with the ability to whitelist/blacklist
00211 // certain mac address ranges based on a mask. Useful for allocating addresses based on
00212 // device manufacturer
00213 class MacPrefixAllocator : public BlockAllocator
00214 {
00215     const MACADR m_macMask;     // the bitmask to filter against
00216     const MACADR m_macPrefix;   // the MAC prefix to scan against
00217     const bool m_whitelist;     // are we operating in Whitelist mode?
00218
00219   public:
00220     MacPrefixAllocator(const MACADR prefix,
00221                        const MACADR mask,
00222                        const bool whitelist,
00223                        const IPADDR4 startIP,
00224                        const int leaseCount,
00225                        DhcpLeaseData *const leaseBlock);
00226     ~MacPrefixAllocator();
00227     virtual bool OfferLease(DhcpLeaseRequest *pLease, int intfNum);
00228     virtual bool RequestLease(DhcpLeaseRequest *pLease, int intfNum);
00229     virtual bool SetStaticLease(DhcpLeaseRequest *pLease);
00230 };
00231
00232 // DHCP Server class
00233 // It requires a lease allocator to be added in order to function.
00234 class Server
00235 {
00236     static Server *theInstance;
00237     LeaseAllocator *m_pLeaseAlloc;
00238     uint32_t m_nextTick;
00239
00240     void ProcessDiscover(PoolPtr pp);
00241     void ProcessRequest(PoolPtr pp);
00242     void ProcessDecline(PoolPtr pp);
00243     void ProcessRelease(PoolPtr pp);
00244     void ProcessInform(PoolPtr pp);
00245
00246     void ProcessParamReq(uint8_t *&optBuf, uint8_t *ReqList, uint8_t reqLen, const DhcpInfo &info,
      IPADDR4 intfIP);
00247
00248   public:
00249     Server();
00250     ~Server();
00251     bool AddLeaseAllocator(LeaseAllocator *newAllocator);
00252     bool ProcessServerMessage(PoolPtr pp);
00253
00254     bool GetDhcpClients(DhcpLeaseData *data);
00255
00256     bool AddInterface(int intfNum);
00257     void RemoveInterface(int intfNum);
00258
00259     static inline bool AddServerInterface(int intfNum)
00260     {
00261         if (theInstance) { return theInstance->AddInterface(intfNum); }
00262         return false;
00263     }
00264     static inline void RemoveServerInterface(int intfNum)
00265     {
00266         if (theInstance) { theInstance->RemoveInterface(intfNum); }
00267     }
00268     static inline void ProcessMessage(PoolPtr pp)
00269     {
00270         if (theInstance) { theInstance->ProcessServerMessage(pp); }
00271     }
00272     static inline Server *GetInstance() { return theInstance; }
00273 };
00274 }   // namespace DHCP
00275
00290 bool AddStandardDHCPServer(int intf = 0, IPADDR4 startAddr = IPADDR4::NullIP());
00291
00292 #endif /* ----- #ifndef __DHCPD_H  ----- */
00293
```

24.321 dhcpinternals.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* @file dhcpinternals.h
00006  *  @brief IPv4 DHCP Client Header File for Internal Features
00007  */
00008
00009 /* @addtogroup groupDhcpIPv4 DHCP - IPv4 DHCP Client
00010  *   @{
00011  */
```

```
00012
00013 #include <basictypes.h>
00014 #include <netinterface.h>
00015 #include <string.h>
00016
00017 #ifndef _NB_DHCP_H
00018 #define _NB_DHCP_H
00019
00020 #define OPT_SIZE 255          // Size of options area
00021 #define DHCP_OPT_OFFS 236     // Offset into DHCP Message for options data
00022 #define DHCP_CLIENT_PORT 68   // Standard port assignment for client
00023 #define DHCP_SERVER_PORT 67   // Standard port assignment for server
00024
00025 //----- DHCP message types -----
00026 #define DHCPDISCOVER 1
00027 #define DHCPOFFER 2
00028 #define DHCPREQUEST 3
00029 #define DHCPDECLINE 4
00030 #define DHCPACK 5
00031 #define DHCPNAK 6
00032 #define DHCPRELEASE 7
00033 #define DHCPINFORM 8
00034 #define DHCPERROR 99
00035
00036 //----- DHCP message op codes -----
00037 #define BOOTREQUEST 1
00038 #define BOOTREPLY 2
00039
00040 //----- DHCP option codes -----
00041 #define DHCPOPT_SUBNET_MASK 1    // Client subnet mask
00042 #define DHCPOPT_ROUTER 3         // Routers
00043 #define DHCPOPT_TIME_SERVER 4    // Time Servers
00044 #define DHCPOPT_DNS_SERVER 6     // DNS Server
00045 #define DHCPOPT_LOG_SERVER 7     // Log Server
00046 #define DHCPOPT_HOST_NAME 12     // Specifies name of client (Host Name Opt)
00047 #define DHCPOPT_DOMAIN_NAME 15   // Domain Name for hosts
00048 #define DHCPOPT_NTP_SERVER 42    // NTP Servers (different from tim)
00049 #define DHCPOPT_REQ_IPADDR4 50   // Client requested IP address
00050 #define DHCPOPT_LEASE_TIME 51    // IP addr lease time assigned by server
00051 #define DHCPOPT_MSG_TYPE 53      // Type of DHCP message
00052 #define DHCPOPT_SERVER_ID 54     // DHCP Server identifier
00053 #define DHCPOPT_PARAM_REQ 55     // Parameter request list
00054 #define DHCPOPT_RENEW_TIME 58    // # of seconds until client renewal state
00055 #define DHCPOPT_REBIND_TIME 59   // # of seconds until client rebinding state
00056 #define DHCPOPT_CLIENT_ID 61     // Unique client identifier
00057 #define DHCPOPT_TFTP_SERVER 66   // TFTP Server name/IP
00058 #define DHCPOPT_BOOTFILE 67      // TFTP Bootfile name
00059 #define DHCPOPT_SMTP_SERVER 69   // SMTP Server name/IP
00060
00061 #define DHCPOPT_END 255   // End of options marker
00062
00063 #define DHCPCOOKIE0 (uint8_t)99   // DHCP message 4-byte cookie values
00064 #define DHCPCOOKIE1 (uint8_t)130
00065 #define DHCPCOOKIE2 (uint8_t)83
00066 #define DHCPCOOKIE3 (uint8_t)99
00067
00068 //----- ProcessDhcpMsg function return definitions -----
00069 /*
00070 #define PDM_DHCPOFFER   1  // Received DHCPOFFER message
00071 #define PDM_DHCPACK     2  // Received DHCPACK message
00072 #define PDM_DHCPNAK     3  // Received DHCPNAK message
00073 #define PDM_COMPLETE    4  // DHCP configuration complete
00074 #define PDM_ERROR      10  // Received unknown message
00075 */
00076
00077 //----- DHCP Message Structure -----
00078 typedef struct dhcp_msg
00079 {
00080     uint8_t op;                  // Message opcode; 1=bootrequest, 2=bootreply
00081     uint8_t htype;               // Hardware address type; 1=10Mb ethernet
00082     uint8_t hlen;                // Hardware address length; 6=10Mb ethernet
00083     uint8_t hops;                // Client sets this to 0
00084     beuint32_t xid;              // Transaction ID; random number
00085     beuint16_t secs;             // Seconds elapsed since client request
00086     beuint16_t flags;            //
00087     IPADDR4 ciaddr;              // Clinet IP address if renew; filled by client
00088     IPADDR4 yiaddr;              // Assigned client IP address
00089     IPADDR4 siaddr;              // IP address of next server to use in bootstrap
00090     IPADDR4 giaddr;              // Relay agent IP address (Gateway)
00091     uint8_t chaddr[16];          // Client hardware address
00092     uint8_t sname[64];           // Optional server host name
00093     uint8_t file[128];           // Boot file name, null terminated string
00094     uint8_t options[OPT_SIZE];   // Optional parameters field
00095 } __attribute__((packed)) DHCP_MESSAGE;
00096
00097 class InterfaceBlock;   // forward
00098
```

```
00099 //----- DHCP message class -----
00100 class DHCPMessage
00101 {
00102    private:
00103     DHCP_MESSAGE Msg;
00104
00105    public:
00106     // constructors
00107     DHCPMessage();
00108     DHCPMessage(puint8_t pData);
00109
00110     // member functions
00111     void SetOp(uint8_t n) { Msg.op = n; }
00112
00113     uint8_t GetMsgType() { return msg_type; }
00114
00115     void SetHtype(uint8_t n) { Msg.htype = n; }
00116     void SetHlen(uint8_t n) { Msg.hlen = n; }
00117     void SetHops(uint8_t n) { Msg.hops = n; }
00118
00119     void SetXid(uint32_t n) { Msg.xid = n; }
00120
00121     void SetRandomXid(const MACADR &ma);
00122
00123     void SetSecs(uint16_t n) { Msg.secs = n; }
00124     void SetFlags(uint16_t n) { Msg.flags = n; }
00125     void SetCiaddr(IPADDR4 i) { Msg.ciaddr = i; }
00126     void SetYiaddr(IPADDR4 i) { Msg.yiaddr = i; }
00127     void SetSiaddr(IPADDR4 i) { Msg.siaddr = i; }
00128     void SetGiaddr(IPADDR4 i) { Msg.giaddr = i; }
00129     void SetChaddr(MACADR ma)
00130     {
00131         memset(Msg.chaddr, 0x0, 16);
00132         for (int i = 0; i < 6; i++)
00133         {
00134             Msg.chaddr[i] = ma.GetByte(i);
00135         }
00136     }
00137
00138     void SetSname(char *s) { memcpy(Msg.sname, s, 64); }
00139     void SetFile(char *s) { memcpy(Msg.file, s, 128); }
00140     void SetOptions(char *s) { memcpy(Msg.options, s, OPT_SIZE); }
00141
00142     void SetOptionsIp(int &offset, IPADDR4 &ip4)
00143     {
00144         uint32_t u32 = (uint32_t)ip4;
00145         Msg.options[offset++] = ((u32 » 24) & 0xff);
00146         Msg.options[offset++] = ((u32 » 16) & 0xff);
00147         Msg.options[offset++] = ((u32 » 8) & 0xff);
00148         Msg.options[offset++] = (u32 & 0xff);
00149     }
00150
00151     void SetOptions(int i, uint8_t n) { Msg.options[i] = n; }
00152
00153     uint32_t GetXid() { return Msg.xid; }
00154     uint16_t GetSecs() { return Msg.secs; }
00155     uint16_t GetFlags() { return Msg.flags; }
00156     IPADDR4 GetCiaddr() { return Msg.ciaddr; }
00157     IPADDR4 GetYiaddr() { return Msg.yiaddr; }
00158     IPADDR4 GetGiaddr() { return Msg.giaddr; }
00159     IPADDR4 GetSiaddr() { return Msg.siaddr; }
00160     uint8_t GetChaddr(int i) { return Msg.chaddr[i]; }
00161
00162     puint8_t GetDataPtr() { return (uint8_t *)(&Msg); }
00163     int GetDataLen() { return sizeof(DHCP_MESSAGE); }
00164     void ShowMsg();                                  // Displays DHCP message for debug purposes
00165     void CreateReply(uint32_t xid, uint32_t yip);    // Sets the message as a blank BOOTP reply
00166     void ClearOptions();                             // Clears options memory space and adds cookie
00167     void ResetBytes();                               // Clear chaddr, sname, file, and option fields
00168     puint8_t GetOptionData(uint8_t code);            // Returns option data for specified op code
00169     puint8_t GetOptionData(uint8_t code, int &length);
00170     void SendMsg(IPADDR4 IpAddr, InterfaceBlock *pIfb);         // Send a DHPC message as a UDP packet
00171     void SendServerMsg(IPADDR4 IpAddr, InterfaceBlock *pIfb);   // Send a DHPC message as a UDP
    packet, as server
00172     BOOL MsgForInterface(InterfaceBlock *pIfb);
00173     uint8_t msg_type;
00174 };
00175
00176 //----- function prototypes -----
00177 void CreateDhcpDiscoverMsg(DHCPMessage &NewMsg, InterfaceBlock *pIfb);
00178 void CreateDhcpRequestMsg(DHCPMessage &OfferMsg, DHCPMessage &NewMsg, InterfaceBlock *pIfb);
00179 void CreateDhcpReleaseMsg(DHCPMessage &NewMsg, InterfaceBlock *pIfb);
00180 void ShowDhcpConfig();                       // Show DHCP configuration paramters
00181 void UpdateIPRuntimeVars(BOOLEAN release);   // Update DHCP & IP runtime vars
00182 void DHCPTimer();                            // Tracks lease time
00183 int ExecDHCPClient(int DhcpState);           // Function that does the work
00184 BOOLEAN ValidDhcpMsg(puint8_t pData);        // Verify the UDP packet contains a DHCP message
```

```
00185 BOOLEAN DhcpConfig(DHCPMessage &Msg);        // Configure client from DHCPACK message
00186
00187 /* function pointers to extend DHCP options */
00188 extern void (*AddDhcpOptionsFunc)(DHCPMessage &NewMsg, int &opt, int MsgType);
00189 extern void (*AddDhcpFieldsFunc)(DHCPMessage &NewMsg, int &opt, int MsgType);
00190 extern void (*ParseDhcpOptions)(DHCPMessage &Msg);
00191
00192 #endif
00193
00194 /* @} */
```

24.322 diagnostics.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005
00006
00007 #ifndef DIAGNOSTICS_INTERNAL_H
00008 #define DIAGNOSTICS_INTERNAL_H
00009
00010 /*The functions in this header are used to add idagnostic reports to the diagnostics tab under the
      config page.
00011 The resulting JSON blob can be accessed directly (replace 10.1.1.190 with your IP or dns name)
00012 http://10.1.1.190:20034/DIAG
00013
00014
00015 You can also acces sub groups from the diagnostics...
00016 http://10.1.1.190:20034/DIAG/Buffers
00017 or
00018 http://10.1.1.190:20034/DIAG/Tasks
00019 */
00020
00021
00022
00023 #include <nbstring.h>
00024
00025 class DiagItemClass
00026 {
00027    private:
00028     NBString Name;
00029     DiagItemClass *m_pNext;
00030     virtual void ServeContent(int fd) = 0;
00031     static void ServerRoot(int fd, const char *pUrl);
00032     void InsertInList();
00033     void RemoveList();
00034
00035    public:
00036         static void ServeBody(int fd);
00037     DiagItemClass(const char *name);
00038     DiagItemClass(const NBString &ns);
00039     ~DiagItemClass();
00040 };
00041
00042 typedef void(diagcallback)(int sock);
00043
00044
00045 //Call back and generate the specific diagnostic element when requested
00046 class DiagCallBack : public DiagItemClass
00047 {
00048     diagcallback *m_pF;
00049     virtual void ServeContent(int fd) { m_pF(fd); }
00050    public:
00051     DiagCallBack(const char *name, diagcallback *pFunc) : DiagItemClass(name) { m_pF = pFunc; }
00052 };
00053
00054
00055
00056 //Report the value of some numeric type
00057 class DiagVarMon : public DiagItemClass
00058 {
00059    private:
00060     volatile void *p;
00061     size_t siz;
00062     uint16_t flag;
00063
00064    public:
00065     DiagVarMon(const char *name, uint32_t &v) : DiagItemClass(name)
00066     {
00067         p = &v;
00068         siz = sizeof(v);
00069         flag = 1;
00070     };
00071     DiagVarMon(const char *name, uint16_t &v) : DiagItemClass(name)
```

```
00072     {
00073         p = &v;
00074         siz = sizeof(v);
00075         flag = 1;
00076     };
00077     DiagVarMon(const char *name, uint8_t &v) : DiagItemClass(name)
00078     {
00079         p = &v;
00080         siz = sizeof(v);
00081         flag = 1;
00082     };
00083     DiagVarMon(const char *name, int32_t &v) : DiagItemClass(name)
00084     {
00085         p = &v;
00086         siz = sizeof(v);
00087         flag = 0;
00088     };
00089     DiagVarMon(const char *name, int16_t &v) : DiagItemClass(name)
00090     {
00091         p = &v;
00092         siz = sizeof(v);
00093         flag = 0;
00094     };
00095     DiagVarMon(const char *name, int8_t &v) : DiagItemClass(name)
00096     {
00097         p = &v;
00098         siz = sizeof(v);
00099         flag = 0;
00100     };
00101     DiagVarMon(const char *name, char &v) : DiagItemClass(name)
00102     {
00103         p = &v;
00104         siz = sizeof(v);
00105         flag = 0;
00106     };
00107     DiagVarMon(const char *name, int &v) : DiagItemClass(name)
00108     {
00109         p = &v;
00110         siz = sizeof(v);
00111         flag = 0;
00112     };
00113     DiagVarMon(const char *name, volatile uint32_t &v) : DiagItemClass(name)
00114     {
00115         p = &v;
00116         siz = sizeof(v);
00117         flag = 1;
00118     };
00119     DiagVarMon(const char *name, volatile uint16_t &v) : DiagItemClass(name)
00120     {
00121         p = &v;
00122         siz = sizeof(v);
00123         flag = 1;
00124     };
00125     DiagVarMon(const char *name, volatile uint8_t &v) : DiagItemClass(name)
00126     {
00127         p = &v;
00128         siz = sizeof(v);
00129         flag = 1;
00130     };
00131     DiagVarMon(const char *name, volatile int32_t &v) : DiagItemClass(name)
00132     {
00133         p = &v;
00134         siz = sizeof(v);
00135         flag = 0;
00136     };
00137     DiagVarMon(const char *name, volatile int16_t &v) : DiagItemClass(name)
00138     {
00139         p = &v;
00140         siz = sizeof(v);
00141         flag = 0;
00142     };
00143     DiagVarMon(const char *name, volatile int8_t &v) : DiagItemClass(name)
00144     {
00145         p = &v;
00146         siz = sizeof(v);
00147         flag = 0;
00148     };
00149     DiagVarMon(const char *name, volatile char &v) : DiagItemClass(name)
00150     {
00151         p = &v;
00152         siz = sizeof(v);
00153         flag = 0;
00154     };
00155     DiagVarMon(const char *name, volatile int &v) : DiagItemClass(name)
00156     {
00157         p = &v;
00158         siz = sizeof(v);
```

```
00159         flag = 0;
00160     };
00161     DiagVarMon(const char *name, volatile float &v) : DiagItemClass(name)
00162     {
00163         p = &v;
00164         siz = sizeof(v);
00165         flag = 2;
00166     };
00167     DiagVarMon(const char *name, volatile double &v) : DiagItemClass(name)
00168     {
00169         p = &v;
00170         siz = sizeof(v);
00171         flag = 2;
00172     };
00173     DiagVarMon(const char *name, float &v) : DiagItemClass(name)
00174     {
00175         p = &v;
00176         siz = sizeof(v);
00177         flag = 2;
00178     };
00179     DiagVarMon(const char *name, double &v) : DiagItemClass(name)
00180     {
00181         p = &v;
00182         siz = sizeof(v);
00183         flag = 2;
00184     };
00185     virtual void ServeContent(int fd);
00186 };
00187
00188
00189
00190 //Report the value of a string....
00191 class DiagStrMon : public DiagItemClass
00192 {
00193    private:
00194     volatile const char *pString;
00195     NBString * pStrStr;
00196
00197    public:
00198     DiagStrMon(const char *name, const char *pstr) : DiagItemClass(name) { pString = pstr;
      pStrStr=0;};
00199     DiagStrMon(const char *name, NBString & s) : DiagItemClass(name) { pString = 0;  pStrStr=&s;};
00200     virtual void ServeContent(int fd);
00201 };
00202
00203
00204
00205 //Report the value of a pointer
00206 class DiagPtrMon : public DiagItemClass
00207 {
00208    private:
00209     volatile void *pPtr;
00210
00211    public:
00212     DiagPtrMon(const char *name, void *p) : DiagItemClass(name) { pPtr = p; };
00213     virtual void ServeContent(int fd);
00214 };
00215
00216 /* Examples
00217 Suppose I have a uint32_t called myVar I want to monitor...
00218 DiagVarMon myvarmon("MyVarName",myVar);
00219
00220 Suppose I have char buffer holding the name opf the current mode...
00221
00222 DiagStrMon MyStrMon("CurMode",cur_mode);
00223
00224
00225 Lets say I want to output a complex JSON object for a custom report....
00226 I write a routine that outputs this object to an fd...
00227 void MyCustomReport(int fd);
00228
00229 DiagCallBack myReporter("MyReportName",MyCustomReport);
00230 */
00231
00232
00233 class ParsedURI;
00234
00235 /* Set up to send the diagnostic report as a JSON Blob to some external URL */
00236 bool SendDiagReport(const char * pUrl,const char *pUser = 0, const char *pPass = 0);
00237 bool SendDiagReport(ParsedURI & uri,const char *pUser = 0, const char *pPass = 0 );
00238
00239
00240
00241
00242
00243
00244 #endif
```

24.323 discoveryservlet.h

```
00001
00002 /*NB_REVISION*/
00003
00004 /*NB_COPYRIGHT*/
00005
00006 #ifndef DISCO_SERVLET
00007 #define DISCO_SERVLET
00008
00009 #include <nbstring.h>
00010 #include <servlets.h>
00011 class InterfaceBlock;
00012
00013 class discover_servlet : public servlet
00014 {
00015     uint16_t state;
00016     int16_t sub_state;
00017     int16_t v;   // Temp value
00018     int16_t result;
00019     InterfaceBlock *m_pIfb;
00020     int m_fd;
00021
00022     uint32_t LastValidResponse;
00023     uint32_t LastAttempt;
00024     IPADDR ServerAddress;
00025
00026     virtual int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00027     virtual void ProcessSelectResult(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00028     void FillInName(NBString &ns);
00029     int GetInterval();
00030
00031   public:
00032     discover_servlet(InterfaceBlock *ib);
00033     bool Started();
00034 };
00035
00036 #endif
```

24.324 dns.h File Reference

NetBurner Domain Name Server Header File.

```
#include <nettypes.h>
#include <nbrtos.h>
#include <buffers.h>
#include <udp.h>
```

Macros

- #define **DNS_OK** (0)

  Success.
- #define **DNS_TIMEOUT** (1)

  Request timed out.
- #define **DNS_NOSUCHNAME** (2)

  Name not found.
- #define **DNS_ERR** (3)

  Other error.
- #define **DNS_A** 1

  32-bit IPv4 address
- #define **DNS_CNAME** 5

  Canonical name record.
- #define **DNS_MB** 7

  Mailing list subscriber list.
- #define **DNS_MG** 8

  Mailing list subscriber list.
- #define **DNS_MX** 15

Mail exchange record.

- #define **DNS_AAAA** 28

  128-bit IPv6 address

Functions

- bool IsNameIPAddress (const char *name)

  Determine if the name is a valid IP Address and does not need to be looked up.

- int fd_dns_part1 (const char *name, const IPADDR &dns_server, uint16_t TYPE=DNS_A, uint16_t TYPE2=0, int ifn=-1)

  Open a UDP socket and initiate a DNS lookup.

- bool fd_dns_processresult (int fd, const char *name, IPADDR &addr_out, uint16_t TYPE=DNS_A, uint16_t TYPE2=DNS_AAAA, uint32_t *ttl=0)

  Process any responses on the UDP socket opened for DNS.

- int fd_outstanding_Responses (int fd)

  Check to see if there are any outstanding DNS requests.

- bool AnyDNSInterFaceActive ()

  Determine if we have an active DNS route; DNS server is set for an active interface.

- int GetHostByName (const char *name, IPADDR *plpaddr, const IPADDR &dns_server, const TickTimeout tout, uint16_t TYPE1=DNS_A, uint16_t TYPE2=extra_dns_t, uint32_t *ttl=NULL)

  Get the IP address associated with the specified domain name.

- int GetHostByNameViaIfNum (const char *name, IPADDR *plpaddr, const IPADDR &dns_server, int ifn, const TickTimeout &tout, uint16_t TYPE1=DNS_A, uint16_t TYPE2=extra_dns_t, uint32_t *ttl=NULL)

  Get the IP address associated with the specified domain name on a specific interface.

24.324.1 Detailed Description

NetBurner Domain Name Server Header File.

24.325 dns.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00026 #ifndef _NB_DNS_H
00027 #define _NB_DNS_H
00028
00029 #include <nettypes.h>
00030 #include <nbrtos.h>
00031 #include <buffers.h>
00032 #include <udp.h>
00033
00037 #define DNS_OK (0)
00038 #define DNS_TIMEOUT (1)
00039 #define DNS_NOSUCHNAME (2)
00040 #define DNS_ERR (3)
00048 #define DNS_A 1
00049 #define DNS_CNAME 5
00050 #define DNS_MB 7
00051 #define DNS_MG 8
00052 #define DNS_MX 15
00053 #define DNS_AAAA 28
00057 #ifdef IPV6
00058 const uint16_t extra_dns_t=DNS_AAAA;
00059 #else
00060 const uint16_t extra_dns_t=DNS_A;
00061 #endif
00062
00063
00073 bool IsNameIPAddress(const char *name);
00074
00091 int fd_dns_part1(const char *name,const  IPADDR &dns_server, uint16_t TYPE = DNS_A, uint16_t TYPE2 =
      0, int ifn = -1);
00092
00093
```

```
00110 #ifndef IPV4ONLY
00111 bool fd_dns_processresult(int fd, const char *name, IPADDR &addr_out, uint16_t TYPE = DNS_A, uint16_t
      TYPE2 = DNS_AAAA, uint32_t *ttl = 0);
00112
00113 inline bool fd_dns_processresult(int fd, const char *name, IPADDR4 &addr_out, uint16_t TYPE1, uint16_t
      TYPE2, uint32_t *ttl)
00114 {
00115     IPADDR ir;
00116     bool result = fd_dns_processresult(fd, name, ir, TYPE1, TYPE2, ttl);
00117     if (result)
00118         addr_out = ir.Extract4();
00119     return result;
00120 }
00121 #else
00122 bool fd_dns_processresult(int fd, const char *name, IPADDR &addr_out, uint16_t TYPE = DNS_A, uint16_t
      TYPE2 = 0, uint32_t *ttl = 0);
00123
00124
00125 #endif
00126
00127
00139 int fd_outstanding_Responses(int fd);
00140
00141
00142 // Used internally
00143 int RootGetHostByName(const char *name, IPADDR & Ipa,const IPADDR & dnsServer,const  TickTimeout &
      timeout, uint16_t TYPE, uint16_t TYPE2,int ifn_select, uint32_t *ttl);
00144
00145
00146
00156  bool AnyDNSInterFaceActive();
00157
00158
00182 inline
00183 int GetHostByName(const char *name,
00184                   IPADDR *pIpaddr,
00185                   const IPADDR &dns_server,
00186                   const TickTimeout tout,
00187                   uint16_t TYPE1 = DNS_A,
00188                   uint16_t TYPE2 = extra_dns_t,
00189                   uint32_t *ttl = NULL)
00190 {
00191     return RootGetHostByName(name,*pIpaddr,dns_server,tout,TYPE1,TYPE2, -1,ttl);
00192 }
00193
00194 #ifndef IPV4ONLY
00195 inline
00196 int GetHostByName(const char *name,
00197                   IPADDR4 *pI4,
00198                   const IPADDR &dns_server,
00199                   const TickTimeout tout,
00200                   uint16_t TYPE1 = DNS_A,
00201                   uint16_t TYPE2 = extra_dns_t,
00202                   uint32_t *ttl = NULL)
00203 {
00204     IPADDR ir;
00205     int rv=RootGetHostByName(name,ir,dns_server,tout,TYPE1,TYPE2, -1,ttl);
00206     if((rv==DNS_OK) && (pI4)) *pI4=ir.Extract4();
00207     return rv;
00208 }
00209 #endif
00210
00235 inline
00236 int GetHostByNameViaIfNum(
00237                    const char *name,
00238                    IPADDR *pIpaddr,
00239                    const IPADDR & dns_server,
00240                    int ifn,
00241                    const TickTimeout &tout,
00242                    uint16_t TYPE1 = DNS_A,
00243                    uint16_t TYPE2 = extra_dns_t,
00244                    uint32_t *ttl = NULL)
00245 {
00246     return RootGetHostByName(name,*pIpaddr,dns_server,tout,TYPE1,TYPE2, ifn,ttl);
00247 }
00248
00249 #ifndef IPV4ONLY
00250 inline
00251 int GetHostByNameViaIfNum(
00252                    const char *name,
00253                    IPADDR4 *pI4,
00254                    const IPADDR & dns_server,
00255                    int ifn,
00256                    const TickTimeout &tout,
00257                    uint16_t TYPE1 = DNS_A,
00258                    uint16_t TYPE2 = extra_dns_t,
00259                    uint32_t *ttl = NULL)
```

```
00260 {
00261     IPADDR ir;
00262     int rv=RootGetHostByName(name,ir,dns_server,tout,TYPE1,TYPE2, ifn,ttl);
00263     if((rv==DNS_OK) && (pI4)) *pI4=ir.Extract4();
00264     return rv;
00265 }
00266
00267 #endif
00268
00269
00270
00271 #ifdef DNS_CACHE
00272 //Internal use
00273 void RegisterResult(const char* name, IPADDR ia,uint32_t ttl);
00274
00284 bool CheckDnsCache(const char * name, IPADDR & ia, uint32_t *ttl);
00285 #endif
00286
00287
00288 #endif
00289
```

24.326 api_f.h File Reference

Embedded Flash File System - FAT.

```
#include <nbrtos.h>
#include "fwerr.h"
```

Macros

- #define **F_SEEK_SET** FN_SEEK_SET

  Beginning of file.
- #define **F_SEEK_END** FN_SEEK_END

  End of file.
- #define **F_SEEK_CUR** FN_SEEK_CUR

  Current position of the file pointer.
- #define f_delvolume(drvnumber) fm_delvolume(drvnumber)

  Un-mounts a flash card.
- #define f_getfreespace(drivenum, pspace) fm_getfreespace(drivenum, pspace)

  Provides information about the drive space usage.
- #define f_chdir(dirname) fm_chdir(dirname)

  Change the directory.
- #define f_mkdir(dirname) fm_mkdir(dirname)

  Makes a new directory.
- #define f_rmdir(dirname) fm_rmdir(dirname)

  Removes a directory.
- #define f_findfirst(filename, find) fm_findfirst(filename, find)

  Find the first file or subdirectory in a specified directory.
- #define f_findnext(find) fm_findnext(find)

  Finds the next file or subdirectory in a specified directory after a previous call to `f_findfirst()` or `f_findnext()`.
- #define f_close(filehandle) fm_close(filehandle)

  Closes an opened file.
- #define f_open(filename, mode) fm_open(filename, mode)

  Opens a file in the file system.
- #define f_read(buf, size, size_st, filehandle) fm_read(buf, size, size_st, filehandle)

  Read data from the current position in a file.
- #define f_write(buf, size, size_st, filehandle) fm_write(buf, size, size_st, filehandle)

  Write data to the file at the current position.

- #define f_seek(filehandle, offset, whence) fm_seek(filehandle, offset, whence)

  Move the stream position of an open file.
- #define f_rewind(filehandle) fm_rewind(filehandle)

  Sets the file position in the open target file to the start of the file.
- #define f_eof(filehandle) fm_eof(filehandle)

  Check whether the current position in the open target file is the end of the file.
- #define f_gettimedate(filename, pctime, pcdate) fm_gettimedate(filename, pctime, pcdate)

  Get the time and date of a file or directory.
- #define f_settimedate(filename, ctime, cdate) fm_settimedate(filename, ctime, cdate)

  Set the time and date of a file or directory.
- #define f_delete(filename) fm_delete(filename)

  Deletes a file.

Functions

- int f_enterFS (void)

  Adds a new task priority to the task list used by the file system.
- void f_releaseFS (void)

  Removes a task priority from the task list used by the file system.

24.326.1 Detailed Description

Embedded Flash File System - FAT.

24.327 api_f.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00020 #ifndef _API_F_H_
00021 #define _API_F_H_
00022
00023 /***************************************************************************
00024  *
00025  *            Copyright (c) 2003-2006 by HCC Embedded
00026  *
00027  * This software is copyrighted by and is the sole property of
00028  * HCC.  All rights, title, ownership, or other interests
00029  * in the software remain the property of HCC.  This
00030  * software may only be used in accordance with the corresponding
00031  * license agreement.  Any unauthorized use, duplication, transmission,
00032  * distribution, or disclosure of this software is expressly forbidden.
00033  *
00034  * This Copyright notice may not be removed or modified without prior
00035  * written consent of HCC.
00036  *
00037  * HCC reserves the right to modify this software without notice.
00038  *
00039  * HCC Embedded
00040  * Budapest 1132
00041  * Victor Hugo Utca 11-15
00042  * Hungary
00043  *
00044  * Tel:  +36 (1) 450 1302
00045  * Fax:  +36 (1) 450 1303
00046  * http: www.hcc-embedded.com
00047  * email: info@hcc-embedded.com
00048  *
00049  ***************************************************************************/
00050
00051 #include <nbrtos.h>
00052
00053 /***************************************************************************
00054  *
00055  * open bracket for C++ compatibility
00056  *
00057  ***************************************************************************/
00058
00059 #ifdef __cplusplus
```

```
00060 extern "C"
00061 {
00062 #endif
00063
00064     /*****************************************************************************
00065      *
00066      * if there is no udefs_f.h is included before
00067      * then this followed #define must be revised to be compatible
00068      * with UNICODE or LFN or 8+3 system
00069      * also FN_MAXPATH and FN_MUTEX_TYPE must be set to original
00070      *
00071      *****************************************************************************/
00072
00073 #ifndef _UDEFS_F_H_
00074
00075     /*****************************************************************************
00076      *
00077      *  if Unicode is used then comment in HCC_UNICODE define
00078      *
00079      *****************************************************************************/
00080
00081     /* #define HCC_UNICODE  */
00082
00083 #ifndef HCC_UNICODE
00084 #define F_LONGFILENAME 1 /*  0 - 8+3 names   1 - long file names   */
00085 #define W_CHAR char
00086 #else
00087 #define F_LONGFILENAME 1 /* don't change it, because unicode version always uses long file name */
00088 #define W_CHAR wchar
00089 #endif
00090
00091 #define FN_MAXPATH 256 /* maximum allowed filename or pathname */
00092 #define FN_MUTEX_TYPE OS_CRIT
00093
00094 #ifdef HCC_UNICODE
00095     typedef unsigned short wchar;
00096 #endif
00097
00098     /*****************************************************************************
00099      *
00100      * End of udefs.h definition checking
00101      *
00102      *****************************************************************************/
00103
00104 #endif /* #ifndef _UDEFS_F_H_ */
00105
00106 /* definition of short filename */
00107 #define F_MAXNAME 8 /*  8 byte name  */
00108 #define F_MAXEXT 3  /*  3 byte extension  */
00109
00110 #ifndef NULL
00111 #define NULL (void *)0
00112 #endif
00113
00114     /* public structure for FN_FILE */
00115     typedef struct
00116     {
00117         void *reference; /* reference which fileint used */
00118     } FN_FILE;
00119
00120 /* F_NAME structure definition */
00121 #if (!F_LONGFILENAME)
00122     typedef struct
00123     {
00124         int drivenum;             /*  drive number 0-A 1-B 2-C  */
00125         char path[FN_MAXPATH];    /*  pathnam  /directory1/dir2/   */
00126         char filename[F_MAXNAME]; /*  filename  */
00127         char fileext[F_MAXEXT];   /*  extension  */
00128     } F_NAME;
00129 #else
00130 #define F_MAXLNAME 256 /* maximum length of long filename */
00131 typedef struct
00132 {
00133     int drivenum;             /*  drive number 0-A 1-B 2-C  */
00134     W_CHAR path[FN_MAXPATH];  /*  pathname /directory1/dir2/   */
00135     W_CHAR lname[F_MAXLNAME]; /*  long file name   */
00136 } F_NAME;
00137 #endif                /*  F_LONGFILENAME   */
00138
00139     typedef struct
00140     {
00141         unsigned long cluster;     /* which cluster is used */
00142         unsigned long prevcluster; /* previous cluster for bad block handling */
00143         unsigned long sectorbegin; /* calculated sector start */
00144         unsigned long sector;      /* current sector */
00145         unsigned long sectorend;   /* last saector position of the cluster */
00146         unsigned long pos;         /* current position */
```

```
00147     } F_POS;
00148
00149     typedef struct
00150     {
00151         char filename[FN_MAXPATH]; /* file name+ext */
00152         char name[F_MAXNAME];      /* file name */
00153         char ext[F_MAXEXT];        /* file extension */
00154         unsigned char attr;        /* attribute of the file */
00155
00156         unsigned short ctime;   /* creation time */
00157         unsigned short cdate;   /* creation date */
00158         unsigned long filesize; /* length of file */
00159
00160         unsigned long cluster; /* current file starting position */
00161         F_NAME findfsname;     /* find properties */
00162         F_POS pos;             /* position of the current list */
00163     } FN_FIND;
00164
00165 #ifdef HCC_UNICODE
00166     typedef struct
00167     {
00168         W_CHAR filename[FN_MAXPATH]; /* file name+ext */
00169         char name[F_MAXNAME];        /* file name */
00170         char ext[F_MAXEXT];          /* file extension */
00171         unsigned char attr;          /* attribute of the file */
00172
00173         unsigned short ctime;   /* creation time */
00174         unsigned short cdate;   /* creation date */
00175         unsigned long filesize; /* length of file */
00176
00177         unsigned long cluster; /* current file starting position */
00178         F_NAME findfsname;     /* find properties */
00179         F_POS pos;             /* position of the current list */
00180     } FN_WFIND;
00181 #endif
00182
00183 /* attribute file/directory bitpattern definitions */
00184 #define F_ATTR_ARC 0x20
00185 #define F_ATTR_DIR 0x10
00186 #define F_ATTR_VOLUME 0x08
00187 #define F_ATTR_SYSTEM 0x04
00188 #define F_ATTR_HIDDEN 0x02
00189 #define F_ATTR_READONLY 0x01
00190
00191 /*  definitions for ctime  */
00192 #define F_CTIME_SEC_SHIFT 0
00193 #define F_CTIME_SEC_MASK 0x001f /* 0-30 in 2seconds */
00194 #define F_CTIME_MIN_SHIFT 5
00195 #define F_CTIME_MIN_MASK 0x07e0 /* 0-59  */
00196 #define F_CTIME_HOUR_SHIFT 11
00197 #define F_CTIME_HOUR_MASK 0xf800 /* 0-23 */
00198
00199 /*  definitions for cdate  */
00200 #define F_CDATE_DAY_SHIFT 0
00201 #define F_CDATE_DAY_MASK 0x001f /* 0-31 */
00202 #define F_CDATE_MONTH_SHIFT 5
00203 #define F_CDATE_MONTH_MASK 0x01e0 /* 1-12 */
00204 #define F_CDATE_YEAR_SHIFT 9
00205 #define F_CDATE_YEAR_MASK 0xfe00 /* 0-119 (1980+value) */
00206
00207     typedef struct
00208     {
00209         unsigned short number_of_cylinders;
00210         unsigned short sector_per_track;
00211         unsigned short number_of_heads;
00212         unsigned long number_of_sectors;
00213         unsigned char media_descriptor;
00214     } F_PHY;
00215
00216 /* media descriptor to be set in getphy function */
00217 #define F_MEDIADESC_REMOVABLE 0xf0
00218 #define F_MEDIADESC_FIX 0xf8
00219
00220 /* return bitpattern for driver getphy function */
00221 #define F_ST_MISSING 0x00000001
00222 #define F_ST_CHANGED 0x00000002
00223 #define F_ST_WRPROTECT 0x00000004
00224
00225     /* Driver definitions */
00226     typedef struct F_DRIVER F_DRIVER;
00227
00228     typedef int (*F_WRITESECTOR)(F_DRIVER *driver, void *data, unsigned long sector);
00229     typedef int (*F_WRITEMULTIPLESECTOR)(F_DRIVER *driver, void *data, unsigned long sector, int cnt);
00230     typedef int (*F_READSECTOR)(F_DRIVER *driver, void *data, unsigned long sector);
00231     typedef int (*F_READMULTIPLESECTOR)(F_DRIVER *driver, void *data, unsigned long sector, int cnt);
00232     typedef int (*F_GETPHY)(F_DRIVER *driver, F_PHY *phy);
00233     typedef long (*F_GETSTATUS)(F_DRIVER *driver);
```

```
00234     typedef void (*F_RELEASE)(F_DRIVER *driver);
00235
00236     typedef struct F_DRIVER
00237     {
00238         FN_MUTEX_TYPE mutex; /* mutex for the driver */
00239         int separated;       /* signal if the driver is separated */
00240
00241         unsigned long user_data; /* user defined data */
00242         void *user_ptr;          /* user define pointer */
00243
00244         /* driver functions */
00245         F_WRITESECTOR writesector;
00246         F_WRITEMULTIPLESECTOR writemultiplesector;
00247         F_READSECTOR readsector;
00248         F_READMULTIPLESECTOR readmultiplesector;
00249         F_GETPHY getphy;
00250         F_GETSTATUS getstatus;
00251         F_RELEASE release;
00252     } _F_DRIVER;
00253
00254     typedef F_DRIVER *(*F_DRIVERINIT)(unsigned long driver_param);
00255
00256 /* When initvolume the driver will assign automatically a free driver */
00257 #define F_AUTO_ASSIGN (unsigned long)(-1)
00258
00259     /* definition for a media and f_format */
00260     enum
00261     {
00262         /* 0 */ F_UNKNOWN_MEDIA,
00263         /* 1 */ F_FAT12_MEDIA,
00264         /* 2 */ F_FAT16_MEDIA,
00265         /* 3 */ F_FAT32_MEDIA
00266     };
00267
00268     /* definition for partitions */
00269     typedef struct
00270     {
00271         unsigned long secnum;           /* number of sectors in this partition */
00272         unsigned char system_indicator; /* use F_SYSIND_XX values*/
00273     } F_PARTITION;
00274
00275 /* select system indication for creating partition */
00276 #define F_SYSIND_DOSFAT12 0x01
00277 #define F_SYSIND_DOSFAT16UPTO32MB 0x04
00278 #define F_SYSIND_DOSFAT16OVER32MB 0x06
00279 #define F_SYSIND_DOSFAT32 0x0b
00280
00281 /* these values for extended partition */
00282 #define F_SYSIND_EXTWIN 0x0f
00283 #define F_SYSIND_EXTDOS 0x05
00284
00285     /* definition for f_getfreespace */
00286     typedef struct
00287     {
00288         unsigned long total;
00289         unsigned long free;
00290         unsigned long used;
00291         unsigned long bad;
00292
00293         unsigned long total_high;
00294         unsigned long free_high;
00295         unsigned long used_high;
00296         unsigned long bad_high;
00297     } FN_SPACE;
00298
00299     /* definition for f_stat*/
00300     typedef struct
00301     {
00302         unsigned long filesize;
00303         unsigned short createdate;
00304         unsigned short createtime;
00305         unsigned short modifieddate;
00306         unsigned short modifiedtime;
00307         unsigned short lastaccessdate;
00308         unsigned char attr; /* 00ADVSHR */
00309         int drivenum;
00310     } F_STAT;
00311
00312 /*****************************************************************************
00313  *
00314  * defines for f_findfirst
00315  *
00316  *****************************************************************************/
00317
00318 /* Beginning of file */
00319 #ifdef SEEK_SET
00320 #define FN_SEEK_SET SEEK_SET
```

```
00321 #else
00322 #define FN_SEEK_SET 0
00323 #endif
00324
00325 /* Current position of file pointer */
00326 #ifdef SEEK_CUR
00327 #define FN_SEEK_CUR SEEK_CUR
00328 #else
00329 #define FN_SEEK_CUR 1
00330 #endif
00331
00332 /* End of file */
00333 #ifdef SEEK_END
00334 #define FN_SEEK_END SEEK_END
00335 #else
00336 #define FN_SEEK_END 2
00337 #endif
00338
00339     /****************************************************************************
00340      *
00341      * structure defines
00342      *
00343      ****************************************************************************/
00344
00345 #if (!FN_CAPI_USED)
00346 #define F_FILE FN_FILE
00347 #define F_FIND FN_FIND
00348 #define F_SPACE FN_SPACE
00349 #define F_MAXPATH FN_MAXPATH
00350
00356 #define F_SEEK_SET FN_SEEK_SET
00357 #define F_SEEK_END FN_SEEK_END
00358 #define F_SEEK_CUR FN_SEEK_CUR
00360 #endif
00361
00362     /****************************************************************************
00363      *
00364      * function defines
00365      *
00366      ****************************************************************************/
00367
00368 #if (!FN_CAPI_USED)
00369 #define f_init fn_init
00370 #define f_getversion fm_getversion
00371 #define f_createdriver(driver, driver_init, driver_param) fm_createdriver(driver, driver_init,
      driver_param)
00372 #define f_releasedriver(driver) fm_releasedriver(driver)
00373 #define f_createpartition(driver, parnum, par) fm_createpartition(driver, parnum, par)
00374 #define f_getpartition(driver, parnum, par) fm_getpartition(driver, parnum, par)
00375 #define f_initvolume(drvnumber, driver_init, driver_param) fm_initvolume(drvnumber, driver_init,
      driver_param)
00376 #define f_initvolumepartition(drvnumber, driver, partition) fm_initvolumepartition(drvnumber, driver,
      partition)
00377 #define f_getlasterror fm_getlasterror
00378
00384 #define f_delvolume(drvnumber) fm_delvolume(drvnumber)
00385 #define f_get_volume_count() fm_get_volume_count()
00386 #define f_get_volume_list(buf) fm_get_volume_list(buf)
00387 #define f_checkvolume(drvnumber) fm_checkvolume(drvnumber)
00388 #define f_format(drivenum, fattype) fm_format(drivenum, fattype)
00389 #define f_getcwd(buffer, maxlen) fm_getcwd(buffer, maxlen)
00390 #define f_getdcwd(drivenum, buffer, maxlen) fm_getdcwd(drivenum, buffer, maxlen)
00391 #define f_chdrive(drivenum) fm_chdrive(drivenum)
00392 #define f_getdrive fm_getdrive
00393
00403 #define f_getfreespace(drivenum, pspace) fm_getfreespace(drivenum, pspace)
00404
00416 #define f_chdir(dirname) fm_chdir(dirname)
00417
00429 #define f_mkdir(dirname) fm_mkdir(dirname)
00430
00445 #define f_rmdir(dirname) fm_rmdir(dirname)
00446
00461 #define f_findfirst(filename, find) fm_findfirst(filename, find)
00462
00477 #define f_findnext(find) fm_findnext(find)
00478 #define f_rename(filename, newname) fm_rename(filename, newname)
00479 #define f_move(filename, newname) fm_move(filename, newname)
00480 #define f_filelength(filename) fm_filelength(filename)
00481
00490 #define f_close(filehandle) fm_close(filehandle)
00491 #define f_flush(filehandle) fm_flush(filehandle)
00492
00515 #define f_open(filename, mode) fm_open(filename, mode)
00516 #define f_truncate(filename, length) fm_truncate(filename, length)
00517 #define f_ftruncate(filehandle, length) fm_ftruncate(filehandle, length)
00518
```

```
00529 #define f_read(buf, size, size_st, filehandle) fm_read(buf, size, size_st, filehandle)
00530
00541 #define f_write(buf, size, size_st, filehandle) fm_write(buf, size, size_st, filehandle)
00542
00553 #define f_seek(filehandle, offset, whence) fm_seek(filehandle, offset, whence)
00554 #define f_seteof(filehandle) fm_seteof(filehandle)
00555
00556 #define f_tell(filehandle) fm_tell(filehandle)
00557 #define f_getc(filehandle) fm_getc(filehandle)
00558 #define f_putc(ch, filehandle) fm_putc(ch, filehandle)
00559
00568 #define f_rewind(filehandle) fm_rewind(filehandle)
00569
00578 #define f_eof(filehandle) fm_eof(filehandle)
00579
00580 #define f_stat(filename, stat) fm_stat(filename, stat)
00581
00592 #define f_gettimedate(filename, pctime, pcdate) fm_gettimedate(filename, pctime, pcdate)
00593
00604 #define f_settimedate(filename, ctime, cdate) fm_settimedate(filename, ctime, cdate)
00605
00616 #define f_delete(filename) fm_delete(filename)
00617
00618 #define f_getattr(filename, attr) fm_getattr(filename, attr)
00619 #define f_setattr(filename, attr) fm_setattr(filename, attr)
00620
00621 #define f_getlabel(drivenum, label, len) fm_getlabel(drivenum, label, len)
00622 #define f_setlabel(drivenum, label) fm_setlabel(drivenum, label)
00623
00624 #define f_get_oem(drivenum, str, maxlen) fm_get_oem(drivenum, str, maxlen)
00625 #endif
00626
00627 #ifdef HCC_UNICODE
00628 #if (!FN_CAPI_USED)
00629 #define f_wgetcwd(buffer, maxlen) fm_wgetcwd(buffer, maxlen)
00630 #define f_wgetdcwd(drivenum, buffer, maxlen) fm_wgetdcwd(drivenum, buffer, maxlen)
00631 #define f_wchdir(dirname) fm_wchdir(dirname)
00632 #define f_wmkdir(dirname) fm_wmkdir(dirname)
00633 #define f_wrmdir(dirname) fm_wrmdir(dirname)
00634 #define f_wfindfirst(filename, find) fm_wfindfirst(filename, find)
00635 #define f_wfindnext(find) fm_wfindnext(find)
00636 #define f_wrename(filename, newname) fm_wrename(filename, newname)
00637 #define f_wmove(filename, newname) fm_wmove(filename, newname)
00638 #define f_wfilelength(filename) fm_wfilelength(filename)
00639 #define f_wopen(filename, mode) fm_wopen(filename, mode)
00640 #define f_wtruncate(filename, length) fm_wtruncate(filename, length)
00641 #define f_wstat(filename, stat) fm_wstat(filename, stat)
00642 #define f_wgettimedate(filename, pctime, pcdate) fm_wgettimedate(filename, pctime, pcdate)
00643 #define f_wsettimedate(filename, ctime, cdate) fm_wsettimedate(filename, ctime, cdate)
00644 #define f_wdelete(filename) fm_wdelete(filename)
00645 #define f_wgetattr(filename, attr) fm_wgetattr(filename, attr)
00646 #define f_wsetattr(filename, attr) fm_wsetattr(filename, attr)
00647 #endif
00648 #endif
00649
00650     /*****************************************************************************
00651      *
00652      * function externs
00653      *
00654      *****************************************************************************/
00655
00656     extern int fn_init(void);
00657     extern char *fn_getversion(void);
00658
00659     extern int registerFFILE(F_FILE *file);
00660
00661     extern char *fm_getversion(void);
00662     extern int fm_initvolume(int drvnumber, F_DRIVERINIT driver_init, unsigned long driver_param);
00663     extern int fm_initvolumepartition(int drvnumber, F_DRIVER *driver, int partition);
00664     extern int fm_createpartition(F_DRIVER *driver, int parnum, F_PARTITION *par);
00665     extern int fm_createdriver(F_DRIVER **driver, F_DRIVERINIT driver_init, unsigned long
      driver_param);
00666     extern int fm_releasedriver(F_DRIVER *driver);
00667     extern int fm_getpartition(F_DRIVER *driver, int parnum, F_PARTITION *par);
00668     extern int fm_delvolume(int drvnumber);
00669     extern int fm_checkvolume(int drvnumber);
00670     extern int fm_get_volume_count(void);
00671     extern int fm_get_volume_list(int *buf);
00672     extern int fm_format(int drivenum, long fattype);
00673     extern int fm_getcwd(char *buffer, int maxlen);
00674     extern int fm_getdcwd(int drivenum, char *buffer, int maxlen);
00675     extern int fm_chdrive(int drivenum);
00676     extern int fm_getdrive(void);
00677     extern int fm_getfreespace(int drivenum, FN_SPACE *pspace);
00678     extern int fm_getlasterror(void);
00679
00680     extern int fm_chdir(const char *dirname);
```

```
00681     extern int fm_mkdir(const char *dirname);
00682     extern int fm_rmdir(const char *dirname);
00683
00684     extern int fm_findfirst(const char *filename, FN_FIND *find);
00685     extern int fm_findnext(FN_FIND *find);
00686     extern int fm_rename(const char *filename, const char *newname);
00687     extern int fm_move(const char *filename, const char *newname);
00688     extern long fm_filelength(const char *filename);
00689
00690     extern int fm_close(FN_FILE *filehandle);
00691     extern int fm_flush(FN_FILE *filehandle);
00692     extern FN_FILE *fm_open(const char *filename, const char *mode);
00693     extern FN_FILE *fm_truncate(const char *filename, unsigned long length);
00694     extern int fm_ftruncate(FN_FILE *filehandle, unsigned long length);
00695
00696     extern long fm_read(void *buf, long size, long size_st, FN_FILE *filehandle);
00697     extern long fm_write(const void *buf, long size, long size_st, FN_FILE *filehandle);
00698
00699     extern int fm_seek(FN_FILE *filehandle, long offset, long whence);
00700
00701     extern long fm_tell(FN_FILE *filehandle);
00702     extern int fm_getc(FN_FILE *filehandle);
00703     extern int fm_putc(int ch, FN_FILE *filehandle);
00704     extern int fm_rewind(FN_FILE *filehandle);
00705     extern int fm_eof(FN_FILE *filehandle);
00706     extern int fm_seteof(FN_FILE *filehandle);
00707
00708     extern int fm_stat(const char *filename, F_STAT *stat);
00709     extern int fm_gettimedate(const char *filename, unsigned short *pctime, unsigned short *pcdate);
00710     extern int fm_settimedate(const char *filename, unsigned short ctime, unsigned short cdate);
00711     extern int fm_delete(const char *filename);
00712
00713     extern int fm_getattr(const char *filename, unsigned char *attr);
00714     extern int fm_setattr(const char *filename, unsigned char attr);
00715
00716     extern int fm_getlabel(int drivenum, char *label, long len);
00717     extern int fm_setlabel(int drivenum, const char *label);
00718
00719     extern int fm_get_oem(int drivenum, char *str, long maxlen);
00720
00721 #if (!FN_CAPI_USED)
00722
00739     extern int f_enterFS(void);
00740
00751     extern void f_releaseFS(void);
00752 #endif
00753
00754 #ifdef HCC_UNICODE
00755     extern int fm_wgetcwd(wchar *buffer, int maxlen);
00756     extern int fm_wgetdcwd(int drivenum, wchar *buffer, int maxlen);
00757     extern int fm_wchdir(const wchar *dirname);
00758     extern int fm_wmkdir(const wchar *dirname);
00759     extern int fm_wrmdir(const wchar *dirname);
00760     extern int fm_wfindfirst(const wchar *filename, FN_WFIND *find);
00761     extern int fm_wfindnext(FN_WFIND *find);
00762     extern int fm_wrename(const wchar *filename, const wchar *newname);
00763     extern int fm_wmove(const wchar *filename, const wchar *newname);
00764     extern long fm_wfilelength(const wchar *filename);
00765     extern FN_FILE *fm_wopen(const wchar *filename, const wchar *mode);
00766     extern FN_FILE *fm_wtruncate(const wchar *filename, unsigned long length);
00767     extern int fm_wstat(const wchar *filename, F_STAT *stat);
00768     extern int fm_wgettimedate(const wchar *filename, unsigned short *pctime, unsigned short *pcdate);
00769     extern int fm_wsettimedate(const wchar *filename, unsigned short ctime, unsigned short cdate);
00770     extern int fm_wdelete(const wchar *filename);
00771     extern int fm_wgetattr(const wchar *filename, unsigned char *attr);
00772     extern int fm_wsetattr(const wchar *filename, unsigned char attr);
00773 #endif
00774
00775     /*****************************************************************************
00776      *
00777      * errorcodes
00778      *
00779      *****************************************************************************/
00780
00781 #include "fwerr.h"
00782
00783     /*****************************************************************************
00784      *
00785      * closing bracket for C++ compatibility
00786      *
00787      *****************************************************************************/
00788
00789 #ifdef __cplusplus
00790 }
00791 #endif
00792
00793 /*****************************************************************************
```

```
00794  *
00795  * end of api_f.h
00796  *
00797  ******************************************************************************/
00798
00799 #endif /* _API_F_H_ */
00800
```

24.328 cfc_mcf.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _CFC_IDE_H_
00004 #define _CFC_IDE_H_
00005
00006 /******************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ******************************************************************************/
00033
00034 #include <effs_fat/fat.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041 #define HCC_HW
00042 #define F_CFC_DRIVE0 0
00043 #define F_CFC_DRIVE1 1
00044
00045 /******************************************************************************/
00046 #define CFC_ERR_NOTPLUGGED -1 /* for high level */
00047
00048     enum
00049     {
00050         /* 0 */ CFC_NO_ERROR,
00051         /*101*/ CFC_ERR_BUSY_ATCYL = 101,
00052         /*102*/ CFC_ERR_BUSY_ATDRQ,
00053         /*103*/ CFC_ERR_BUSY_ATCMD,
00054         /*104*/ CFC_ERR_TIMEOUT,
00055         /*105*/ CFC_ERR_STATE,
00056         /*106*/ CFC_ERR_SECCOU,
00057         /*107*/ CFC_ERR_NOTAVAILABLE
00058     };
00059
00060     /*****************************************************************************
00061      *
00062      *  Big endian definitions
00063      *
00064      *****************************************************************************/
00065
00066 #define MOTOuint16_t(x) ((((x) >> 8) & 0x00ff) | (((x) << 8) & 0xff00))
00067
00068     /*****************************************************************************
00069      *
00070      *  Functions
00071      *
00072      *****************************************************************************/
00073
00074     // extern int cfc_initfunc(unsigned long driver_param); /* driver init function */
00075     extern F_DRIVER *cfc_initfunc(unsigned long driver_param); /* driver init function */
```

```
00076 #define CFC_PAGE_SIZE 512                                       /* CFC page size in bytes */
00077
00078 #ifdef __cplusplus
00079 }
00080 #endif
00081
00082 /****************************************************************************
00083  *
00084  * end cfc_ide.h
00085  *
00086  ****************************************************************************/
00087
00088 #endif /* _CFC_IDE_H_ */
```

24.329 chkdsk.h

```
00001 /*NB_REVISION*/
00002
00003 /****************************************************************************
00004  *
00005  *            Copyright (c) 2003-2006 by HCC Embedded
00006  *
00007  * This software is copyrighted by and is the sole property of
00008  * HCC.  All rights, title, ownership, or other interests
00009  * in the software remain the property of HCC.  This
00010  * software may only be used in accordance with the corresponding
00011  * license agreement.  Any unauthorized use, duplication, transmission,
00012  * distribution, or disclosure of this software is expressly forbidden.
00013  *
00014  * This Copyright notice may not be removed or modified without prior
00015  * written consent of HCC.
00016  *
00017  * HCC reserves the right to modify this software without notice.
00018  *
00019  * HCC Embedded
00020  * Budapest 1132
00021  * Victor Hugo Utca 11-15
00022  * Hungary
00023  *
00024  * Tel:  +36 (1) 450 1302
00025  * Fax:  +36 (1) 450 1303
00026  * http: www.hcc-embedded.com
00027  * email: info@hcc-embedded.com
00028  *
00029  ****************************************************************************/
00030
00031 #ifndef __CHKDSK_H
00032 #define __CHKDSK_H
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039 #define CHKDSK_LOG_ENABLE
00040 #ifdef CHKDSK_LOG_ENABLE
00041 #define CHKDSK_LOG_SIZE 8192
00042 #endif
00043 #define CHKDSK_MAX_DIR_DEPTH 64 /* max. stack= ~(CHKDSK_MAX_DIR_DEPTH*85)+1100 */
00044
00045 #define CHKDSK_ERASE_BAD_CHAIN 0x1      /* erase all bad chains */
00046 #define CHKDSK_ERASE_LOST_CHAIN 0x2     /* erase all lost chains */
00047 #define CHKDSK_ERASE_LOST_BAD_CHAIN 0x4 /* erase all lost bad chains */
00048
00049     enum
00050     {
00051         FC_NO_ERROR,
00052         FC_WRITE_ERROR = 50,
00053         FC_READ_ERROR,
00054         FC_CLUSTER_ERROR,
00055         FC_ALLOCATION_ERROR
00056     };
00057
00058     extern int f_checkdisk(int drivenum, int param);
00059
00060 #ifdef __cplusplus
00061 };
00062 #endif
00063
00064 #endif
```

24.330 common.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _COMMON_H_
00004 #define _COMMON_H_
00005
00006 /******************************************************************************
00007  *
00008  *            Copyright (c) 2003-2006 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ******************************************************************************/
00033
00034 #include <effs_fat/fat.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041 /* retry counter for read and write */
00042 #define RDWR_RETRY 3
00043
00044     /* EXPORTS */
00045
00046     extern unsigned long drvbldnum;
00047     extern FN_MUTEX_TYPE fat_gmutex;
00048
00049     extern int fn_setlasterror(F_MULTI *fm, int errorcode);
00050     extern void fn_setlasterror_noret(F_MULTI *fm, int errorcode);
00051     extern int fn_createdriver(F_MULTI *fm, F_DRIVER **driver, F_DRIVERINIT driver_init, unsigned long
    driver_param);
00052     extern int fn_releasedriver(F_MULTI *fm, F_DRIVER *driver);
00053     extern int fn_createpartition(F_MULTI *fm, F_DRIVER *driver, int parnum, const F_PARTITION *par);
00054     extern int fn_initvolume(F_MULTI *fm, int drvnumber, F_DRIVERINIT driver_init, unsigned long
    driver_param);
00055     extern int fn_initvolumepartition(F_MULTI *fm, int drvnumber, F_DRIVER *driver, int partition);
00056     extern int fn_ftruncate(F_MULTI *fm, FN_FILE *filehandle, unsigned long length);
00057     extern int fn_getpartition(F_DRIVER *driver, int parnum, F_PARTITION *par);
00058
00059     extern int _f_readsector(F_VOLUME *vi, void *data, unsigned long sector, int cnt);
00060     extern int _f_writesector(F_VOLUME *vi, void *data, unsigned long sector, int cnt);
00061     extern int _f_getclustervalue(F_VOLUME *vi, unsigned long cluster, unsigned long *pvalue);
00062     extern void _f_clustertopos(const F_VOLUME *vi, unsigned long cluster, F_POS *pos);
00063     extern unsigned long _f_getdecluster(const F_VOLUME *vi, F_DIRENTRY *de);
00064     extern int _f_checkstatus(const F_VOLUME *vi, long *pstatus);
00065     extern int _f_getvolume(F_MULTI *fm, int drivenum, F_VOLUME **pvi);
00066     extern unsigned short _f_get16bitl(void *ptr);
00067     extern unsigned long _f_get32bitl(void *ptr);
00068     extern void _f_set16bitl(void *ptr, unsigned short num);
00069     extern void _f_set32bitl(void *ptr, unsigned long num);
00070     extern int _f_checklocked(long drvnum, const F_POS *pos);
00071     extern int _f_checkreadlocked(F_VOLUME *vi, long drvnum, F_POS *pos, FN_FILEINT **fapp);
00072     extern int _f_checkappendlocked(long drvnum, const F_POS *pos, FN_FILEINT *ofile);
00073     extern void _f_initentry(F_DIRENTRY *de, const char *name, const char *ext);
00074     extern int _f_alloccluster(F_VOLUME *vi, unsigned long *pcluster);
00075     extern int _f_dobadblock(F_VOLUME *vi, FN_FILEINT *f);
00076     extern int _f_setclustervalue(F_VOLUME *vi, unsigned long cluster, unsigned long data);
00077     extern int _f_writefatsector(F_VOLUME *vi);
00078     extern void _f_setdecluster(const F_VOLUME *vi, F_DIRENTRY *de, unsigned long cluster);
00079     extern int _f_fseek(F_VOLUME *vi, F_MULTI *fm, FN_FILEINT *f, unsigned long offset);
00080     extern int _f_getcurrsector(F_VOLUME *vi, FN_FILEINT *f, char *ptr, unsigned int *cnt);
00081     extern int _f_removechain(F_VOLUME *vi, unsigned long cluster);
00082     extern FN_FILEINT *_f_check_handle(FN_FILE *filehandle);
00083     extern int _f_writedirsector(F_VOLUME *vi);
```

```
00084     extern unsigned long _f_getmaxcluster(const F_VOLUME *vi);
00085     extern int _f_writezeros(F_VOLUME *vi, F_MULTI *fm, FN_FILEINT *f, unsigned long num);
00086     extern void _f_syncfiles(F_VOLUME *vi, F_MULTI *fm, const FN_FILEINT *file);
00087     int _f_findopensize(unsigned long *ofsize, int drivenum, F_POS *pos);
00088
00089 #if F_MAXSEEKPOS
00090     extern void _fn_removeseekpos(FN_FILEINT *f);
00091     extern void _fn_updateseekpos(FN_FILEINT *f);
00092     extern void _fn_initseekdivisor(FN_FILEINT *f, F_VOLUME *vi);
00093 #endif
00094
00095 #ifdef FATCACHE_ENABLE
00096     extern int _f_fatcache_flush(F_VOLUME *vi, int clear);
00097 #endif
00098
00099     extern int _f_mutex_get(F_MULTI *fm, F_VOLUME *vi);
00100     extern void _f_mutex_put(F_MULTI *fm);
00101
00102 #if (!FN_CAPI_USED)
00103 #ifdef USE_MALLOC
00104     extern F_MULTI *g_multi[FN_MAXTASK];
00105 #else
00106     extern F_MULTI g_multi[FN_MAXTASK];
00107 #endif
00108 #endif
00109
00110     extern int fnGetTask(F_MULTI **fm);
00111
00112 #ifdef INTERNAL_MEMFN
00113     extern void *_f_memset(void *, int, unsigned long);
00114     extern void *_f_memcpy(void *, void *, unsigned long);
00115 #endif
00116
00117 #ifdef __cplusplus
00118 }
00119 #endif
00120
00121 /****************************************************************************
00122  *
00123  * end of common.h
00124  *
00125  ****************************************************************************/
00126
00127 #endif /* _COMMON_H_ */
```

24.331 debug.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _DEBUG_H_
00004 #define _DEBUG_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2006 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 /* set this define to 1 if program is running on PC and debug file is required */
00035 #if 0
00036
00037 #include <stdio.h>
00038
```

```
00039 #ifdef _HCC_COMMON_C_
00040 FILE *debfile=0;
00041 #else
00042 extern FILE *debfile;
00043 #endif
00044
00045 #define DEBOPEN \
00046     if (!debfile) debfile = fopen("C:/fattest.txt", "wt+");
00047 #define DEBPR0(s) fprintf(debfile, s);
00048 #define DEBPR1(s, p1) fprintf(debfile, s, p1);
00049 #define DEBPR2(s, p1, p2) fprintf(debfile, s, p1, p2);
00050 #define DEBPR3(s, p1, p2, p3) fprintf(debfile, s, p1, p2, p3);
00051
00052 #else
00053
00054 #define DEBOPEN
00055 #define DEBPR0(s)
00056 #define DEBPR1(s, p1)
00057 #define DEBPR2(s, p1, p2)
00058 #define DEBPR3(s, p1, p2, p3)
00059
00060 #endif
00061
00062 /*****************************************************************************
00063  *
00064  * end of debug.h
00065  *
00066  *****************************************************************************/
00067
00068 #endif /* _DEBUG_H_ */
```

24.332 effs_utils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-------------------------------------------------------------------
00006  EFFS, Embedded FAT File System Utilities
00007  -------------------------------------------------------------------*/
00008
00009 #ifndef _EFFS_UTILS_H_
00010 #define _EFFS_UTILS_H_
00011
00012 #include <effs_fat/fat.h>
00013
00014 #ifdef __cplusplus
00015 extern "C"
00016 {
00017 #endif
00018
00019     /*-------------------------------------------------------------------
00020      fgets()
00021
00022       Description
00023       Reads at most len-1 characters from fp until a newline is found.
00024       The characters including the newline (0x0A) are stored in buf.
00025       The buffer is terminated with a NULL.
00026
00027       Returns
00028       fgets returns the buffer passed to it, with the data filled in.
00029       If end of file occurs with some data already accumulated, the
00030       data is returned with no other indication. If no data are read,
00031       NULL is returned instead.
00032
00033       Note that fgets returns all of the data, while gets removes the
00034       trailing newline (with no indication that it has done so.)
00035      -------------------------------------------------------------------*/
00036     extern char *f_fgets(char *buf, int len, F_FILE *fp);
00037
00038     /*-------------------------------------------------------------------
00039      int fputs(const char *s, FILE *fp);
00040
00041      Description:
00042      fputs writes the string at s (but without the trailing null) to
00043      the specified file, fp.
00044
00045      Returns:
00046       If successful, the result is 0; otherwise, the result is EOF.
00047
00048      Portability
00049       ANSII C requires fputs, but does not specify that the result
00050       on success must be 0; any non-negative value is permitted.
00051      -------------------------------------------------------------------*/
00052     extern int f_fputs(const char *s, F_FILE *fp);
```

```
00053
00054     /*-------------------------------------------------------------------
00055      EFFS fprintf implementation
00056      -------------------------------------------------------------------*/
00057     extern int f_fprintf(F_FILE *fp, const char *format, ...);
00058
00059 #ifdef __cplusplus
00060 }
00061 #endif
00062
00063 #endif /* _EFFS_UTILS_H_ */
```

24.333 fat.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FAT_H_
00004 #define _FAT_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003-2006 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 //#ifdef __cplusplus
00035 // extern "C" {
00036 //#endif
00037
00038 #include <effs_fat/api_f.h>
00039 #include <effs_fat/port_f.h>
00040 #include <effs_fat/udefs_f.h>
00041
00042 #define NOR_DRV_NUM 0
00043 #define STDRAM_DRV_NUM 1
00044 #define MMC_DRV_NUM 2
00045 #define CFC_DRV_NUM 3
00046 #define HDD_DRV_NUM 3
00047 #define FATRAM_DRV_NUM 4
00048 #define ONBOARD_MMC_DRV_NUM 5
00049
00067 #define f_mountfat f_initvolume
00068
00069 #define F_SECTOR_SIZE 512
00070
00071 typedef struct
00072 {
00073     unsigned char jump_code[3];
00074     unsigned char OEM_name[8];
00075     unsigned short bytes_per_sector;
00076     unsigned char sector_per_cluster;
00077     unsigned short reserved_sectors;
00078     unsigned char number_of_FATs;
00079     unsigned short max_root_entry;
00080     unsigned short number_of_sectors_less32; /*  <32M  */
00081     unsigned char media_descriptor;
00082     unsigned short sector_per_FAT;
00083     unsigned short sector_per_Track;
00084     unsigned short number_of_heads;
00085     unsigned long number_of_hidden_sectors;
00086     unsigned long number_of_sectors;
00087
00088     /* only on fat32 */
```

```
00089     unsigned long sector_per_FAT32;
00090     unsigned short extflags;
00091     unsigned short fsversion;
00092     unsigned long rootcluster;
00093     unsigned short fsinfo;
00094     unsigned short bkbootsec;
00095     unsigned char reserved[12];
00096
00097     /* fat12-fat16-fat32 */
00098     unsigned short logical_drive_num;
00099     unsigned char extended_signature;
00100     unsigned long serial_number;
00101     unsigned char volume_name[11];
00102     unsigned char FAT_name[8];
00103     unsigned char executable_marker[2];
00104 } F_BOOTRECORD;
00105
00106 /* number of sectors after mbr */
00107 #define F_SPACE_AFTER_MBR 63
00108
00109 /* media descriptor to be set in getphy function */
00110 //#define   F_MEDIADESC_REMOVABLE   0xf0
00111 //#define   F_MEDIADESC_FIX         0xf8
00112
00113 #if F_LONGFILENAME
00114
00115 typedef struct
00116 {
00117     W_CHAR name[261]; /* with zero term */
00118     unsigned char ord;
00119     unsigned char chksum;
00120     unsigned char state;
00121     unsigned long start;
00122     unsigned long end;
00123 } F_LFNINT;
00124
00125 enum
00126 {
00127     /* 0 */ F_LFNSTATE_LFN,  /* lfn is useable */
00128     /* 1 */ F_LFNSTATE_SFN,  /* lfn is useable, contains short filename */
00129     /* 2 */ F_LFNSTATE_NEXT, /* lfn need more entry */
00130     /* 3 */ F_LFNSTATE_INV   /* invalid lfn */
00131 };
00132
00133 #endif /* F_LONGFILENAME */
00134
00135 typedef struct
00136 {
00137     char name[F_MAXNAME]; /* 8+3 filename */
00138     char ext[F_MAXEXT];   /* 8+3 extension */
00139     unsigned char attr;   /* 00ADVSHR */
00140
00141     unsigned char ntres;
00142     unsigned char crttimetenth;
00143     unsigned char crttime[2];
00144     unsigned char crtdate[2];
00145     unsigned char lastaccessdate[2];
00146
00147     unsigned char clusterhi[2]; /* FAT32 only */
00148     unsigned char ctime[2];
00149     unsigned char cdate[2];
00150     unsigned char clusterlo[2]; /* fat12,fat16,fat32 */
00151     unsigned char filesize[4];
00152 } F_DIRENTRY;
00153
00154 /* 1st char in 8+3 if entry is deleted*/
00155 #define F_DELETED_CHAR ((char)0xe5)
00156
00157 /* lower case name */
00158 #define NTRES_LOW 0x08
00159
00160 /* define for long filename entry in directory entry*/
00161 #define F_ATTR_LFN (F_ATTR_VOLUME | F_ATTR_SYSTEM | F_ATTR_HIDDEN | F_ATTR_READONLY)
00162
00163 #if F_LONGFILENAME
00164
00165 typedef struct
00166 {
00167     unsigned char ord;
00168
00169     unsigned char lfn_1;
00170     unsigned char lfnhi_1;
00171
00172     unsigned char lfn_2;
00173     unsigned char lfnhi_2;
00174
00175     unsigned char lfn_3;
```

```
00176     unsigned char lfnhi_3;
00177
00178     unsigned char lfn_4;
00179     unsigned char lfnhi_4;
00180
00181     unsigned char lfn_5;
00182     unsigned char lfnhi_5;
00183
00184     unsigned char attr; /* 00ADVSHR */
00185     unsigned char type; /* always 0 */
00186
00187     unsigned char chksum;
00188
00189     unsigned char lfn_6;
00190     unsigned char lfnhi_6;
00191
00192     unsigned char lfn_7;
00193     unsigned char lfnhi_7;
00194
00195     unsigned char lfn_8;
00196     unsigned char lfnhi_8;
00197
00198     unsigned char lfn_9;
00199     unsigned char lfnhi_9;
00200
00201     unsigned char lfn_10;
00202     unsigned char lfnhi_10;
00203
00204     unsigned char lfn_11;
00205     unsigned char lfnhi_11;
00206
00207     unsigned char clusterlo[2]; /* fat12,fat16,fat32 */
00208
00209     unsigned char lfn_12;
00210     unsigned char lfnhi_12;
00211
00212     unsigned char lfn_13;
00213     unsigned char lfnhi_13;
00214
00215 } F_LFN;
00216
00217 #endif /* F_LONGFILENAME  */
00218
00219 /* definitions for FAT entry */
00220 #define F_CLUSTER_FREE ((unsigned long)0x00000000)
00221 #define F_CLUSTER_RESERVED ((unsigned long)0x0ffffff0)
00222 #define F_CLUSTER_BAD ((unsigned long)0x0ffffff7)
00223 #define F_CLUSTER_LAST ((unsigned long)0x0ffffff8)
00224 #define F_CLUSTER_LASTF32R ((unsigned long)0x0fffffff)
00225
00226 typedef struct
00227 {
00228     unsigned long sector; /* start sector */
00229     unsigned long num;    /* number of sectors */
00230 } F_SECTOR;
00231
00232 #if F_MAXFILES > 0xffff /* maximum number of files */
00233 #error F_MAXFILES should be less than 65535
00234 #elif F_MAXFILES > 0x7fff
00235 #define F_MAXFILES_SHIFT 16
00236 #elif F_MAXFILES > 0x3fff
00237 #define F_MAXFILES_SHIFT 15
00238 #elif F_MAXFILES > 0x1fff
00239 #define F_MAXFILES_SHIFT 14
00240 #elif F_MAXFILES > 0x0fff
00241 #define F_MAXFILES_SHIFT 13
00242 #elif F_MAXFILES > 0x07ff
00243 #define F_MAXFILES_SHIFT 12
00244 #elif F_MAXFILES > 0x03ff
00245 #define F_MAXFILES_SHIFT 11
00246 #elif F_MAXFILES > 0x01ff
00247 #define F_MAXFILES_SHIFT 10
00248 #elif F_MAXFILES > 0x00ff
00249 #define F_MAXFILES_SHIFT 9
00250 #elif F_MAXFILES > 0x007f
00251 #define F_MAXFILES_SHIFT 8
00252 #elif F_MAXFILES > 0x003f
00253 #define F_MAXFILES_SHIFT 7
00254 #elif F_MAXFILES > 0x001f
00255 #define F_MAXFILES_SHIFT 6
00256 #elif F_MAXFILES > 0x000f
00257 #define F_MAXFILES_SHIFT 5
00258 #else
00259 #define F_MAXFILES_SHIFT 4
00260 #endif
00261
00262 /*  definitions for FN_FILE internally used  */
```

```
00263
00264 typedef struct
00265 {
00266     int N;
00267     char *ptr;
00268 #ifdef USE_MALLOC
00269     F_POS *pos;
00270 #else
00271     F_POS *pos;
00272     F_POS posbuf[WR_DATACACHE_SIZE];
00273 #endif
00274 } t_WrDataCache;
00275
00276 typedef struct FN_FILEINT FN_FILEINT;
00277
00278 typedef struct FN_FILEINT
00279 {
00280     FN_FILE file;
00281     long modified;
00282     int drivenum; /*  0-A 1-B 2-C  */
00283     unsigned long abspos;
00284     unsigned long relpos;
00285     unsigned long filesize;
00286     unsigned char data[F_SECTOR_SIZE];
00287     int datawritten;
00288     t_WrDataCache WrDataCache;
00289     unsigned long startcluster;
00290     F_POS pos;
00291     F_POS dirpos;
00292     long state;
00293 #if F_MAXSEEKPOS
00294     long seekpos[F_MAXSEEKPOS];
00295     long seekprev[F_MAXSEEKPOS];
00296     long seekshift;
00297 #endif
00298     FN_FILEINT *syncfile;
00299     char mode;
00300     char dummy[3];
00301 } _FN_FILEINT;
00302
00303 /* this bit signal if synchronization is required in append and read in state */
00304 #define F_FILE_ST_SYNC 0x0001
00305 #define F_FILE_ST_EOF 0x0002
00306
00307 typedef struct
00308 {
00309     unsigned long clfree;
00310     unsigned long clused;
00311     unsigned long clbad;
00312 } F_CLSPACE;
00313
00314 #ifdef FATCACHE_ENABLE
00315 typedef struct
00316 {
00317     unsigned long sector;
00318     int modified;
00319 } t_CacheDsc;
00320
00321 typedef struct
00322 {
00323     int N;
00324     t_CacheDsc *dsc;
00325 #ifdef USE_MALLOC
00326     unsigned char *data;
00327 #else
00328     t_CacheDsc dsc_array[FATCACHE_SIZE];
00329     unsigned char data[FATCACHE_SIZE * F_SECTOR_SIZE];
00330 #endif
00331 } t_FatCache;
00332 #endif
00333
00334 typedef struct
00335 {
00336     long state;
00337
00338     F_BOOTRECORD bootrecord;
00339
00340     F_SECTOR firstfat;
00341
00342     F_SECTOR root;
00343     F_SECTOR data;
00344
00345 #ifdef FATCACHE_ENABLE
00346     t_FatCache fatcache;
00347     unsigned char *fat;
00348 #else
00349     unsigned char fat[F_SECTOR_SIZE];
```

```
00350 #endif
00351     unsigned long fatsector;
00352     long fatmodified;
00353
00354 #if F_LONGFILENAME
00355 #ifdef DIRCACHE_ENABLE
00356 #ifdef USE_MALLOC
00357     unsigned char *dircache;
00358 #else
00359     unsigned char dircache[DIRCACHE_SIZE * F_SECTOR_SIZE];
00360 #endif
00361     unsigned long dircache_start;
00362     unsigned long dircache_size;
00363 #endif
00364 #endif
00365     unsigned char direntry[F_SECTOR_SIZE];
00366     unsigned long direntrysector;
00367
00368     unsigned long lastalloccluster;
00369
00370     W_CHAR *cwd;
00371
00372     long mediatype;
00373     F_CLSPACE clspace; /* calculated disk space */
00374     char cspaceok;
00375 #if defined FATBITFIELD_ENABLE && defined USE_MALLOC
00376     unsigned char *fatbitfield;
00377 #endif
00378     int partition;
00379     unsigned long sectorstart;
00380     unsigned long sectornum;
00381     F_PHY phy;
00382
00383     F_DRIVER *driver;
00384
00385     unsigned char sectorbuffer[F_SECTOR_SIZE];
00386
00387 } F_VOLUME;
00388
00389 #define F_FAT12_MAX_CLUSTER 0xFF0
00390 #define F_FAT16_MAX_CLUSTER 0xFFF0
00391
00392 typedef struct
00393 {
00394     F_VOLUME volumes[FN_MAXVOLUME]; /* volumes */
00395     FN_FILEINT files[F_MAXFILES];
00396     unsigned long drvbldnum; /* drive build number for file.reference */
00397 } FN_FILESYSTEM;
00398
00399 extern FN_FILESYSTEM f_filesystem;
00400
00401 #if (!FN_CAPI_USED)
00402 typedef struct
00403 {
00404     long ID;        /* task id */
00405     int f_curdrive; /* current drive */
00406
00407     struct
00408     {
00409         W_CHAR cwd[FN_MAXPATH]; /* current working folders in this volume */
00410     } f_vols[FN_MAXVOLUME];
00411
00412     FN_MUTEX_TYPE *pmutex;
00413     unsigned char current_bank;
00414     int lasterror; /* last error in this task */
00415 } F_MULTI;
00416 #endif
00417
00418 /*  current file opening modes */
00419
00420 enum
00421 {
00422     /*  0 */ FN_FILE_CLOSE,
00423     /*  1 */ FN_FILE_RD,
00424     /*  2 */ FN_FILE_WR,
00425     /*  3 */ FN_FILE_A,
00426     /*  4 */ FN_FILE_RDP,
00427     /*  5 */ FN_FILE_WRP,
00428     /*  6 */ FN_FILE_AP,
00429     /*  7 */ FN_FILE_WRERR,
00430     /*  8 */ FN_FILE_LOCKED
00431 };
00432
00433 #define FN_FILE_ABORT_FLAG 0x40 /* signal for file is aborted */
00434
00435 /*  current drive modes */
00436
```

```
00437 enum
00438 {
00439     /*  0 */ F_STATE_NONE,
00440     /*  1 */ F_STATE_NEEDMOUNT,
00441     /*  2 */ F_STATE_WORKING
00442 };
00443
00444 /****************************************************************************
00445  *
00446  * externed functions
00447  *
00448  ****************************************************************************/
00449
00450 // extern int fn_init(void);
00451 // extern char *fn_getversion(void);
00452 extern int fn_delvolume(F_MULTI *fm, int drvnumber);
00453 extern int fn_get_volume_count(F_MULTI *fm);
00454 extern int fn_get_volume_list(F_MULTI *fm, int *buf);
00455 extern int fn_checkvolume(F_MULTI *fm, int drvnumber);
00456 extern int fn_format(F_MULTI *fm, int drivenum, long fattype);
00457 extern int fn_getcwd(F_MULTI *fm, char *buffer, int maxlen);
00458 extern int fn_getdcwd(F_MULTI *fm, int drivenum, char *buffer, int maxlen);
00459 extern int fn_chdrive(F_MULTI *fm, int drivenum);
00460 extern int fn_getdrive(F_MULTI *fm);
00461 extern int fn_getfreespace(F_MULTI *fm, int drivenum, FN_SPACE *pspace);
00462
00463 extern int fn_chdir(F_MULTI *fm, const char *dirname);
00464 extern int fn_mkdir(F_MULTI *fm, const char *dirname);
00465 extern int fn_rmdir(F_MULTI *fm, const char *dirname);
00466
00467 extern int fn_findfirst(F_MULTI *fm, const char *filename, FN_FIND *find);
00468 extern int fn_findnext(F_MULTI *fm, FN_FIND *find);
00469 extern int fn_rename(F_MULTI *fm, const char *filename, const char *newname);
00470 extern int fn_move(F_MULTI *fm, const char *filename, const char *newname);
00471 extern long fn_filelength(F_MULTI *fm, const char *filename);
00472
00473 extern int fn_close(F_MULTI *fm, FN_FILE *filehandle);
00474 extern int fn_flush(F_MULTI *fm, FN_FILE *file);
00475 extern FN_FILE *fn_open(F_MULTI *fm, const char *filename, const char *mode);
00476 extern FN_FILE *fn_truncate(F_MULTI *fm, const char *filename, unsigned long length);
00477
00478 extern long fn_read(F_MULTI *fm, void *buf, long size, long size_st, FN_FILE *filehandle);
00479 extern long fn_write(F_MULTI *fm, const void *buf, long size, long size_st, FN_FILE *filehandle);
00480
00481 extern int fn_seek(F_MULTI *fm, FN_FILE *filehandle, long offset, long whence);
00482 extern int fn_seteof(F_MULTI *fm, FN_FILE *filehandle);
00483
00484 extern long fn_tell(F_MULTI *fm, FN_FILE *filehandle);
00485 extern int fn_getc(F_MULTI *fm, FN_FILE *filehandle);
00486 extern int fn_putc(F_MULTI *fm, int ch, FN_FILE *filehandle);
00487 extern int fn_rewind(F_MULTI *fm, FN_FILE *filehandle);
00488 extern int fn_eof(F_MULTI *, FN_FILE *filehandle);
00489
00490 extern int fn_gettimedate(F_MULTI *fm, const char *filename, unsigned short *pctime, unsigned short
      *pcdate);
00491 extern int fn_settimedate(F_MULTI *fm, const char *filename, unsigned short ctime, unsigned short
      cdate);
00492 extern int fn_delete(F_MULTI *fm, const char *filename);
00493 extern int fn_stat(F_MULTI *fm, const char *filename, F_STAT *stat);
00494
00495 extern int fn_getattr(F_MULTI *fm, const char *filename, unsigned char *attr);
00496 extern int fn_setattr(F_MULTI *fm, const char *filename, unsigned char attr);
00497
00498 extern int fn_getlabel(F_MULTI *fm, int drivenum, char *label, long len);
00499 extern int fn_setlabel(F_MULTI *fm, int drivenum, const char *label);
00500
00501 extern int fn_get_oem(F_MULTI *fm, int drivenum, char *str, long maxlen);
00502
00503 #ifdef HCC_UNICODE
00504 extern int fn_wgetcwd(F_MULTI *fm, wchar *buffer, int maxlen);
00505 extern int fn_wgetdcwd(F_MULTI *fm, int drivenum, wchar *buffer, int maxlen);
00506 extern int fn_wchdir(F_MULTI *fm, const wchar *dirname);
00507 extern int fn_wmkdir(F_MULTI *fm, const wchar *dirname);
00508 extern int fn_wrmdir(F_MULTI *fm, const wchar *dirname);
00509 extern int fn_wfindfirst(F_MULTI *fm, const wchar *filename, FN_WFIND *find);
00510 extern int fn_wfindnext(F_MULTI *fm, FN_WFIND *find);
00511 extern int fn_wrename(F_MULTI *fm, const wchar *filename, const wchar *newname);
00512 extern int fn_wmove(F_MULTI *fm, const wchar *filename, const wchar *newname);
00513 extern long fn_wfilelength(F_MULTI *fm, const wchar *filename);
00514 extern FN_FILE *fn_wopen(F_MULTI *fm, const wchar *filename, const wchar *mode);
00515 extern FN_FILE *fn_wtruncate(F_MULTI *fm, const wchar *filename, unsigned long length);
00516 extern int fn_wstat(F_MULTI *fm, const wchar *filename, F_STAT *stat);
00517 extern int fn_wgettimedate(F_MULTI *fm, const wchar *filename, unsigned short *pctime, unsigned short
      *pcdate);
00518 extern int fn_wsettimedate(F_MULTI *fm, const wchar *filename, unsigned short ctime, unsigned short
      cdate);
00519 extern int fn_wdelete(F_MULTI *fm, const wchar *filename);
```

```
00520 extern int fn_wgetattr(F_MULTI *fm, const wchar *filename, unsigned char *attr);
00521 extern int fn_wsetattr(F_MULTI *fm, const wchar *filename, unsigned char attr);
00522 #endif
00523
00524 #include "fat_m.h"
00525
00526 #define _f_toupper(ch) (((ch) >= 'a' && (ch) <= 'z') ? ((ch) - 'a' + 'A') : (ch))
00527
00528 #ifdef HCC_UNICODE
00529 extern wchar *_towchar(wchar *nconv, const char *s);
00530 #endif
00531 extern int _f_addentry(F_VOLUME *vi, F_NAME *fsname, F_POS *pos, F_DIRENTRY **pde);
00532 extern int _f_getdirsector(F_VOLUME *vi, unsigned long sector);
00533
00534 //#ifdef __cplusplus
00535 //}
00536 //#endif
00537
00538 /****************************************************************************
00539  *
00540  * end of fat.h
00541  *
00542  ****************************************************************************/
00543
00544 #endif /* _FAT_H_ */
```

24.334 fat_m.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FAT_M_H_
00004 #define _FAT_M_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039 #ifdef HCC_UNICODE
00040     extern int fm_wgetcwd(wchar *buffer, int maxlen);
00041     extern int fm_wgetdcwd(int drivenum, wchar *buffer, int maxlen);
00042     extern int fm_wchdir(const wchar *dirname);
00043     extern int fm_wmkdir(const wchar *dirname);
00044     extern int fm_wrmdir(const wchar *dirname);
00045     extern int fm_wfindfirst(const wchar *filename, FN_WFIND *find);
00046     extern int fm_wfindnext(FN_WFIND *find);
00047     extern int fm_wrename(const wchar *filename, const wchar *newname);
00048     extern int fm_wmove(const wchar *filename, const wchar *newname);
00049     extern long fm_wfilelength(const wchar *filename);
00050     extern FN_FILE *fm_wopen(const wchar *filename, const wchar *mode);
00051     extern FN_FILE *fm_wtruncate(const wchar *filename, unsigned long length);
00052     extern int fm_wstat(const wchar *filename, F_STAT *stat);
00053     extern int fm_wgettimedate(const wchar *filename, unsigned short *pctime, unsigned short *pcdate);
00054     extern int fm_wsettimedate(const wchar *filename, unsigned short ctime, unsigned short cdate);
00055     extern int fm_wdelete(const wchar *filename);
00056     extern int fm_wgetattr(const wchar *filename, unsigned char *attr);
00057     extern int fm_wsetattr(const wchar *filename, unsigned char attr);
```

```
00058 #endif
00059
00060 #ifdef __cplusplus
00061 }
00062 #endif
00063
00064 /****************************************************************************
00065  *
00066  * end of fat_m.h
00067  *
00068  ****************************************************************************/
00069
00070 #endif /* _FAT_M_H_ */
```

24.335 effs_fat/fwerr.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FWERR_H_
00004 #define _FWERR_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039     /****************************************************************************
00040      *
00041      * Filesystem's errorcodes
00042      *
00043      ****************************************************************************/
00044
00045     enum
00046     {
00047         /*  0 */ F_NO_ERROR,
00048         /*  1 */ F_ERR_INVALIDDRIVE,
00049         /*  2 */ F_ERR_NOTFORMATTED,
00050         /*  3 */ F_ERR_INVALIDDIR,
00051         /*  4 */ F_ERR_INVALIDNAME,
00052         /*  5 */ F_ERR_NOTFOUND,
00053         /*  6 */ F_ERR_DUPLICATED,
00054         /*  7 */ F_ERR_NOMOREENTRY,
00055         /*  8 */ F_ERR_NOTOPEN,
00056         /*  9 */ F_ERR_EOF,
00057         /* 10 */ F_ERR_RESERVED,
00058         /* 11 */ F_ERR_NOTUSEABLE,
00059         /* 12 */ F_ERR_LOCKED,
00060         /* 13 */ F_ERR_ACCESSDENIED,
00061         /* 14 */ F_ERR_NOTEMPTY,
00062         /* 15 */ F_ERR_INITFUNC,
00063         /* 16 */ F_ERR_CARDREMOVED,
00064         /* 17 */ F_ERR_ONDRIVE,
00065         /* 18 */ F_ERR_INVALIDSECTOR,
00066         /* 19 */ F_ERR_READ,
00067         /* 20 */ F_ERR_WRITE,
00068         /* 21 */ F_ERR_INVALIDMEDIA,
00069         /* 22 */ F_ERR_BUSY,
```

```
00070         /* 23 */ F_ERR_WRITEPROTECT,
00071         /* 24 */ F_ERR_INVFATTYPE,
00072         /* 25 */ F_ERR_MEDIATOOSMALL,
00073         /* 26 */ F_ERR_MEDIATOOLARGE,
00074         /* 27 */ F_ERR_NOTSUPPSECTORSIZE,
00075         /* 28 */ F_ERR_UNKNOWN,
00076         /* 29 */ F_ERR_DRVALREADYMNT,
00077         /* 30 */ F_ERR_TOOint32_tNAME,
00078         /* 31 */ F_ERR_NOTFORREAD,
00079         /* 32 */ F_ERR_DELFUNC,
00080         /* 33 */ F_ERR_ALLOCATION,
00081         /* 34 */ F_ERR_INVALIDPOS,
00082         /* 35 */ F_ERR_NOMORETASK,
00083         /* 36 */ F_ERR_NOTAVAILABLE,
00084         /* 37 */ F_ERR_TASKNOTFOUND,
00085         /* 38 */ F_ERR_UNUSABLE
00086     };
00087
00088 #ifdef __cplusplus
00089 }
00090 #endif
00091
00092 /****************************************************************************
00093  *
00094  * end of fwerr.h
00095  *
00096  ****************************************************************************/
00097
00098 #endif /* _FWERR_H_ */
```

24.336 file/fwerr.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FW_ERR_H_
00004 #define _FW_ERR_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039     /****************************************************************************
00040      *
00041      * FW errorcodes
00042      *
00043      ****************************************************************************/
00044
00045     enum
00046     {
00047         /*  0 */ FW_NO_ERROR,
00048         /*  1 */ FW_ERR_INVALIDDRIVE,
00049         /*  2 */ FW_ERR_NOTFORMATTED,
00050         /*  3 */ FW_ERR_INVALIDDIR,
00051         /*  4 */ FW_ERR_INVALIDNAME,
00052         /*  5 */ FW_ERR_NOTFOUND,
00053         /*  6 */ FW_ERR_DUPLICATED,
```

```
00054         /*  7 */ FW_ERR_NOMOREENTRY,
00055         /*  8 */ FW_ERR_NOTOPEN,
00056         /*  9 */ FW_ERR_EOF,
00057         /* 10 */ FW_ERR_RESERVED,
00058         /* 11 */ FW_ERR_NOTUSEABLE,
00059         /* 12 */ FW_ERR_LOCKED,
00060         /* 13 */ FW_ERR_ACCESSDENIED,
00061         /* 14 */ FW_ERR_NOTEMPTY,
00062         /* 15 */ FW_ERR_INITFUNC,
00063         /* 16 */ FW_ERR_CARDREMOVED,
00064         /* 17 */ FW_ERR_ONDRIVE,
00065         /* 18 */ FW_ERR_INVALIDSECTOR,
00066         /* 19 */ FW_ERR_READ,
00067         /* 20 */ FW_ERR_WRITE,
00068         /* 21 */ FW_ERR_INVALIDMEDIA,
00069         /* 22 */ FW_ERR_BUSY,
00070         /* 23 */ FW_ERR_WRITEPROTECT,
00071         /* 24 */ FW_ERR_INVFATTYPE,
00072         /* 25 */ FW_ERR_MEDIATOOSMALL,
00073         /* 26 */ FW_ERR_MEDIATOOLARGE,
00074         /* 27 */ FW_ERR_NOTSUPPSECTORSIZE,
00075         /* 28 */ FW_ERR_UNKNOWN,
00076         /* 29 */ FW_ERR_DRVALREADYMNT,
00077         /* 30 */ FW_ERR_TOOLONGNAME,
00078         /* 31 */ FW_ERR_NOTFORREAD
00079     };
00080
00081     /*****************************************************************************
00082      *
00083      * FS FAT errorcodes
00084      *
00085      *****************************************************************************/
00086
00087 #define F_NO_ERROR FW_NO_ERROR
00088 #define F_ERR_INVALIDDRIVE FW_ERR_INVALIDDRIVE
00089 #define F_ERR_NOTFORMATTED FW_ERR_NOTFORMATTED
00090 #define F_ERR_INVALIDDIR FW_ERR_INVALIDDIR
00091 #define F_ERR_INVALIDNAME FW_ERR_INVALIDNAME
00092 #define F_ERR_NOTFOUND FW_ERR_NOTFOUND
00093 #define F_ERR_DUPLICATED FW_ERR_DUPLICATED
00094 #define F_ERR_NOMOREENTRY FW_ERR_NOMOREENTRY
00095 #define F_ERR_NOTOPEN FW_ERR_NOTOPEN
00096 #define F_ERR_EOF FW_ERR_EOF
00097 #define F_ERR_RESERVED FW_ERR_RESERVED
00098 #define F_ERR_NOTUSEABLE FW_ERR_NOTUSEABLE
00099 #define F_ERR_LOCKED FW_ERR_LOCKED
00100 #define F_ERR_ACCESSDENIED FW_ERR_ACCESSDENIED
00101 #define F_ERR_NOTEMPTY FW_ERR_NOTEMPTY
00102 #define F_ERR_INITFUNC FW_ERR_INITFUNC
00103 #define F_ERR_CARDREMOVED FW_ERR_CARDREMOVED
00104 #define F_ERR_ONDRIVE FW_ERR_ONDRIVE
00105 #define F_ERR_INVALIDSECTOR FW_ERR_INVALIDSECTOR
00106 #define F_ERR_READ FW_ERR_READ
00107 #define F_ERR_WRITE FW_ERR_WRITE
00108 #define F_ERR_INVALIDMEDIA FW_ERR_INVALIDMEDIA
00109 #define F_ERR_BUSY FW_ERR_BUSY
00110 #define F_ERR_WRITEPROTECT FW_ERR_WRITEPROTECT
00111 #define F_ERR_INVFATTYPE FW_ERR_INVFATTYPE
00112 #define F_ERR_MEDIATOOSMALL FW_ERR_MEDIATOOSMALL
00113 #define F_ERR_MEDIATOOLARGE FW_ERR_MEDIATOOLARGE
00114 #define F_ERR_NOTSUPPSECTORSIZE FW_ERR_NOTSUPPSECTORSIZE
00115
00116     /*****************************************************************************
00117      *
00118      * FS STD errorcodes
00119      *
00120      *****************************************************************************/
00121
00122 #define FS_NOERR FW_NO_ERROR
00123 #define FS_INVALIDDRIVE FW_ERR_INVALIDDRIVE
00124 #define FS_INVALIDDIR FW_ERR_INVALIDDIR
00125 #define FS_INVALIDNAME FW_ERR_INVALIDNAME
00126 #define FS_NOMOREENTRY FW_ERR_NOMOREENTRY
00127 #define FS_DRIVEERROR FW_ERR_ONDRIVE
00128 #define FS_DUPLICATED FW_ERR_DUPLICATED
00129 #define FS_NOTFOUND FW_ERR_NOTFOUND
00130 #define FS_NOTEMPTY FW_ERR_NOTEMPTY
00131 #define FS_NOTUSEABLE FW_ERR_NOTUSEABLE
00132 #define FS_NOTFORREAD FW_ERR_NOTFORREAD
00133 #define FS_NOTOPEN FW_ERR_NOTOPEN
00134 #define FS_BUSY FW_ERR_BUSY
00135 #define FS_NOTFORMATTED FW_ERR_NOTFORMATTED
00136 #define FS_NOPERMISSION FW_ERR_ACCESSDENIED
00137 #define FS_DRVALREADYMNT FW_ERR_DRVALREADYMNT
00138 #define FS_TOOLONGNAME FW_ERR_TOOLONGNAME
00139
00140 #ifdef __cplusplus
```

```
00141 }
00142 #endif
00143
00144 #endif /* _FWERR_H_ */
```

24.337 mmc_dsc.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _MMC_DSC_H_
00004 #define _MMC_DSC_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  *****************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039 #define _T_LOWVOLTAGE 0x80
00040 #define _T_MMC 0x01
00041 #define _T_SD 0x02
00042 #define _T_SDV2 0x04
00043 #define _T_SDHC 0x08
00044
00045     typedef struct
00046     {
00047         unsigned char initok; /* card initialized */
00048         unsigned char cardtype;
00049         unsigned char use_crc;
00050         unsigned long number_of_sectors;
00051         unsigned char bcs; /* block count supported 0-no 1-yes */
00052
00053         unsigned char CSD[16];
00054
00055         unsigned char TRANSPEED;
00056         unsigned char R_BL_LEN;
00057         unsigned short CSIZE;
00058         unsigned char CSIZE_M;
00059
00060         unsigned char TAAC;
00061         unsigned char NSAC;
00062         unsigned char R2W;
00063     } t_mmc_dsc;
00064
00065 #ifdef __cplusplus
00066 }
00067 #endif
00068
00069 /*****************************************************************************
00070  *
00071  * end of mmc.h
00072  *
00073  *****************************************************************************/
00074
00075 #endif /* _MMC_H_ */
```

24.338 mmc_mcf.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _MMC_H_
00004 #define _MMC_H_
00005
00006 /***************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ***************************************************************************/
00033
00034 #include <effs_fat/common.h>
00035 #include <effs_fat/fat.h>
00036 #include <effs_fat/mmc_dsc.h>
00037
00038 #ifdef __cplusplus
00039 extern "C"
00040 {
00041 #endif
00042 /*
00043    SD/MMC shares the (Q)SPI with WLAN (wodule) at default installation
00044       Uncommenting NB_ENABLE_USER_QSPI in predefs.h changes the driver
00045       to the user (Q)SPI driver defined in qspi.h and does not then support
00046       using the SD/MMC and wodule in the same system.
00047
00048       predef.h must precede this include file.
00049
00050 */
00051 #ifdef _PREDEF_H_
00052 #ifndef NB_ENABLE_USER_QSPI
00053 #define SD_SHARES_SPI (1)
00054 #endif /* #ifndef NB_ENABLE_USER_QSPI    */
00055 #else  /* #ifndef _PREDEF_H_ */
00056 #error predef.h must be included before mmc_mcf.h is included
00057 #endif /* #ifndef _PREDEF_H_ */
00058
00059     /* Uncomment SD_IRQ_SPI definition to enable interrupt driven SPI for the SD
00060        Card access.  This will have a small decrease in the read and write speeds but
00061        will signifigantly lighten the load of the CPU while reading and writing to the SD Card.
00062        Other tasks will function more smoothly durring file transfers with IRQs enabled here.
00063     */
00064     //#define SD_IRQ_SPI
00065
00066 #define SDHC_ENABLE 1   /* enable SDHC support */
00067 #define USE_CRC 1       /* use CRC for communication */
00068 #define CRC_ROM_TABLE 1 /* put CRC table in ROM */
00069     extern unsigned long MMC_CRC_Enable;
00070
00071     int spi_init(void);                   /* init SPI */
00072     void spi_set_baudrate(unsigned long); /* set baudrate */
00073     unsigned long spi_get_baudrate(void); /* get baudrate */
00074     void spi_tx1(unsigned char);          /* transmit 1 byte */
00075     void spi_tx2(unsigned short);         /* transmit 2 bytes */
00076     void spi_tx4(unsigned long);          /* transmit 4 bytes */
00077     void spi_tx512(unsigned char *);      /* transmit 512 bytes */
00078     unsigned char spi_rx1(void);          /* receive 1 byte */
00079     void spi_rx512(unsigned char *);      /* receive 512 bytes */
00080     void spi_cs_lo(void);                 /* CS low */
00081     void spi_cs_hi(void);                 /* CS high */
00082
00083     int get_cd(void); /* get Card Detect state */
00084     int get_wp(void); /* get Write Protect state */
00085
```

```
00086     extern F_DRIVER *mmc_initfunc(unsigned long driver_param);
00087
00088 #define MMC_ERR_NOTPLUGGED -1 /* for high level */
00089
00090 #define F_MMC_DRIVE0 0
00091 #define F_MMC_DRIVE1 1
00092 #define F_MMC_DRIVE2 2
00093 #define F_MMC_DRIVE3 3
00094
00095     enum
00096     {
00097         MMC_NO_ERROR,
00098         MMC_ERR_NOTINITIALIZED = 101,
00099         MMC_ERR_INIT,
00100         MMC_ERR_CMD,
00101         MMC_ERR_STARTBIT,
00102         MMC_ERR_BUSY,
00103         MMC_ERR_CRC,
00104         MMC_ERR_WRITE,
00105         MMC_ERR_WRITEPROTECT,
00106         MMC_ERR_NOTAVAILABLE
00107     };
00108
00109 #ifdef __cplusplus
00110 }
00111 #endif
00112
00113 /*****************************************************************************
00114  *
00115  * end of mmc.h
00116  *
00117  *****************************************************************************/
00118
00119 #endif /* _MMC_H_ */
```

24.339 multi_drive_mmc_mcf.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _MMC_H_
00004 #define _MMC_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  *****************************************************************************/
00033
00034 #include <effs_fat/common.h>
00035 #include <effs_fat/fat.h>
00036 #include <effs_fat/mmc_dsc.h>
00037
00038 #ifdef __cplusplus
00039 extern "C"
00040 {
00041 #endif
00042 /*
00043    SD/MMC shares the (Q)SPI with WLAN (wodule) at default installation
00044       Uncommenting NB_ENABLE_USER_QSPI in predefs.h changes the driver
00045       to the user (Q)SPI driver defined in qspi.h and does not then support
00046       using the SD/MMC and wodule in the same system.
00047
00048       predef.h must precede this include file.
```

```
00049
00050 */
00051 #ifdef _PREDEF_H_
00052 #ifndef NB_ENABLE_USER_QSPI
00053 #define SD_SHARES_SPI (1)
00054 #endif /* #ifndef NB_ENABLE_USER_QSPI    */
00055 #else  /* #ifndef _PREDEF_H_ */
00056 #error predef.h must be included before mmc_mcf.h is included
00057 #endif /* #ifndef _PREDEF_H_ */
00058
00059     /* Uncomment SD_IRQ_SPI definition to enable interrupt driven SPI for the SD
00060        Card access.  This will have a small decrease in the read and write speeds but
00061        will signifigantly lighten the load of the CPU while reading and writing to the SD Card.
00062        Other tasks will function more smoothly durring file transfers with IRQs enabled here.
00063     */
00064     //#define SD_IRQ_SPI
00065
00066 #define SDHC_ENABLE 1   /* enable SDHC support */
00067 #define USE_CRC 1       /* use CRC for communication */
00068 #define CRC_ROM_TABLE 1 /* put CRC table in ROM */
00069     extern unsigned long MMC_CRC_Enable;
00070
00071     int spi_init(int drv);                          /* init SPI */
00072     void spi_set_baudrate(int drv, unsigned long); /* set baudrate */
00073     unsigned long spi_get_baudrate(int drv);       /* get baudrate */
00074     void spi_tx1(int drv, unsigned char);          /* transmit 1 byte */
00075     void spi_tx2(int drv, unsigned short);         /* transmit 2 bytes */
00076     void spi_tx4(int drv, unsigned long);          /* transmit 4 bytes */
00077     void spi_tx512(int drv, unsigned char *);      /* transmit 512 bytes */
00078     unsigned char spi_rx1(int drv);                /* receive 1 byte */
00079     void spi_rx512(int drv, unsigned char *);      /* receive 512 bytes */
00080     void spi_cs_lo(int drv);                       /* CS low */
00081     void spi_cs_hi(int drv);                       /* CS high */
00082
00083     int get_cd(int drv); /* get Card Detect state */
00084     int get_wp(int drv); /* get Write Protect state */
00085
00086     extern F_DRIVER *mmc_initfunc(unsigned long driver_param);
00087
00088 #define MMC_ERR_NOTPLUGGED -1 /* for high level */
00089
00090 #define F_MMC_DRIVE0 0
00091 #define F_MMC_DRIVE1 1
00092 #define F_MMC_DRIVE2 2
00093 #define F_MMC_DRIVE3 3
00094
00095     enum
00096     {
00097         MMC_NO_ERROR,
00098         MMC_ERR_NOTINITIALIZED = 101,
00099         MMC_ERR_INIT,
00100         MMC_ERR_CMD,
00101         MMC_ERR_STARTBIT,
00102         MMC_ERR_BUSY,
00103         MMC_ERR_CRC,
00104         MMC_ERR_WRITE,
00105         MMC_ERR_WRITEPROTECT,
00106         MMC_ERR_NOTAVAILABLE
00107     };
00108
00109 #ifdef __cplusplus
00110 }
00111 #endif
00112
00113 /*****************************************************************************
00114  *
00115  * end of mmc.h
00116  *
00117  *****************************************************************************/
00118
00119 #endif /* _MMC_H_ */
```

24.340 port_f.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _PORT_F_H_
00004 #define _PORT_F_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003-2006 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
```

```
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ******************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039     extern unsigned long f_getrand(unsigned long rand);
00040     extern unsigned short f_getdate(void);
00041     extern unsigned short f_gettime(void);
00042
00043 #if (!FN_CAPI_USED)
00044     extern int f_mutex_get(FN_MUTEX_TYPE *);
00045     extern int f_mutex_put(FN_MUTEX_TYPE *);
00046     extern int f_mutex_create(FN_MUTEX_TYPE *);
00047     extern int f_mutex_delete(FN_MUTEX_TYPE *);
00048     extern long fn_gettaskID(void);
00049 #endif
00050
00051 #ifdef __cplusplus
00052 }
00053 #endif
00054
00055 /*****************************************************************************
00056  *
00057  * end of port_f.h
00058  *
00059  *****************************************************************************/
00060
00061 #endif /* _PORT_F_H_ */
```

24.341 ramdrv_f.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _RAMDRV_F_H_
00004 #define _RAMDRV_F_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ******************************************************************************/
```

```
00033
00034 #include <effs_fat/fat.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041     extern F_DRIVER *f_ramdrvinit(unsigned long driver_param);
00042
00043 #define F_RAM_DRIVE0 0
00044 #define F_RAM_DRIVE1 1
00045
00046 #define RAMDRV_CNT 2   // DO NOT CHANGE!
00047
00048     typedef struct
00049     {
00050         char *ramdrv;
00051         unsigned long maxsector;
00052         int use;
00053         F_DRIVER *driver;
00054     } t_RamDrv;
00055
00056     extern char ramdrv0[];
00057     extern char ramdrv1[];
00058     extern F_DRIVER t_drivers[];
00059     extern t_RamDrv RamDrv[];
00060
00061     enum
00062     {
00063         RAM_NO_ERROR,
00064         RAM_ERR_SECTOR = 101,
00065         RAM_ERR_NOTAVAILABLE
00066     };
00067
00068 #ifdef __cplusplus
00069 }
00070 #endif
00071
00072 /*****************************************************************************
00073  *
00074  *  End of ramdrv.c
00075  *
00076  *****************************************************************************/
00077
00078 #endif /* _RAMDRV_H_ */
```

24.342 sdhc_mcf.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SDHC_H_
00006 #define _SDHC_H_
00007
00008 #include "fat.h"
00009 //#include "common.h"
00010 #include "effs_fat/common.h"
00011
00012 #ifdef __cplusplus
00013 extern "C"
00014 {
00015 #endif
00016
00017 // Logging levels
00018 #define SDHC_LOG_LEVEL_OFF (0)
00019 #define SDHC_LOG_LEVEL_INFO (1)
00020 #define SDHC_LOG_LEVEL_WARNING (2)
00021 #define SDHC_LOG_LEVEL_ERROR (3)
00022 #define SDHC_LOG_LEVEL_DEBUG (4)
00023
00024 #ifdef _DEBUG
00025 //#undef SDHC_LOG_LEVEL
00026 #define SDHC_LOG_LEVEL SDHC_LOG_LEVEL_DEBUG   // Secify appropriate logging level here if required
00027 #else
00028 //#define SDHC_LOG_LEVEL SDHC_LOG_LEVEL_DEBUG   // Secify appropriate logging level here if required
00029 #define SDHC_LOG_LEVEL SDHC_LOG_LEVEL_OFF   // Secify appropriate logging level here if required
00030 #endif
00031
00032 #define IsLogEnabled(x) (SDHC_LOG_LEVEL >= x)
00033
00034 #if defined(SDHC_LOG_LEVEL) && (SDHC_LOG_LEVEL != SDHC_LOG_LEVEL_OFF)
00035
00036 #include <stdio.h>
```

```
00037
00038 #if (SDHC_LOG_LEVEL >= SDHC_LOG_LEVEL_INFO)
00039 #define SDHC_LOG_INFO(args...) iprintf(args)
00040 #else
00041 #define SDHC_LOG_INFO(args...)
00042 #endif
00043
00044 #if (SDHC_LOG_LEVEL >= SDHC_LOG_LEVEL_WARNING)
00045 #define SDHC_LOG_WARNING(args...) iprintf(args)
00046 #else
00047 #define SDHC_LOG_WARNING(...)
00048 #endif
00049
00050 #if (SDHC_LOG_LEVEL >= SDHC_LOG_LEVEL_ERROR)
00051 #define SDHC_LOG_ERROR(args...) iprintf(args)
00052 #else
00053 #define SDHC_LOG_ERROR(...)
00054 #endif
00055
00056 #if (SDHC_LOG_LEVEL >= SDHC_LOG_LEVEL_DEBUG)
00057 #define SDHC_LOG_DEBUG(args...) iprintf(args)
00058 #else
00059 #define SDHC_LOG_DEBUG(...)
00060 #endif
00061
00062 #else
00063 #define SDHC_LOG_INFO(...)
00064 #define SDHC_LOG_WARNING(...)
00065 #define SDHC_LOG_ERROR(...)
00066 #define SDHC_LOG_DEBUG(...)
00067 #endif
00068
00069 #define SDHC_LOG_LINE SDHC_LOG_DEBUG("Got to line %d in file %s\r\n", __LINE__, __FILE__);
00070 #define SDHC_LOG_FUNC(args...)                \
00071     {                                         \
00072         SDHC_LOG_DEBUG("%s: ", __FUNCTION__); \
00073         SDHC_LOG_DEBUG(args);                 \
00074     }
00075 #define SDHC_LOG_TICK(args...)   // SDHC_LOG_INFO("%s: %lu, %s\n\r", __FUNCTION__, TimeTick, args);
00076
00077 #define MAX_SDHC_DRIVES (1)
00078 #define F_SDHC_DRIVE0 0
00079 //#define F_SDHC_AUTOASSIGN     -1
00080
00081 //#define   ENABLE_HIGH_SPEED_MODE (1)  // Uncomment to enable High Speed Mode if supported
00082 //#define   ENABLE_PREERASE (1)     // Enables execution of pre-erase command (can increase write
      speed on some cards)
00083 //#define ENABLE_BLOCS_COUNT_CMD    (1)
00084 #define ESDHC_ENDIAN_CONVERSION (1)   // Must be define to make card format compatible with PC
00085
00086 #define CMD_RETRIES_COUNT 1
00087 #define DAT_RETRIES_COUNT 3
00088 #define SECTOR_SIZE ((uint32_t)F_SECTOR_SIZE)
00089 #define RW_BUFF_SECTORS (128UL)
00090
00091 #define INIT_BAUDRATE (100000UL)   // Hz
00092 #ifdef __MIMXRT10xx__
00093 #define WORK_BAUDRATE (50000000UL)   // Hz
00094 #else
00095 #define WORK_BAUDRATE (25000000UL)   // Hz
00096 #endif
00097 #define HIGHSPED_BAUDRATE (31250000)   //(50000000UL)
00098
00099 #define ESDHC_IRQ_LEVEL (5U)   // Defines the IRQ level of ESDHC interrupt routine 0-6, 0 - auto
00100
00101     int sdhc_init(int drv);  /* Init SDHC drive */
00102     int sdhc_close(int drv); /* Init SDHC drive */
00103
00104     int sdhc_read(int drv, void *buff, unsigned long sector, unsigned int count);  /* Read data from
      SDHC device */
00105     int sdhc_write(int drv, void *buff, unsigned long sector, unsigned int count); /* Write data to
      SDHC device*/
00106     int sdhc_get_cd(int drv);                                                      /* Get Card Detect
      state */
00107     int sdhc_get_wp(int drv);                                                      /* Get Write
      Protect state */
00108     int sdhc_check_card_status(int drv);
00109     void sdhc_init_card(int drv);
00110     unsigned long sdhc_get_card_size(int drv);
00111
00112     extern F_DRIVER *sdhc_initfunc(unsigned long driver_param);
00113
00114     enum
00115     {
00116         SDHC_NO_ERROR,
00117         SDHC_ERR_NOTINITIALIZED = 101,
00118         SDHC_ERR_INIT,
```

```
00119         SDHC_ERROR,   // A general or an unknown error occurred during the operation
00120         SDHC_ERR_CMD,
00121         SDHC_ERR_TIMEOUT,
00122         SDHC_ERR_BUSY,
00123         SDHC_ERR_CRC,
00124         SDHC_ERR_READ,
00125         SDHC_ERR_WRITE,
00126         SDHC_ERR_WRITEPROTECT,
00127         SDHC_ERR_LOCKED,
00128         SDHC_ERR_NOTAVAILABLE
00129     };
00130
00131 #ifdef __cplusplus
00132 }
00133 #endif
00134
00135 /****************************************************************************
00136  *
00137  * end of sdhc_mcf.h
00138  *
00139  ****************************************************************************/
00140
00141 #endif /* _SDHC_H_ */
```

24.343 udefs_f.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _UDEFS_F_H_
00004 #define _UDEFS_F_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039 /****************************************************************************
00040  *
00041  * enable this if CAPI (Common API) is used
00042  *
00043  ****************************************************************************/
00044 #define FN_CAPI_USED 0
00045
00046 /****************************************************************************
00047  *
00048  * OEM name
00049  *
00050  ****************************************************************************/
00051 #define OEM_NAME "MSDOS5.0"
00052     /*#define OEM_NAME "EFFSFAT"*/
00053
00054     /****************************************************************************
00055      *
00056      * CAPI selected includes
00057      *
00058      ****************************************************************************/
00059
```

```
00060 #if FN_CAPI_USED
00061 #include "../../fw_port.h"
00062 #else
00063
00064 /****************************************************************************
00065  *
00066  *  if Unicode is used then comment in HCC_UNICODE define
00067  *
00068  ****************************************************************************/
00069 /* #define HCC_UNICODE */
00070
00071 #ifndef HCC_UNICODE
00072 #define F_LONGFILENAME 1 /*  0 - 8+3 names   1 - long file names   */
00073 #define W_CHAR char
00074 #else
00075 #define F_LONGFILENAME 1 /* don't change it, because unicode version always uses long file name */
00076 #define W_CHAR wchar
00077 #endif
00078
00079 #ifdef HCC_UNICODE
00080 typedef unsigned short wchar;
00081 #endif
00082
00083 /****************************************************************************
00084  *
00085  * volumes definitions
00086  *
00087  ****************************************************************************/
00088
00089 #define FN_MAXVOLUME 5 /* maximum number of volumes */
00090 #define FN_MAXTASK 10  /* maximum number of task */
00091
00092 #define FN_MAXPATH 256 /* maximum allowed filename or pathname */
00093
00094 #define FN_CURRDRIVE 0 /* setting the current drive at startup (-1 means no default current drive)*/
00095
00096 #define FN_MUTEX_TYPE OS_CRIT
00097
00098 /* select path separator */
00099 #if 1
00100 #define F_SEPARATORCHAR '/'
00101 #else
00102 #define F_SEPARATORCHAR '\\'
00103 #endif
00104
00105 /****************************************************************************
00106  *
00107  * Last error usage
00108  *
00109  ****************************************************************************/
00110
00111 #if 0
00112 /* simple asignment */
00113 #define F_SETLASTERROR(ec) (fm->lasterror = (ec))
00114 #define F_SETLASTERROR_NORET(ec) (fm->lasterror = (ec))
00115 #elif 1
00116 /* function calls used for it */
00117 #define F_SETLASTERROR(ec) fn_setlasterror(fm, ec)
00118 #define F_SETLASTERROR_NORET(ec) fn_setlasterror_noret(fm, ec)
00119 #elif 0
00120 /* no last error is used (save code space) */
00121 #define F_SETLASTERROR(ec) (ec)
00122 #define F_SETLASTERROR_NORET(ec)
00123 #endif
00124
00125 /****************************************************************************
00126  *
00127  * Close bracket for non CAPI
00128  *
00129  ****************************************************************************/
00130
00131 #endif /* FN_CAPI_USED */
00132
00133     /****************************************************************************
00134      *
00135      * Common defines (for non CAPI and CAPI
00136      *
00137      ****************************************************************************/
00138
00139 #define F_MAXFILES 10 /* maximum number of files */
00140
00141 #define F_MAXSEEKPOS 8 /* number of division of fast seeking */
00142
00143     /****************************************************************************
00144      *
00145      * functions definitions
00146      *
```

```
00147      *****************************************************************************/
00148
00149     /* Use internal mem functions (memcpy,memset) or switch to library functions */
00150     //#define INTERNAL_MEMFN
00151
00152     /* Use malloc for cache items */
00153     /* #define USE_MALLOC */
00154
00155 #ifdef USE_MALLOC
00156 #define _malloc(x) malloc(x) /* normally use malloc from library */
00157 #define _free(x) free(x)     /* normally use free from library */
00158 #endif
00159
00160 /* Enable FAT caching */
00161 #define FATCACHE_ENABLE
00162 #if F_LONGFILENAME
00163 #define DIRCACHE_ENABLE
00164 #endif
00165
00166 /* define of allocation of faster searching mechanism */
00167 #ifdef USE_MALLOC
00168 #define FATBITFIELD_ENABLE
00169 #endif
00170
00171 #ifdef FATCACHE_ENABLE
00172 #define FATCACHE_BLOCKS 4
00173 #define FATCACHE_READAHEAD 8 /* max. 256 */
00174 #endif
00175
00176 #if F_LONGFILENAME
00177 #ifdef DIRCACHE_ENABLE
00178 #define DIRCACHE_SIZE 32 /* max. 32 (<=max. cluster size) */
00179 #endif
00180 #endif
00181
00182 #define WR_DATACACHE_SIZE 32 /* min. 1 !!!! */
00183
00184 #ifdef FATCACHE_ENABLE
00185 #define FATCACHE_SIZE (FATCACHE_BLOCKS * FATCACHE_READAHEAD)
00186 #endif
00187
00188 #ifdef INTERNAL_MEMFN
00189 #define _memcpy(d, s, l) _f_memcpy(d, s, l)
00190 #define _memset(d, c, l) _f_memset(d, c, l)
00191 #else
00192 #include <string.h>
00193 #define _memcpy(d, s, l) memcpy(d, s, l)
00194 #define _memset(d, c, l) memset(d, c, l)
00195 #endif
00196
00197 #ifdef USE_MALLOC
00198 #include <stdlib.h>
00199 #endif
00200
00201     /***************************************************************************
00202      *
00203      * Last access date
00204      *
00205      ***************************************************************************/
00206
00207 #define F_UPDATELASTACCESSDATE 0
00208     /* it defines if a file is opened for read to update lastaccess time */
00209
00210     /***************************************************************************
00211      *
00212      * Opened file size
00213      *
00214      ***************************************************************************/
00215
00216 #define F_FINDOPENFILESIZE 1
00217     /* set F_FINDOPENFILESIZE to 0 if filelength needs to return with 0 for an opened file  */
00218     /* other case filelength functions can return with opened file length also */
00219
00220     /***************************************************************************
00221      *
00222      * closing bracket for C++
00223      *
00224      ***************************************************************************/
00225
00226 #ifdef __cplusplus
00227 }
00228 #endif
00229
00230 /***************************************************************************
00231  *
00232  * end of udefs_f.h
00233  *
```

```
00234  ******************************************************************************/
00235
00236 #endif /* _UDEFS_F_H_ */
```

24.344 endian.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __NB_ENDIAN_H
00006 #define __NB_ENDIAN_H
00007
00008 #include <cpu.h>
00009
00010 #ifdef NB_BIG_ENDIAN
00011 #ifdef NB_LITTLE_ENDIAN
00012 #error Both NB_BIG_ENDIAN and NB_LITTLE_ENDIAN defined.
00013 #endif
00014 #endif
00015
00016 #ifdef NB_BIG_ENDIAN
00017 #define HTOBES(x) (x)
00018 #define HTOBEL(x) (x)
00019
00020 #define HTOLES(x) ((((x)&0xff00) » 8) | (((x)&0xff) « 8))
00021
00022 #define HTOLEL(x) ((((x) » 24) & 0xff) | (((x)&0xff00) « 8) | (((x) » 8) & 0xff00) | (((x)&0xff) «
      24))
00023 #endif /* -----  #ifdef NB_BIG_ENDIAN  ----- */
00024
00025 #ifdef NB_LITTLE_ENDIAN
00026 #define HTOBES(x) ((((x)&0xff00) » 8) | (((x)&0xff) « 8))
00027
00028 #define HTOBEL(x) ((((x) » 24) & 0xff) | (((x)&0xff00) « 8) | (((x) » 8) & 0xff00) | (((x)&0xff) «
      24))
00029
00030 #define HTOLES(x) (x)
00031 #define HTOLEL(x) (x)
00032 #endif /* -----  #ifdef NB_LITTLE_ENDIAN  ----- */
00033
00034 #endif /* ----- #ifndef __NB_ENDIAN_H  ----- */
```

24.345 ethernet.h File Reference

NetBurner Ethernet API.

```
#include <basictypes.h>
#include <buffers.h>
#include <nettypes.h>
```

Macros

- #define **ETHERNET_ETHERTYPE_IPv4** (uint16_t)(0x0800)

  Internet Protocol, Version 4 (IPv4)
- #define **ETHERNET_ETHERTYPE_ARP** (uint16_t)(0x0806)

  Address Resolution Protocol (ARP)
- #define **ETHERNET_ETHERTYPE_IPv6** (uint16_t)(0x86DD)

  Internet Protocol, Version 6 (IPv6)
- #define **ETHERNET_ETHERTYPE_AARP** (uint16_t)(0x80F3)

  AppleTalk Address Resolution Prot. (AARP)
- #define **ETHERNET_ETHERTYPE_IPX** (uint16_t)(0x8137)

  Novell Internet Packet Exchange (IPX) (alt.)
- #define **ETHERNET_ETHERTYPE_EAPOL** (uint16_t)(0x888E)

  Extensible Authorization Protocol (EAP) over LAN.
- #define **ETHERNET_ETHERTYPE_VLAN** (uint16_t)(0x8100)

  Virtual Private Network (VLAN)

- #define NO_AUTOMATIC_2ND_ETHERNET extern const bool bAutomatic2ndEther = false;

  Disable automatic initialization of second Ethernet interface.

Functions

- void AddEthernetInterfaces ()

  Add an Ethernet interface.
- void ManualEthernetConfig (int interface, BOOL speed100Mbit, BOOL fullDuplex, BOOL autoNegotiate)

  Manually configure Ethernet speed and duplex settings.
- void DisablePHY (int ifn)

  Disable the specified Ethernet PHY.
- void EnablePHY (int ifn)

  Disable the specified Ethernet PHY.

24.345.1 Detailed Description

NetBurner Ethernet API.

24.346 ethernet.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef _NB_ETHERNET_H
00016 #define _NB_ETHERNET_H
00017
00018 #include <basictypes.h>
00019 #include <buffers.h>
00020 #include <nettypes.h>
00021
00022 // Definitions
00023
00024 /*
00025    EtherType Field ( Ethernet Version II )
00026       ETHERNET_ETHERTYPE_IPv4       - Internet Protocol, Version 4 (IPv4)
00027       ETHERNET_ETHERTYPE_ARP        - Address Resolution Protocol (ARP)
00028       ETHERNET_ETHERTYPE_IPv6       - Internet Protocol, Version 6 (IPv6)
00029       ETHERNET_ETHERTYPE_AARP       - AppleTalk Address Resolution Prot. (AARP)
00030       ETHERNET_ETHERTYPE_IPX        - Novell Internet Packet Exchange (IPX) (alt.)
00031       ETHERNET_ETHERTYPE_EAPOL      - Extensible Authorization Protocol (EAP) over LAN
00032
00033 */
00034
00039 #define ETHERNET_ETHERTYPE_IPv4 (uint16_t)(0x0800)
00040 #define ETHERNET_ETHERTYPE_ARP (uint16_t)(0x0806)
00041 #define ETHERNET_ETHERTYPE_IPv6 (uint16_t)(0x86DD)
00042 #define ETHERNET_ETHERTYPE_AARP (uint16_t)(0x80F3)
00043 #define ETHERNET_ETHERTYPE_IPX (uint16_t)(0x8137)
00044 #define ETHERNET_ETHERTYPE_EAPOL (uint16_t)(0x888E)
00045 #define ETHERNET_ETHERTYPE_VLAN (uint16_t)(0x8100)
00048 #define IP_20BYTE_ID (0x4500)
00049
00050 // Data Structures
00051 /*
00052    Ethernet Type II Frame Header
00053
00054       macAddrDst  - Destination MAC address
00055       macAddrSrc  - Source MAC address
00056       etherType   - Protocol
00057 */
00058 typedef struct _EthernetFrameHeader
00059 {
00060     MACADDRESS_48 destinationMacAddress;
00061     MACADDRESS_48 sourceMacAddress;
00062     beuint16_t etherType;
00063
00064 } __attribute__((packed)) EthernetFrameHeader;
00065
00072 void AddEthernetInterfaces();
00073
00091 void ManualEthernetConfig(int interface, BOOL speed100Mbit, BOOL fullDuplex, BOOL autoNegotiate);
00092
```

```
00102 void DisablePHY(int ifn);
00103
00114 void EnablePHY(int ifn);
00115
00123 #define NO_AUTOMATIC_2ND_ETHERNET extern const bool bAutomatic2ndEther = false;
00124
00125 #endif
00126
```

24.347 fastlog.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <predef.h>
00006 #include <stdint.h>
00007 #include <stdarg.h>
00008 #include <nbrtos.h>
00009 #include <string.h>
00010
00011 #ifndef FASTLOG_BUFSIZ
00012 #define FASTLOG_BUFSIZ (0x20000)
00013 #endif
00014 //#define ENABLE_FASTLOG
00015
00016 int loggerputchars(void *data, const char *c, int n);
00017 namespace fastlog {
00018
00019 typedef enum {
00020     eEntryType_Empty,
00021     eEntryType_ULong,
00022     eEntryType_String,
00023     eEntryType_BinPtr,
00024 } eEntryType_t;
00025
00026 class logger {
00027 private:
00028     class CritObj {
00029         logger *pLogger;
00030     public:
00031         CritObj(logger *l)
00032             : pLogger(l)
00033         {
00034 //             USER_ENTER_CRITICAL()
00035 //                 return;
00036             if (pLogger->useFromIsr)
00037             {
00038                 USER_ENTER_CRITICAL()
00039             }
00040             else
00041             {
00042                 pLogger->crit.Enter();
00043             }
00044         }
00045         ~CritObj()
00046         {
00047 //             USER_EXIT_CRITICAL()
00048 //                 return;
00049             if (pLogger->useFromIsr)
00050             {
00051                 USER_EXIT_CRITICAL()
00052             }
00053             else
00054             {
00055                 pLogger->crit.Leave();
00056             }
00057         }
00058     };
00059     struct entry_t {
00060         uint16_t tickOffset;
00061         uint16_t line;
00062         uint16_t fileId;
00063         uint8_t  tcb_idx;
00064         eEntryType_t type;
00065         union {
00066             uint32_t data_ulong;
00067             struct {
00068                 uint16_t dataLen;
00069                 uint8_t data_bin[];
00070             }__attribute__((packed));
00071             struct {
00072                 uint16_t strLen;
00073                 char data_str[];
00074             }__attribute__((packed));
```

```
00075         };
00076         uint32_t getSize() const;
00077     } __attribute__((packed));
00078 public:
00079     class fileEntry {
00080         fileEntry *pNext;
00081         const char *fileName;
00082         uint16_t id;
00083     public:
00084         fileEntry(const char *fileName);
00085         inline uint16_t getId() const { return id; }
00086         inline const char *getFilename() const { return fileName; }
00087         inline const fileEntry *getNext() const { return pNext; }
00088         inline static const fileEntry *getFirst()
00089         { return fileIds; }
00090
00091     private:
00092         static fileEntry * fileIds;
00093         static uint16_t count;
00094     };
00095
00096     enum eFullAction_t {
00097         eFull_Roll,
00098         eFull_Drop,
00099         eFull_DumpAndClear
00100     };
00101 private:
00102     int      logFd;
00103     uint32_t start;
00104     uint32_t end;
00105     uint32_t rollovers;
00106     tick_t   startTick;
00107     tick_t   endTick;
00108     OS_CRIT  crit;
00109     bool     useFromIsr;
00110     bool     empty;
00111     eFullAction_t fullAction;
00112     uint32_t  sentinel;
00113     uint8_t logBuf[FASTLOG_BUFSIZ];
00114
00115     bool getNextByte(uint32_t *idx, uint8_t *b);
00116     bool DoFull(uint32_t entrySize);
00117     bool allocateEntry(uint32_t newEnd, uint32_t entrySize);
00118     void createEntry(entry_t &e);
00119     const char *getFileName(uint16_t id);
00120
00121     void dumpBin(uint32_t idx, uint16_t len);
00122     void dumpstr(uint32_t idx, uint16_t len);
00123     bool getEntry(uint32_t *idx, entry_t *e);
00124     void dumpEntry(tick_t prevTick, entry_t &e, uint32_t *dataStart);
00125
00126     int log_vsprintf(uint16_t fileId, uint16_t lineNum, const char *format, va_list &vl);
00127 public:
00128     logger(int logFd = 1, eFullAction_t fullAction = eFull_Roll, bool useFromIsr = false);
00129
00130     void log(uint16_t fileId, uint16_t lineNum, uint16_t len, uint8_t *pdata, eEntryType_t t);
00131
00132     inline void log_line(uint16_t fileId, uint16_t lineNum)
00133     { log(fileId, lineNum, 0, NULL, eEntryType_Empty); }
00134     inline void log_ulong(uint16_t fileId, uint16_t lineNum, uint32_t val)
00135     { log(fileId, lineNum, sizeof(val), (uint8_t*)&(val), eEntryType_ULong); }
00136
00137     inline void log_bin(uint16_t fileId, uint16_t lineNum, uint16_t len, uint8_t *pdata)
00138     { log(fileId, lineNum, len, pdata, eEntryType_BinPtr); }
00139
00140     inline void log_str(uint16_t fileId, uint16_t lineNum, const char *s)
00141     { log(fileId, lineNum, strlen(s), (uint8_t*)s, eEntryType_String); }
00142
00143     int log_sprintf(uint16_t fileId, uint16_t lineNum, const char *format, ...);
00144
00145     void dumpLog(bool clearAfter = true, int type = -1);
00146     void clear();
00147     void setLogFd(int fd) { logFd = fd; }
00148
00149     friend int ::loggerputchars(void *data, const char *c, int n);
00150 };
00151
00152 }
00153
00154 extern fastlog::logger FastStdLogger;
00155
00156 #ifdef ENABLE_FASTLOG
00157 #define FASTLOG_INIT_FILE() \
00158 static fastlog::logger::fileEntry _fastlog_ThisFile(__FILE__)
00159
00160 #define FASTLOG_LINE(_logger) (_logger).log_line(_fastlog_ThisFile.getId(), __LINE__)
00161 #define FASTLOG_ULONG(_logger, val) (_logger).log_ulong(_fastlog_ThisFile.getId(), __LINE__, (val))
```

```
00162 #define FASTLOG_BINARY(_logger, len, data) (_logger).log_bin(_fastlog_ThisFile.getId(), __LINE__,
      (len), (data))
00163 #define FASTLOG_STRING(_logger, s) (_logger).log_str(_fastlog_ThisFile.getId(), __LINE__, (s))
00164 #define FASTLOG_SPRINTF(_logger, ...) (_logger).log_sprintf(_fastlog_ThisFile.getId(), __LINE__,
      __VA_ARGS__)
00165
00166 #define FASTLOG_SHOWLOG(_logger) (_logger).dumpLog()
00167 #else
00168 #define FASTLOG_INIT_FILE()
00169
00170 #define FASTLOG_LINE(_logger)
00171 #define FASTLOG_ULONG(_logger, val)
00172 #define FASTLOG_BINARY(_logger, len, data)
00173 #define FASTLOG_STRING(_logger, s)
00174 #define FASTLOG_SPRINTF(_logger, ...)
00175
00176 #define FASTLOG_SHOWLOG(_logger)
00177 #endif
```

24.348 fd_adapter.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __FD_ADAPTER_H
00006 #define __FD_ADAPTER_H
00007
00008 #include <basictypes.h>
00009 #include <iosys.h>
00010 #include <buffers.h>
00011
00012 #define MAX_FDBUFFER_FIFO_BUFFERS   (20)
00013
00014 class fd_adapter
00015 {
00016    protected:
00017     int my_fd;
00018     bool inDtor;
00019
00020     virtual int read(char *buf, int nbytes) = 0;
00021     virtual int write(const char *buf, int nbytes) = 0;
00022     virtual int close() = 0;
00023     static int sread(int fd, char *buf, int nbytes);
00024     static int swrite(int fd, const char *buf, int nbytes);
00025     static int sclose(int fd);
00026     static fd_adapter *GetFromFD(int fd);
00027
00028    public:
00029     fd_adapter() {my_fd=0; inDtor = false;};
00030     // You _cannot_ call ::close inside the pure virtual base class.
00031     //   The reason is that by the time we get here, the derived class is destroyed,
00032     //   which means that the vtable we hit is the base class, which in turn
00033     //   has a pure virtual method for close. When the naked scope close method
00034     //   is called, this will hit the fd_adapter sclose which
00035     ~fd_adapter() { inDtor = true; if (my_fd) ::close(my_fd); };
00036     operator int() { if(my_fd==0) GetActiveFD(); return my_fd;};
00037     int GetActiveFD();
00038
00039 };
00040
00041 class FDCounter : public fd_adapter
00042 {
00043     int nwr;
00044
00045     virtual int read(char *buf, int nbytes);
00046     virtual int write(const char *buf, int nbytes);
00047     virtual int close();
00048
00049    public:
00050     FDCounter() { nwr = 0; };
00051     ~FDCounter() { if (my_fd) ::close(my_fd); }
00052
00053     int SpaceUsed() { return nwr; };
00054 };
00055
00056 class FDFlash : public fd_adapter
00057 {
00058     uint32_t m_len;
00059     uint32_t m_cs;
00060     uint8_t buffer[64];
00061     uint8_t *pd;
00062     uint32_t blen;
00063
00064     virtual int read(char *buf, int nbytes);
```

```
00065     virtual int write(const char *buf, int nbytes);
00066     virtual int close();
00067
00068     void Flush();
00069
00070    public:
00071     FDFlash(uint8_t *pw)
00072     {
00073         pd = pw;
00074         m_len = 0;
00075         m_cs = 0;
00076         blen = 0;
00077         buffer[0] = '\0';
00078     };
00079
00080     uint32_t cs() { return m_cs; };
00081
00082     uint32_t len() { return m_len; };
00083 };
00084
00085 class FDBuffer : public fd_adapter
00086 {
00087     fifo_buffer_storage bs{MAX_FDBUFFER_FIFO_BUFFERS, 0};
00088     int nwr;
00089
00090     int read(char *buf, int nbytes) override;
00091     int write(const char *buf, int nbytes) override;
00092     int close() override;
00093
00094    public:
00095     void Reset()
00096     {
00097         bs.Reset(MAX_FDBUFFER_FIFO_BUFFERS);
00098         nwr = 0;
00099     };
00100
00101     bool StreamTo(int fd);   // Returns true when done
00102
00103     FDBuffer() { nwr = 0; };
00104
00105     int SpaceUsed() { return nwr; };
00106 };
00107
00108 class FDcBuf : public fd_adapter
00109 {
00110     OS_CRIT crit;
00111     uint8_t *cbuf;
00112     int rdIdx;
00113     int wrIdx;
00114     int bufLen;
00115     int blockWaitSkip;
00116     bool empty;
00117
00118     int read(char *buf, int nbytes) override;
00119     int write(const char *buf, int nbytes) override;
00120     int close() override;
00121
00122    public:
00123     inline void Reset()
00124     {
00125         rdIdx = 0;
00126         wrIdx = 0;
00127         empty = true;
00128     }
00129
00130     FDcBuf(uint8_t *buf, int bufLen, int blockWaitSkip = 1)
00131         : cbuf(buf), rdIdx(0), wrIdx(0), bufLen(bufLen),
00132             blockWaitSkip(blockWaitSkip), empty(true)
00133     { }
00134     ~FDcBuf()
00135     {
00136         if (my_fd)
00137             ::close(my_fd);
00138
00139     }
00140
00141     inline int SpaceUsed() { return wrIdx - rdIdx + ((rdIdx > wrIdx)*bufLen); };
00142 };
00143 #endif /* ----- #ifndef __FD_ADAPTER_H  ----- */
```

24.349 fd_drivers.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
```

```
00004
00005 #ifndef _FD_DRIVERS_H_INC
00006 #define _FD_DRIVERS_H_INC
00007
00008 #define DATA_F_FILEP 1
00009 #define DATA_FS_FILEP 2
00010 #define DATA_CHAR_BUFFER 3
00011
00012 #endif /* ----- #ifndef _FD_DRIVERS_H_INC  ----- */
```

24.350 fdiprintf.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef FD_IPRINTF
00006 #define FD_IPRINTF
00007
00008 #include <stdarg.h>
00009
00010 int fdiprintf(int fc, const char *format, ...);
00011 int vfdiprintf(int fc, const char *format, va_list arg);
00012
00013 #endif
```

24.351 fdprintf.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef FD_PRINTF
00006 #define FD_PRINTF
00007
00008 #include <stdarg.h>
00009
00010 int fdprintf(int fc, const char *format, ...);
00011 int fdiprintf(int fc, const char *format, ...);
00012 int vfdprintf(int fc, const char *format, va_list arg);
00013 int vfdiprintf(int fc, const char *format, va_list arg);
00014
00015 #endif
```

24.352 effsstd.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*
00006  *-------------------------------------------------------------------
00007  * Embedded Flash File System for on-chip flash memory (EFFS-STD)
00008  * configuration file for common parameters.
00009  * This file is part of an example that allocates flash space
00010  * to the file system, and the rest to the application.
00011  *
00012  * Note:
00013  *    COMPCODEFLAGS contain starting and ending addresses
00014  *    of application. To add file system you must modify the ending
00015  *    address to provide for the flash used. Not to do so will result
00016  *    in unpredictable results.
00017  *
00018  * See"
00019  *    "NetBurner Embedded Flash File System, Hardware and Software
00020  *    Guide" for detailed information.
00021  *
00022  *-------------------------------------------------------------------
00023  */
00024
00025 #ifndef _EFFSSTD_H_
00026 #define _EFFSSTD_H_
00027
00028 /*********************************************************************
00029  * Definitions
00030  *********************************************************************/
00031 /* On-chip Flash NOR */
00032 //#define USE_NOR
00033
```

```
00034 /* Drive numbers */
00035 #define NOR_DRV_NUM 0
00036 #define STDRAM_DRV_NUM 1
00037 #define MMC_DRV_NUM 2
00038 #define CFC_DRV_NUM 3
00039 #define HDD_DRV_NUM 3
00040 #define FATRAM_DRV_NUM 4
00041 #define S19_DRV_NUM 5
00042
00043 #define FS_NO_ERROR FS_NOERR
00044
00045 #endif /* #ifndef _EFFSSTD_H_ */
```

24.353 flashdrv.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FLASHDRV_H_
00004 #define _FLASHDRV_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  *****************************************************************************/
00033
00034 #include <file/fsf.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041     extern int fs_mount_flashdrive(FS_VOLUMEDESC *vd, FS_PHYGETID phyfunc);
00042     extern long fs_getmem_flashdrive(FS_PHYGETID phyfunc);
00043
00044 #ifdef __cplusplus
00045 }
00046 #endif
00047
00048 /*****************************************************************************
00049  *
00050  * end of flashdrv.h
00051  *
00052  *****************************************************************************/
00053
00054 #endif /* _FLASHDRV_H_ */
```

24.354 fsf.h File Reference

Embedded Flash File System, EFFS-STD.

```
#include <file/fsm.h>
#include <file/fwerr.h>
#include <file/port_s.h>
#include <file/udefs.h>
#include "fsmf.h"
```

Macros

- #define **FS_SEEK_SET** 0

  Beginning of file.
- #define **FS_SEEK_CUR** 1

  Current position of the file pointer.
- #define **FS_SEEK_END** 2

  End of file.
- #define fs_getfreespace(drivenum, space) fsm_getfreespace(drivenum, space)

  Provides information about the drive space usage.
- #define fs_mkdir(dirname) fsm_mkdir(dirname)

  Makes a new directory.
- #define fs_chdir(dirname) fsm_chdir(dirname)

  Change the directory.
- #define fs_rmdir(dirname) fsm_rmdir(dirname)

  Removes a directory.
- #define fs_delete(filename) fsm_delete(filename)

  Deletes a file.
- #define fs_findfirst(filename, find) fsm_findfirst(filename, find)

  Find the first file or subdirectory in a specified directory.
- #define fs_findnext(find) fsm_findnext(find)

  Finds the next file or subdirectory in a specified directory after a previous call to `fs_findfirst()` or `fs_findnext()`.
- #define fs_open(filename, mode) fsm_open(filename, mode)

  Opens a file in the file system.
- #define fs_close(filehandle) fsm_close(filehandle)

  Closes an opened file.
- #define fs_write(buf, size, size_st, filehandle) fsm_write(buf, size, size_st, filehandle)

  Write data to the file at the current position.
- #define fs_read(buf, size, size_st, filehandle) fsm_read(buf, size, size_st, filehandle)

  Read data from the current position in a file.
- #define fs_seek(filehandle, offset, whence) fsm_seek(filehandle, offset, whence)

  Move the stream position of an open file.
- #define fs_eof(filehandle) fsm_eof(filehandle)

  Check whether the current position in the open target file is the end of the file.
- #define fs_rewind(filehandle) fsm_rewind(filehandle)

  Sets the file position in the open target file to the start of the file.
- #define fs_settimedate(filename, ctime, cdate) fsm_settimedate(filename, ctime, cdate)

  Set the time and date of a file or directory.
- #define fs_gettimedate(filename, pctime, pcdate) fsm_gettimedate(filename, pctime, pcdate)

  Get the time and date of a file or directory.

24.354.1 Detailed Description

Embedded Flash File System, EFFS-STD.

24.355 fsf.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00023 #ifndef _FSF_H_
00024 #define _FSF_H_
00025
00026 /***************************************************************************
00027  *
00028  *            Copyright (c) 2003 by HCC Embedded
00029  *
00030  * This software is copyrighted by and is the sole property of
00031  * HCC.  All rights, title, ownership, or other interests
00032  * in the software remain the property of HCC.  This
00033  * software may only be used in accordance with the corresponding
00034  * license agreement.  Any unauthorized use, duplication, transmission,
00035  * distribution, or disclosure of this software is expressly forbidden.
00036  *
00037  * This Copyright notice may not be removed or modified without prior
00038  * written consent of HCC.
00039  *
00040  * HCC reserves the right to modify this software without notice.
00041  *
00042  * HCC Embedded
00043  * Budapest 1132
00044  * Victor Hugo Utca 11-15
00045  * Hungary
00046  *
00047  * Tel:  +36 (1) 450 1302
00048  * Fax:  +36 (1) 450 1303
00049  * http: www.hcc-embedded.com
00050  * email: info@hcc-embedded.com
00051  *
00052  ***************************************************************************/
00053
00054 #ifdef __cplusplus
00055 extern "C"
00056 {
00057 #endif
00058
00059 #include <file/fsm.h>
00060 #include <file/fwerr.h>
00061 #include <file/port_s.h>
00062 #include <file/udefs.h>
00063
00064     /* ////////////////////////////////////////////////////////////////////////////// */
00065     /*  */
00066     /*  Init Functions   */
00067     /*  */
00068     /* ////////////////////////////////////////////////////////////////////////////// */
00069
00070     extern char *fg_getversion(void);
00071     extern void fg_init(void);
00072     extern int fg_mountdrive(int drivenum, void *buffer, long buffsize, FS_DRVMOUNT mountfunc,
      FS_PHYGETID phyfunc);
00073     extern int fg_unmountdrive(int drvnum);
00074     extern int fg_format(FS_MULTI *fm, int drivenum);
00075     extern int fg_getfreespace(FS_MULTI *fm, int drivenum, FS_SPACE *space);
00076     extern int fg_get_drive_list(int *buf);
00077     extern int fg_get_drive_count(void);
00078
00079     /* ////////////////////////////////////////////////////////////////////////////// */
00080     /*  */
00081     /*  Directory handler functions  */
00082     /*  */
00083     /* ////////////////////////////////////////////////////////////////////////////// */
00084
00085     extern int fg_getdrive(FS_MULTI *fm);
00086     extern int fg_chdrive(FS_MULTI *fm, int drivenum);
00087
00088     extern int fg_getcwd(FS_MULTI *fm, char *buffer, int maxlen);
00089     extern int fg_getdcwd(FS_MULTI *fm, int drivenum, char *buffer, int maxlen);
00090
00091     extern int fg_mkdir(FS_MULTI *fm, const char *dirname);
00092     extern int fg_chdir(FS_MULTI *fm, const char *dirname);
00093     extern int fg_rmdir(FS_MULTI *fm, const char *dirname);
00094
00095     extern int fg_setlabel(FS_MULTI *fm, int drivenum, const char *label);
00096     extern int fg_getlabel(FS_MULTI *fm, int drivenum, char *label, long len);
00097
00098     /* ////////////////////////////////////////////////////////////////////////////// */
00099     /*  */
00100     /*  files functions  */
00101     /*  */
00102     /* ////////////////////////////////////////////////////////////////////////////// */
```

```
00103
00104     extern int fg_rename(FS_MULTI *fm, const char *filename, const char *newname);
00105     extern int fg_move(FS_MULTI *fm, const char *filename, const char *newname);
00106     extern int fg_delete(FS_MULTI *fm, const char *filename);
00107
00108     extern long fg_filelength(FS_MULTI *fm, const char *filename);
00109
00110     extern int fg_findfirst(FS_MULTI *fm, const char *filename, FS_FIND *find);
00111     extern int fg_findnext(FS_MULTI *fm, FS_FIND *find);
00112
00113     /* ////////////////////////////////////////////////////////////////////////////// */
00114     /*  */
00115     /*  file read/write functions    */
00116     /*  */
00117     /* ////////////////////////////////////////////////////////////////////////////// */
00118
00119     extern int registerFSFILE(FS_FILE *file);
00120
00121     extern FS_FILE *fg_open(FS_MULTI *fm, const char *filename, const char *mode);
00122     extern int fg_close(FS_MULTI *fm, FS_FILE *filehandle);
00123     extern long fg_write(FS_MULTI *fm, const void *buf, long size, long size_st, FS_FILE *filehandle);
00124     extern long fg_read(FS_MULTI *fm, void *buf, long size, long size_st, FS_FILE *filehandle);
00125     extern int fg_seek(FS_MULTI *fm, FS_FILE *filehandle, long offset, long whence);
00126     extern long fg_tell(FS_MULTI *fm, FS_FILE *filehandle);
00127     extern int fg_eof(FS_MULTI *fm, FS_FILE *filehandle);
00128     extern int fg_rewind(FS_MULTI *fm, FS_FILE *filehandle);
00129     extern int fg_putc(FS_MULTI *fm, int ch, FS_FILE *filehandle);
00130     extern int fg_getc(FS_MULTI *fm, FS_FILE *filehandle);
00131     extern int fg_flush(FS_MULTI *fm, FS_FILE *filehandle);
00132     extern int fg_seteof(FS_MULTI *fm, FS_FILE *filehandle);
00133
00134     extern int fg_settimedate(FS_MULTI *fm, const char *filename, unsigned short ctime, unsigned short
      cdate);
00135     extern int fg_gettimedate(FS_MULTI *fm, const char *filename, unsigned short *ctime, unsigned
      short *cdate);
00136     extern int fg_getpermission(FS_MULTI *fm, const char *filename, unsigned long *psecure);
00137     extern int fg_setpermission(FS_MULTI *fm, const char *filename, unsigned long secure);
00138
00139     extern FS_FILE *fg_truncate(FS_MULTI *fm, const char *filename, unsigned long length);
00140
00147 /* Beginning of file */
00148 #ifdef SEEK_SET
00149 #define FS_SEEK_SET SEEK_SET
00150 #else
00151 #define FS_SEEK_SET 0
00152 #endif
00153
00154 /* Current position of file pointer */
00155 #ifdef SEEK_CUR
00156 #define FS_SEEK_CUR SEEK_CUR
00157 #else
00158 #define FS_SEEK_CUR 1
00159 #endif
00160
00161 /* End of file */
00162 #ifdef SEEK_END
00163 #define FS_SEEK_END SEEK_END
00164 #else
00165 #define FS_SEEK_END 2
00166 #endif
00169 #ifdef HCC_UNICODE
00170     extern int fg_wgetcwd(FS_MULTI *fm, W_CHAR *buffer, int maxlen);
00171     extern int fg_wgetdcwd(FS_MULTI *fm, int drivenum, W_CHAR *buffer, int maxlen);
00172     extern int fg_wmkdir(FS_MULTI *fm, const W_CHAR *dirname);
00173     extern int fg_wchdir(FS_MULTI *fm, const W_CHAR *dirname);
00174     extern int fg_wrmdir(FS_MULTI *fm, const W_CHAR *dirname);
00175     extern int fg_wrename(FS_MULTI *fm, const W_CHAR *filename, const W_CHAR *newname);
00176     extern int fg_wmove(FS_MULTI *fm, const W_CHAR *filename, const W_CHAR *newname);
00177     extern int fg_wdelete(FS_MULTI *fm, const W_CHAR *filename);
00178     extern long fg_wfilelength(FS_MULTI *fm, const W_CHAR *filename);
00179     extern int fg_wfindfirst(FS_MULTI *fm, const W_CHAR *filename, FS_WFIND *find);
00180     extern int fg_wfindnext(FS_MULTI *fm, FS_WFIND *find);
00181     extern FS_FILE *fg_wopen(FS_MULTI *fm, const W_CHAR *filename, const W_CHAR *mode);
00182     extern int fg_wsettimedate(FS_MULTI *fm, const W_CHAR *filename, unsigned short ctime, unsigned
      short cdate);
00183     extern int fg_wgettimedate(FS_MULTI *fm, const W_CHAR *filename, unsigned short *ctime, unsigned
      short *cdate);
00184     extern int fg_wgetpermission(FS_MULTI *fm, const W_CHAR *filename, unsigned long *psecure);
00185     extern int fg_wsetpermission(FS_MULTI *fm, const W_CHAR *filename, unsigned long secure);
00186     extern FS_FILE *fg_wtruncate(FS_MULTI *fm, const W_CHAR *filename, unsigned long length);
00187 #endif
00188
00189     /* ////////////////////////////////////////////////////////////////////////////// */
00190     /*  */
00191     /*  internal common functions for secure parts  */
00192     /*  */
00193     /* ////////////////////////////////////////////////////////////////////////////// */
```

```
00194
00195     extern int _fg_flush(FS_VOLUMEINFO *vi);
00196     extern int _fg_getvolumeinfo(FS_MULTI *fm, int drivenum, FS_VOLUMEINFO **pvi);
00197     extern int _fg_findpath(FS_VOLUMEINFO *vi, FS_NAME *fsname);
00198     extern int _fg_findfile(FS_VOLUMEINFO *vi, W_CHAR *name, unsigned short dirnum, unsigned short
      *pdirnum);
00199     extern int _fg_addentry(FS_VOLUMEINFO *vi, FS_NAME *fsname, FS_DIRENTRY **pde);
00200     extern int _fg_find(FS_VOLUMEINFO *vi, FS_NAME *fsname, FS_DIRENTRY **pde, unsigned short
      *pdirnum);
00201     extern int _fg_findfilewc(FS_VOLUMEINFO *vi, W_CHAR *name, unsigned short dirnum, unsigned short
      *pdirnum, unsigned short startpos);
00202     extern void _fg_setdiscsectors(FS_VOLUMEINFO *vi, unsigned short sector);
00203     extern int _fg_getsector(FS_VOLUMEINFO *vi, long secnum, void *data, long offset, long datalen);
00204     extern void _fg_removedename(FS_VOLUMEINFO *vi, FS_DIRENTRY *de);
00205     extern int _fg_setdename(W_CHAR *s, FS_VOLUMEINFO *vi, FS_DIRENTRY *de);
00206     extern void _fg_getdename(W_CHAR *s, FS_VOLUMEINFO *vi, FS_DIRENTRY *de);
00207     extern int _fg_namecheckwc(const W_CHAR *name, const W_CHAR *s);
00208     extern int _fg_copychainintomirror(FS_VOLUMEINFO *vi, FS_FILEINT *f, FS_DIRENTRY *de);
00209     extern void _fg_cleanupfile(FS_VOLUMEINFO *vi, FS_FILEINT *f);
00210     extern int _fg_fseek(FS_VOLUMEINFO *vi, FS_MULTI *fm, FS_FILEINT *f, long offset);
00211     extern int _fg_checkfilelock(FS_VOLUMEINFO *vi, FS_DIRENTRY *de, long m_mode);
00212
00213     extern FS_FILESYSTEM fg_filesystem;
00214
00215 #include "fsmf.h"
00216
00217 #define fs_getversion fsm_getversion
00218 #define fs_init fsm_init
00219 #define fs_mountdrive(drivenum, buffer, buffsize, mountfunc, phyfunc) fsm_mountdrive(drivenum, buffer,
      buffsize, mountfunc, phyfunc)
00220 #define fs_unmountdrive(drvnum) fsm_unmountdrive(drvnum)
00221 #define fs_format(drivenum) fsm_format(drivenum)
00222
00232 #define fs_getfreespace(drivenum, space) fsm_getfreespace(drivenum, space)
00233 #define fs_releaseFS(ID) fsm_releaseFS(ID)
00234 #define fs_get_drive_list(buf) fsm_get_drive_list(buf)
00235 #define fs_get_drive_count fsm_get_drive_count
00236 #define fs_getdrive fsm_getdrive
00237 #define fs_chdrive(drivenum) fsm_chdrive(drivenum)
00238 #define fs_getcwd(buffer, maxlen) fsm_getcwd(buffer, maxlen)
00239 #define fs_getdcwd(drivenum, buffer, maxlen) fsm_getdcwd(drivenum, buffer, maxlen)
00240
00252 #define fs_mkdir(dirname) fsm_mkdir(dirname)
00253
00265 #define fs_chdir(dirname) fsm_chdir(dirname)
00266
00281 #define fs_rmdir(dirname) fsm_rmdir(dirname)
00282 #define fs_setlabel(drivenum, label) fsm_setlabel(drivenum, label)
00283 #define fs_getlabel(drivenum, label, len) fsm_getlabel(drivenum, label, len)
00284 #define fs_rename(filename, newname) fsm_rename(filename, newname)
00285 #define fs_move(filename, newname) fsm_move(filename, newname)
00286
00297 #define fs_delete(filename) fsm_delete(filename)
00298 #define fs_filelength(filename) fsm_filelength(filename)
00299
00314 #define fs_findfirst(filename, find) fsm_findfirst(filename, find)
00315
00330 #define fs_findnext(find) fsm_findnext(find)
00331
00354 #define fs_open(filename, mode) fsm_open(filename, mode)
00355
00364 #define fs_close(filehandle) fsm_close(filehandle)
00365
00376 #define fs_write(buf, size, size_st, filehandle) fsm_write(buf, size, size_st, filehandle)
00377
00388 #define fs_read(buf, size, size_st, filehandle) fsm_read(buf, size, size_st, filehandle)
00389
00400 #define fs_seek(filehandle, offset, whence) fsm_seek(filehandle, offset, whence)
00401 #define fs_tell(filehandle) fsm_tell(filehandle)
00402
00411 #define fs_eof(filehandle) fsm_eof(filehandle)
00412 #define fs_seteof(filehandle) fsm_seteof(filehandle)
00413
00422 #define fs_rewind(filehandle) fsm_rewind(filehandle)
00423 #define fs_putc(ch, filehandle) fsm_putc(ch, filehandle)
00424 #define fs_getc(filehandle) fsm_getc(filehandle)
00425 #define fs_flush(filehandle) fsm_flush(filehandle)
00426
00437 #define fs_settimedate(filename, ctime, cdate) fsm_settimedate(filename, ctime, cdate)
00438
00449 #define fs_gettimedate(filename, pctime, pcdate) fsm_gettimedate(filename, pctime, pcdate)
00450
00451 #define fs_getpermission(filename, psecure) fsm_getpermission(filename, psecure)
00452 #define fs_setpermission(filename, secure) fsm_setpermission(filename, secure)
00453 #define fs_truncate(filename, length) fsm_truncate(filename, length)
00454
00455 #ifdef HCC_UNICODE
```

```
00456 #define fs_wgetcwd(buffer, maxlen) fsm_wgetcwd(buffer, maxlen)
00457 #define fs_wgetdcwd(drivenum, buffer, maxlen) fsm_wgetdcwd(drivenum, buffer, maxlen)
00458 #define fs_wmkdir(dirname) fsm_wmkdir(dirname)
00459 #define fs_wchdir(dirname) fsm_wchdir(dirname)
00460 #define fs_wrmdir(dirname) fsm_wrmdir(dirname)
00461 #define fs_wrename(filename, newname) fsm_wrename(filename, newname)
00462 #define fs_wmove(filename, newname) fsm_wmove(filename, newname)
00463 #define fs_wdelete(filename) fsm_wdelete(filename)
00464 #define fs_wfilelength(filename) fsm_wfilelength(filename)
00465 #define fs_wfindfirst(filename, find) fsm_wfindfirst(filename, find)
00466 #define fs_wfindnext(find) fsm_wfindnext(find)
00467 #define fs_wopen(filename, mode) fsm_wopen(filename, mode)
00468 #define fs_wsettimedate(filename, ctime, cdate) fsm_wsettimedate(filename, ctime, cdate)
00469 #define fs_wgettimedate(filename, ctime, cdate) fsm_wgettimedate(filename, ctime, cdate)
00470 #define fs_wgetpermission(filename, psecure) fsm_wgetpermission(filename, psecure)
00471 #define fs_wsetpermission(filename, secure) fsm_wsetpermission(filename, secure)
00472 #define fs_wtruncate(filename, length) fsm_wtruncate(filename, length)
00473 #endif
00474
00475 #ifdef __cplusplus
00476 }
00477 #endif
00478
00479 /****************************************************************************
00480  *
00481  * End of fsf.h
00482  *
00483  ****************************************************************************/
00484
00485 #endif /* _FSF_H_ */
00486
```

24.356 fsm.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FSM_H_
00004 #define _FSM_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #include <file/port_s.h>
00035 #include <file/udefs.h>
00036
00037 #ifdef __cplusplus
00038 extern "C"
00039 {
00040 #endif
00041
00042 #define FS_MAXDENAME 13 /* maximum direntry name */
00043
00044     /*  Directory entry definition  */
00045
00046     typedef struct
00047     {
00048         char attr;                  /* attribute of the file */
00049         W_CHAR lname[FS_MAXDENAME]; /* file name */
00050         unsigned short nlfn;        /* next lfn entry */
00051
```

```
00052         unsigned short ctime; /* creation time */
00053         unsigned short cdate; /* creation date */
00054
00055         unsigned short sector; /* start sector */
00056         unsigned short dirnum; /* directory relative number */
00057
00058         long len; /* length of file */
00059
00060         unsigned long secure; /* security code */
00061     } FS_DIRENTRY;
00062
00063     typedef struct
00064     {
00065         char attr;            /* attribute of the file */
00066         unsigned short nlfn1; /* next lfn entry */
00067         unsigned short nlfn2; /* next lfn entry */
00068     } FS_DIRENTRY_LFN;
00069
00070 #define FS_MAXLFN ((sizeof(FS_DIRENTRY) - sizeof(FS_DIRENTRY_LFN)) » 1)
00071
00072 #define FS_MAXLNAME (FS_MAXDENAME + 4 * FS_MAXLFN) /*  maximum name length -> 13+4*13 => 65  */
00073
00074     /*  definitions for file/entry attribute  */
00075
00076 #define FS_ATTR_DE 0x01         /* entry is used for direntry */
00077 #define FS_ATTR_DIR 0x02        /* directory */
00078 #define FS_ATTR_LFN1 0x04       /* structure holds long file name in the 1st half */
00079 #define FS_ATTR_LFN2 0x08       /* structure holds long file name in the 2nd half */
00080 #define FS_ATTR_LFN1NXT 0x10    /* there is next entry of 1 on next bottom */
00081 #define FS_ATTR_LFN1NXTTOP 0x20 /* there is next entry of 1 on next top */
00082 #define FS_ATTR_LFN2NXT 0x40    /* there is next entry of 2 on next bottom */
00083 #define FS_ATTR_LFN2NXTTOP 0x80 /* there is next entry of 2 on next top */
00084
00085 #define FS_ATTR_ALLLFN1 (FS_ATTR_LFN1 | FS_ATTR_LFN1NXT | FS_ATTR_LFN1NXTTOP)
00086 #define FS_ATTR_ALLLFN2 (FS_ATTR_LFN2 | FS_ATTR_LFN2NXT | FS_ATTR_LFN2NXTTOP)
00087     /*  definitions for ctime  */
00088
00089 #define FSSEC_ATTR_ARC (0x20UL « (31 - 6))
00090 #define FSSEC_ATTR_DIR (0x10UL « (31 - 6))
00091 #define FSSEC_ATTR_VOLUME (0x08UL « (31 - 6))
00092 #define FSSEC_ATTR_SYSTEM (0x04UL « (31 - 6))
00093 #define FSSEC_ATTR_HIDDEN (0x02UL « (31 - 6))
00094 #define FSSEC_ATTR_READONLY (0x01UL « (31 - 6))
00095
00096 #define FS_CTIME_SEC_SHIFT 0
00097 #define FS_CTIME_SEC_MASK 0x001f /* 0-30 in 2seconds */
00098 #define FS_CTIME_MIN_SHIFT 5
00099 #define FS_CTIME_MIN_MASK 0x07e0 /* 0-59  */
00100 #define FS_CTIME_HOUR_SHIFT 11
00101 #define FS_CTIME_HOUR_MASK 0xf800 /* 0-23 */
00102
00103     /*  definitions for cdate  */
00104
00105 #define FS_CDATE_DAY_SHIFT 0
00106 #define FS_CDATE_DAY_MASK 0x001f /* 0-31 */
00107 #define FS_CDATE_MONTH_SHIFT 5
00108 #define FS_CDATE_MONTH_MASK 0x01e0 /* 1-12 */
00109 #define FS_CDATE_YEAR_SHIFT 9
00110 #define FS_CDATE_YEAR_MASK 0xfe00 /* 0-119 (1980+value) */
00111
00112     /*  definitions for dirnum variable  */
00113
00114 #define FS_DIR_ROOT ((unsigned short)0xffff)
00115 #define FS_DIR_LABEL ((unsigned short)0xfff0)
00116
00117     typedef struct
00118     {
00119         unsigned long total;
00120         unsigned long free;
00121         unsigned long used;
00122         unsigned long bad;
00123     } FS_SPACE;
00124
00125     /*  struct for FS_NAME  */
00126
00127     typedef struct
00128     {
00129         int drivenum;             /*  0-A 1-B 2-C  */
00130         W_CHAR path[FS_MAXPATH];  /*   /directory1/dir2/   */
00131         W_CHAR lname[FS_MAXPATH]; /*  filename  */
00132         unsigned short dirnum;    /*  0xffff-root other case in subdir n  */
00133     } FS_NAME;
00134
00135     /*  struct for find file  */
00136
00137     typedef struct
00138     {
```

```
00139         char attr;                /* attribute of the file/entry */
00140         char filename[FS_MAXPATH]; /* file name+ext */
00141
00142         unsigned short ctime; /* creation time */
00143         unsigned short cdate; /* creation date */
00144
00145         long len; /* length of file */
00146
00147         unsigned long secure; /* secure */
00148
00149         FS_NAME findfsname;     /* find properties */
00150         unsigned short findpos; /* find position */
00151     } FS_FIND;
00152
00153 #ifdef HCC_UNICODE
00154     typedef struct
00155     {
00156         char attr;                  /* attribute of the file/entry */
00157         W_CHAR filename[FS_MAXPATH]; /* file name+ext */
00158
00159         unsigned short ctime; /* creation time */
00160         unsigned short cdate; /* creation date */
00161
00162         long len; /* length of file */
00163
00164         unsigned long secure; /* secure */
00165
00166         FS_NAME findfsname;     /* find properties */
00167         unsigned short findpos; /* find position */
00168     } FS_WFIND;
00169 #endif
00170
00171     /*  definitions for FS_FILE  */
00172
00173     typedef struct
00174     {
00175         long reference; /* reference which fileint used */
00176     } FS_FILE;
00177
00178     /*  definitions for FS_FILE internally used  */
00179
00180     typedef struct
00181     {
00182         FS_FILE file;
00183         FS_DIRENTRY *direntry; /* link to directory list */
00184         long pos;              /* current position for read, file size for write */
00185         long relpos;           /*  relative position in a sector  */
00186         int mode;              /* mode to open 0-close, 1-read, 2-write/append */
00187         int drivenum;          /* drive number */
00188         char *buffer;          /* rd/write buffer */
00189         int modified;          /* if write buffer is modified */
00190
00191         unsigned short *sector;     /*  this points where to write/read next sector info  */
00192         unsigned short sectorstart; /* after file is closed this has to be copied into direntry */
00193
00194         unsigned short *discard;     /*  this points where to write/read last discard sector is  */
00195         unsigned short discardstart; /* after file is closed this has to be set as discardable */
00196         long len;                    /* file size, this is copied after fs_close */
00197         int loaded;                  /* signalled if sector is loaded */
00198     } FS_FILEINT;
00199
00200     /*  definitions for fs_file mode  */
00201
00202 #define FS_FILE_CLOSE 0
00203 #define FS_FILE_RD 1
00204 #define FS_FILE_RDP 2
00205 #define FS_FILE_WR 3
00206 #define FS_FILE_WRP 4
00207 #define FS_FILE_A 5
00208 #define FS_FILE_AP 6
00209 #define FS_FILE_ABORT 7
00210
00211     /*  definitions for FLASH physical functions   */
00212
00213     typedef int (*FS_PHYREAD)(void *data, long block, long blockrel, long datalen);
00214     typedef int (*FS_PHYERASE)(long block);
00215     typedef int (*FS_PHYWRITE)(void *data, long block, long relsector, long size, long sdata);
00216     typedef int (*FS_PHYVERIFY)(void *data, long block, long relsector, long size, long sdata);
00217     typedef int (*FS_PHYCHECK)(long block);
00218     typedef long (*FS_PHYSIGN)(long block);
00219     typedef int (*FS_PHYCACHE)(void *data, long block, long page, long pagenum, long sdata);
00220     typedef int (*FS_PHYBLKCPY)(long destblock, long soublock);
00221
00222     /*  definitions for FLASH info and phy  */
00223
00224     typedef struct
00225     {
```

```
00226         long maxblock;               /* maximum number of block can be used */
00227         long blocksize;              /* block size in bytes */
00228         long sectorsize;             /* sector size wanted to use (less than block size */
00229         long sectorperblock;         /* sector per block (block size/sector size); */
00230         long blockstart;             /* where relative physically block start */
00231         long descsize;               /* max size of fat+directory+block index */
00232         long descblockstart;         /* 1st block which is used for descriptor above (NOR) */
00233         long descblockend;           /* last block which is used for descriptor above (NOR) */
00234         long separatedir;            /* if directory used separatelly from FAT (NAND) */
00235         long cacheddescsize;         /* cached descriptor size in descriptor < descsize (NOR) */
00236         long cachedpagenum;          /* cached pagenum (page/block NAND) */
00237         long cachedpagesize;         /* cached page size (page size.
      cachedpagenum*cachedpagesize=blocksize */
00238         FS_PHYREAD ReadFlash;        /* read content */
00239         FS_PHYERASE EraseFlash;      /* erase a block */
00240         FS_PHYWRITE WriteFlash;      /* write content */
00241         FS_PHYVERIFY VerifyFlash;    /* verify content */
00242         FS_PHYCHECK CheckBadBlock;   /* check if block is bad block (NAND) */
00243         FS_PHYSIGN GetBlockSignature; /* get block signature data (NAND) */
00244         FS_PHYCACHE WriteVerifyPage; /* Write and verify page (NAND) */
00245         FS_PHYBLKCPY BlockCopy;      /* HW/SW accelerated block copy in physical (NAND/NOR) optional
      */
00246         unsigned char *chkeraseblk;  /* buffer for preerasing blocks optional */
00247     } FS_FLASH;
00248
00249     typedef int (*FS_PHYGETID)(FS_FLASH *flash);
00250
00251     /*  definitions for fat descriptor  */
00252
00253     typedef struct
00254     {
00255         unsigned long crc32;     /* crc of this structure */
00256         unsigned long reference; /* reference counter */
00257         long nextdesc;           /* which desc needs to be written */
00258         unsigned long dircrc32;  /* directory crc32 */
00259         /*  FAT + Direntries + Block index is allocated here, the extra's size is flash->descsize  */
00260     } FS_FATDESC;
00261
00262     typedef struct
00263     {
00264         char *desc;              /* NOR+NAND */
00265         char *changes;           /* next changes pointer NAND+NOR */
00266         unsigned long reference; /* reference counter NAND+NOR */
00267         long free;               /* free space in cache NAND+NOR */
00268         long currdescnum;        /* current descriptor block for NOR */
00269     } FS_WRCACHE;
00270     /*  definitions for volume info   */
00271
00272     typedef struct
00273     {
00274         int drivenum;              /* which drive is to belong 0-a, 1-b, ... */
00275         char *buffer;              /* next alloc pointer for alloc data function */
00276         long freemem;              /* free memory space on alloc */
00277         long usedmem;              /* used memory */
00278         long maxsectornum;         /* maximum sector used */
00279         unsigned short *fat;       /* pointer to memory FAT (data found after volumeinfo) */
00280         unsigned short *fatmirror; /* pointer to memory FAT (data found after volumeinfo) */
00281
00282         W_CHAR *cwd; /* current working folder in this volume */
00283
00284         unsigned int maxdirentry;   /* directory entry used */
00285         FS_DIRENTRY *direntries;    /* pointer to dirinfo */
00286         long sectorsize;            /* sector size */
00287         int maxfile;                /* maximum number of used file */
00288         FS_FILEINT *files;          /* s_fileint pointers */
00289         FS_FLASH *flash;            /* flash device properties */
00290         FS_FATDESC *fatdesc;        /* user driver data 1 */
00291         char *ramdrivedata;         /* ramdrive data pointer */
00292         unsigned char *zerosector;  /* nandflash zero sector */
00293         long *wearlevel;            /* used for wear leveling */
00294         long resetwear;             /* signal if wear leveling useable or need resetting */
00295         long maxfat;                /* maximum number of fat */
00296         long currfat;               /* current fat */
00297         long prevfat;               /* previous fat */
00298         long *fatbits;              /* preerased blocks sectors state */
00299         long fatbitsblock;          /* preerased blocks logical number */
00300         unsigned short *blockindex; /* block orders (maxblock used size); */
00301         char *rdbuffer;             /* temporary block reading then writing (block size) */
00302         FS_WRCACHE cache;           /* descriptor cache */
00303         long laststaticwear;        /* last static weared block */
00304         long staticcou;             /* static counter for period counter */
00305         FS_MUTEX_TYPE mutex;        /* for multitasking */
00306     } FS_VOLUMEINFO;
00307
00308     /*  definitions for multicwd   */
00309
00310 #if (!FS_CAPI_USED)
```

```
00311     typedef struct
00312     {
00313         long ID;
00314         int fs_curdrive; /* current drive */
00315         struct
00316         {
00317             W_CHAR cwd[FS_MAXPATH]; /* current working folder in this volume */
00318         } fs_vols[FS_MAXVOLUME];
00319 
00320         FS_MUTEX_TYPE *pmutex; /* for multitasking */
00321 
00322     } FS_MULTI;
00323 #endif
00324 
00325     extern void fl_releaseFS(long ID);
00326 
00327     extern long fs_gettaskID(void);
00328 
00329     /*  definitions for driver functions   */
00330 
00331     typedef int (*FS_DRVFUNC1)(FS_VOLUMEINFO *vi);
00332     typedef int (*FS_DRVFUNC2)(FS_VOLUMEINFO *vi, FS_FILEINT *file, void *data, long datalen);
00333     typedef int (*FS_DRVFUNC4)(FS_VOLUMEINFO *vi, long secnum, void *data, long offset, long datalen);
00334 
00335     /*  definitions for volume descriptor  */
00336 
00337     typedef struct
00338     {
00339         FS_DRVFUNC1 storefat;    /* function pointer */
00340         FS_DRVFUNC2 storesector; /* function pointer */
00341         FS_DRVFUNC4 getsector;   /* function pointer */
00342         FS_DRVFUNC1 format;      /* function pointer */
00343         FS_VOLUMEINFO *vi;       /* volumeinfo pointer */
00344         int state;               /* state of this volume  */
00345     } FS_VOLUMEDESC;
00346 
00347     /*  definitions for volumedesc state  */
00348 
00349 #define FS_VOL_OK 0           /* mounted, formatted */
00350 #define FS_VOL_NOTMOUNT 1     /* not mounted (init value) */
00351 #define FS_VOL_NOTFORMATTED 2 /* not formatted */
00352 #define FS_VOL_NOMEMORY 3     /* not enougth memory */
00353 #define FS_VOL_NOMORE 4       /* no more drive availabale */
00354 #define FS_VOL_DRVERROR 5     /* driver error */
00355 
00356     /*  definitions for drive function mount  */
00357 
00358     typedef int (*FS_DRVMOUNT)(FS_VOLUMEDESC *vd, FS_PHYGETID phyfunc);
00359 
00360     /*  definitions for file system  */
00361 
00362     typedef struct
00363     {
00364         FS_VOLUMEDESC vd[FS_MAXVOLUME]; /* volumes */
00365     } FS_FILESYSTEM;
00366 
00367     /* define fat entries */
00368 
00369 #define FS_FAT_FREE ((unsigned short)0x0FFFF)    /* - free of used */
00370 #define FS_FAT_EOF ((unsigned short)0x0FFF0)     /* - end of file */
00371 #define FS_FAT_NOTUSED ((unsigned short)0x0FFF1) /* - not useable (maybe bad block or reserved) */
00372 #define FS_FAT_DISCARD ((unsigned short)0x0FFF2) /* - needs to be discard */
00373 #define FS_FAT_CHBLK ((unsigned short)0x0FFF3)   /* - cache block */
00374 #define FS_FAT_DIR ((unsigned short)0x0FFF8)     /* - directory entry, if separated */
00375 
00376     /*  crc defs  */
00377 
00378 #define FS_CRCINIT 0xffffffffL
00379 
00380     /*  functions for middle layer file system  */
00381 
00382     extern void *fsm_allocdata(FS_VOLUMEINFO *vi, long size);
00383     extern int fsm_checkname(W_CHAR *lname);
00384     extern int fsm_checknamewc(const W_CHAR *lname);
00385     extern long fsm_setnameext(char *s, char *name, char *ext);
00386     extern int fsm_setmaxfile(FS_VOLUMEINFO *vi, long maxfile);
00387     extern int fsm_setsectorsize(FS_VOLUMEINFO *vi, long sectorsize);
00388     extern void fsm_memcpy(void *d, void *s, long len);
00389     extern unsigned long fsm_calccrc32(unsigned long dwcrc, const void *vbuf, unsigned long dwlen);
00390     extern int fsm_findfreeblock(FS_VOLUMEINFO *vi, unsigned short *sector);
00391     extern void fsm_memset(void *d, unsigned char fill, long len);
00392     extern int fsm_findfreesector(FS_VOLUMEINFO *vi, unsigned short *badsector);
00393     extern void fsm_swapbadblock(FS_VOLUMEINFO *vi, unsigned short badsector);
00394     extern void fsm_wearleveling(FS_VOLUMEINFO *vi);
00395     extern void fsm_addsectorchain(FS_VOLUMEINFO *vi, FS_FILEINT *file, unsigned short sector);
00396     extern long fsm_checksectorfree(FS_VOLUMEINFO *vi, long sector);
00397     extern long fsm_checksectorbad(FS_VOLUMEINFO *vi, long sector);
```

```
00398     extern unsigned long _fs_checkfreeblocks(FS_VOLUMEINFO *vi, unsigned long sbnum);
00399     extern W_CHAR _fsm_toupper(W_CHAR ch);
00400 #ifdef HCC_UNICODE
00401     extern W_CHAR *_fsm_towchar(W_CHAR *nconv, const char *s);
00402     extern void _fsm_fromwchar(char *d, W_CHAR *s);
00403 #endif
00404 #ifdef __cplusplus
00405 }
00406 #endif
00407
00408 /****************************************************************************
00409  *
00410  * End of fsm.h
00411  *
00412  ****************************************************************************/
00413
00414 #endif /* _FSM_H_ */
```

24.357 fsmf.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FSMF_H_
00004 #define _FSMF_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #ifdef __cplusplus
00035 extern "C"
00036 {
00037 #endif
00038
00039     /* ////////////////////////////////////////////////////////////////////////////////// */
00040     /*  */
00041     /*  Init Functions   */
00042     /*  */
00043     /* ////////////////////////////////////////////////////////////////////////////////// */
00044
00045 #define fsm_getversion fg_getversion
00046 #define fsm_init fg_init
00047 #define fsm_mountdrive fg_mountdrive
00048 #define fsm_unmountdrive fg_unmountdrive
00049     extern int fsm_getfreespace(int drivenum, FS_SPACE *space);
00050     extern void fsm_releaseFS(long ID);
00051
00052     extern int fsm_format(int drivenum);
00053     extern int fsm_get_drive_list(int *buf);
00054     extern int fsm_get_drive_count(void);
00055
00056     /* ////////////////////////////////////////////////////////////////////////////////// */
00057     /*  */
00058     /*  Directory handler functions  */
00059     /*  */
00060     /* ////////////////////////////////////////////////////////////////////////////////// */
00061
00062     extern int fsm_getdrive(void);
00063     extern int fsm_chdrive(int drivenum);
00064
00065     extern int fsm_getcwd(char *buffer, int maxlen);
```

```
00066     extern int fsm_getdcwd(int drivenum, char *buffer, int maxlen);
00067
00068     extern int fsm_mkdir(const char *dirname);
00069     extern int fsm_chdir(const char *dirname);
00070     extern int fsm_rmdir(const char *dirname);
00071
00072     /* ////////////////////////////////////////////////////////////////////////////// */
00073     /*  */
00074     /*  files functions  */
00075     /*  */
00076     /* ////////////////////////////////////////////////////////////////////////////// */
00077
00078     extern int fsm_rename(const char *filename, const char *newname);
00079     extern int fsm_move(const char *filename, const char *newname);
00080     extern int fsm_delete(const char *filename);
00081
00082     extern long fsm_filelength(const char *filename);
00083
00084     extern int fsm_findfirst(const char *filename, FS_FIND *find);
00085     extern int fsm_findnext(FS_FIND *find);
00086
00087     extern int fsm_getpermission(const char *filename, unsigned long *psecure);
00088     extern int fsm_setpermission(const char *filename, unsigned long secure);
00089
00090     /* ////////////////////////////////////////////////////////////////////////////// */
00091     /*  */
00092     /*  file read/write functions    */
00093     /*  */
00094     /* ////////////////////////////////////////////////////////////////////////////// */
00095
00096     extern FS_FILE *fsm_open(const char *filename, const char *mode);
00097     extern int fsm_close(FS_FILE *filehandle);
00098     extern long fsm_write(const void *buf, long size, long size_st, FS_FILE *filehandle);
00099     extern long fsm_read(void *buf, long size, long size_st, FS_FILE *filehandle);
00100     extern int fsm_seek(FS_FILE *filehandle, long offset, long whence);
00101     extern long fsm_tell(FS_FILE *filehandle);
00102     extern int fsm_eof(FS_FILE *filehandle);
00103     extern int fsm_rewind(FS_FILE *filehandle);
00104     extern int fsm_putc(int ch, FS_FILE *filehandle);
00105     extern int fsm_getc(FS_FILE *filehandle);
00106     extern int fsm_flush(FS_FILE *filehandle);
00107     extern int fsm_seteof(FS_FILE *filehandle);
00108
00109     extern int fsm_settimedate(const char *filename, unsigned short ctime, unsigned short cdate);
00110     extern int fsm_gettimedate(const char *filename, unsigned short *ctime, unsigned short *cdate);
00111     extern int fsm_getlabel(int drivenum, char *label, long len);
00112     extern int fsm_setlabel(int drivenum, const char *label);
00113     extern FS_FILE *fsm_truncate(const char *filename, unsigned long length);
00114
00115 #ifdef HCC_UNICODE
00116     extern int fsm_wgetcwd(W_CHAR *buffer, int maxlen);
00117     extern int fsm_wgetdcwd(int drivenum, W_CHAR *buffer, int maxlen);
00118     extern int fsm_wmkdir(const W_CHAR *dirname);
00119     extern int fsm_wchdir(const W_CHAR *dirname);
00120     extern int fsm_wrmdir(const W_CHAR *dirname);
00121     extern int fsm_wrename(const W_CHAR *filename, const W_CHAR *newname);
00122     extern int fsm_wmove(const W_CHAR *filename, const W_CHAR *newname);
00123     extern int fsm_wdelete(const W_CHAR *filename);
00124     extern long fsm_wfilelength(const W_CHAR *filename);
00125     extern int fsm_wfindfirst(const W_CHAR *filename, FS_WFIND *find);
00126     extern int fsm_wfindnext(FS_WFIND *find);
00127     extern int fsm_wgetpermission(const W_CHAR *filename, unsigned long *psecure);
00128     extern int fsm_wsetpermission(const W_CHAR *filename, unsigned long secure);
00129     extern FS_FILE *fsm_wopen(const W_CHAR *filename, const W_CHAR *mode);
00130     extern int fsm_wsettimedate(const W_CHAR *filename, unsigned short ctime, unsigned short cdate);
00131     extern int fsm_wgettimedate(const W_CHAR *filename, unsigned short *ctime, unsigned short *cdate);
00132     extern FS_FILE *fsm_wtruncate(const W_CHAR *filename, unsigned long length);
00133 #endif
00134
00135     /*****************************************************************************
00136      *
00137      * internal common functions for reantrancy
00138      *
00139      *****************************************************************************/
00140
00141     extern int _fsm_checksemaphore(FS_MULTI *fm, FS_VOLUMEINFO *vi);
00142     extern void _fsm_releasesemaphore(FS_MULTI *fm);
00143     extern int _fsm_settask(FS_MULTI **fm);
00144
00145 #ifdef __cplusplus
00146 }
00147 #endif
00148
00149 /*****************************************************************************
00150  *
00151  * End of fsmf.h
00152  *
```

```
00153  ****************************************************************************/
00154
00155 #endif /* _FSMF_H_ */
```

24.358 fstaticw.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _FSSTATIC_H_
00004 #define _FSSTATIC_H_
00005
00006 /****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ****************************************************************************/
00033
00034 #include <file/fsf.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041 #define FS_STATIC_DISTANCE 1024 /* distance between statically used and free block */
00042 #define FS_STATIC_PERIOD 1024   /* period, when to check static distance */
00043
00044     extern int fs_staticwear(FS_MULTI *fm, int drvnum); /* for normal */
00045     extern int fsm_staticwear(int drvnum);              /* for multitask */
00046
00047 #define fs_staticwear(drvnum) fsm_staticwear(drvnum)
00048
00049 #ifdef __cplusplus
00050 }
00051 #endif
00052
00053 /****************************************************************************
00054  *
00055  * End of fstatic.h
00056  *
00057  ****************************************************************************/
00058
00059 #endif /* _FSSTATIC_H_ */
```

24.359 nflshdrv.h

```
00001 #ifndef _NFLSHDRV_H_
00002 #define _NFLSHDRV_H_
00003
00004 /****************************************************************************
00005  *
00006  *            Copyright (c) 2003 by HCC Embedded
00007  *
00008  * This software is copyrighted by and is the sole property of
00009  * HCC.  All rights, title, ownership, or other interests
00010  * in the software remain the property of HCC.  This
00011  * software may only be used in accordance with the corresponding
00012  * license agreement.  Any unauthorized use, duplication, transmission,
00013  * distribution, or disclosure of this software is expressly forbidden.
00014  *
```

```
00015  * This Copyright notice may not be removed or modified without prior
00016  * written consent of HCC.
00017  *
00018  * HCC reserves the right to modify this software without notice.
00019  *
00020  * HCC Embedded
00021  * Budapest 1132
00022  * Victor Hugo Utca 11-15
00023  * Hungary
00024  *
00025  * Tel:  +36 (1) 450 1302
00026  * Fax:  +36 (1) 450 1303
00027  * http: www.hcc-embedded.com
00028  * email: info@hcc-embedded.com
00029  *
00030  ******************************************************************************/
00031
00032 #include "fsf.h"
00033
00034 #ifdef __cplusplus
00035 extern "C" {
00036 #endif
00037
00038 #define FS_NAND_RESERVEDBLOCK 3 /* defines reserved block in nandflash  */
00039
00040 extern int fs_mount_nandflashdrive(FS_VOLUMEDESC *vd,FS_PHYGETID phyfunc);
00041 extern long fs_getmem_nandflashdrive(FS_PHYGETID phyfunc);
00042
00043 #ifdef __cplusplus
00044 }
00045 #endif
00046
00047 /*****************************************************************************
00048  *
00049  * end of nflshdrv.h
00050  *
00051  ******************************************************************************/
00052
00053 #endif  /* _NFLSHDRV_H_ */
```

24.360 port_s.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _PORT_S_H_
00004 #define _PORT_S_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  ******************************************************************************/
00033
00034 #include <file/udefs.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041 // Uses native NetBurner memcpy and memset functions
00042 #define USE_NB_MEM_FUNCS
00043
```

```
00044 #ifdef USE_NB_MEM_FUNCS
00045 #include <string.h>
00046 #endif
00047
00048     extern unsigned short fs_getdate(void);
00049     extern unsigned short fs_gettime(void);
00050
00051     extern int fs_mutex_get(FS_MUTEX_TYPE *);
00052     extern int fs_mutex_put(FS_MUTEX_TYPE *);
00053     extern int fs_mutex_create(FS_MUTEX_TYPE *);
00054     extern int fs_mutex_delete(FS_MUTEX_TYPE *);
00055
00056     extern long fs_gettaskID(void);
00057
00058 /*  definitions for ctime  */
00059 #define F_CTIME_SEC_SHIFT 0
00060 #define F_CTIME_SEC_MASK 0x001f /* 0-30 in 2seconds */
00061 #define F_CTIME_MIN_SHIFT 5
00062 #define F_CTIME_MIN_MASK 0x07e0 /* 0-59  */
00063 #define F_CTIME_HOUR_SHIFT 11
00064 #define F_CTIME_HOUR_MASK 0xf800 /* 0-23 */
00065
00066 /*  definitions for cdate  */
00067 #define F_CDATE_DAY_SHIFT 0
00068 #define F_CDATE_DAY_MASK 0x001f /* 0-31 */
00069 #define F_CDATE_MONTH_SHIFT 5
00070 #define F_CDATE_MONTH_MASK 0x01e0 /* 1-12 */
00071 #define F_CDATE_YEAR_SHIFT 9
00072 #define F_CDATE_YEAR_MASK 0xfe00 /* 0-119 (1980+value) */
00073
00074 #ifdef __cplusplus
00075 }
00076 #endif
00077
00078 /*****************************************************************************
00079  *
00080  * end of port_s.h
00081  *
00082  *****************************************************************************/
00083
00084 #endif /* _PORT_S_H_ */
```

24.361 ramdrv_s.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _RAMDRV_S_H_
00004 #define _RAMDRV_S_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  *****************************************************************************/
00033
00034 #include <file/fsf.h>
00035
00036 #ifdef __cplusplus
00037 extern "C"
00038 {
00039 #endif
00040
00041     extern int fs_mount_ramdrive(FS_VOLUMEDESC *vd, FS_PHYGETID phyfunc);
```

```
00042
00043 #ifdef __cplusplus
00044 }
00045 #endif
00046
00047 /*****************************************************************************
00048  *
00049  * end of ramdrv.h
00050  *
00051  *****************************************************************************/
00052
00053 #endif /* _RAMDRV_H_ */
```

24.362 udefs.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _UDEFSSTD_H_
00004 #define _UDEFSSTD_H_
00005
00006 /*****************************************************************************
00007  *
00008  *            Copyright (c) 2003 by HCC Embedded
00009  *
00010  * This software is copyrighted by and is the sole property of
00011  * HCC.  All rights, title, ownership, or other interests
00012  * in the software remain the property of HCC.  This
00013  * software may only be used in accordance with the corresponding
00014  * license agreement.  Any unauthorized use, duplication, transmission,
00015  * distribution, or disclosure of this software is expressly forbidden.
00016  *
00017  * This Copyright notice may not be removed or modified without prior
00018  * written consent of HCC.
00019  *
00020  * HCC reserves the right to modify this software without notice.
00021  *
00022  * HCC Embedded
00023  * Budapest 1132
00024  * Victor Hugo Utca 11-15
00025  * Hungary
00026  *
00027  * Tel:  +36 (1) 450 1302
00028  * Fax:  +36 (1) 450 1303
00029  * http: www.hcc-embedded.com
00030  * email: info@hcc-embedded.com
00031  *
00032  *****************************************************************************/
00033
00034 #ifndef NB_BARE_METAL
00035 extern "C++"
00036 {
00037 #include <nbrtos.h>
00038 }
00039 #endif
00040
00041 #ifdef __cplusplus
00042 extern "C"
00043 {
00044 #endif
00045
00046 /*****************************************************************************
00047  *
00048  * if Common Interface is used then set CAPI_USED to 1
00049  *
00050  *****************************************************************************/
00051 #define FS_CAPI_USED 0
00052
00053     /*****************************************************************************
00054      *
00055      * functions definitions
00056      *
00057      *****************************************************************************/
00058
00059 #if FS_CAPI_USED
00060 #include "../../fw_port.h"
00061 #else
00062
00063 /*****************************************************************************
00064  *
00065  * volumes definitions
00066  *
00067  *****************************************************************************/
00068
00069 #define FS_MAXVOLUME 5 /* maximum number of volumes */
00070 #define FS_MAXTASK 10  /* maximum number of task */
```

```
00071
00072 #define FS_MAXPATH 256 /* maximum lenght for path */
00073
00074 #ifndef NB_BARE_METAL
00075 #define FS_MUTEX_TYPE OS_CRIT
00076 #else
00077 typedef struct
00078 {
00079     unsigned short val;
00080     short int taskID;
00081 } _fsm_mutex;
00082 #define FS_MUTEX_TYPE _fsm_mutex
00083 #endif
00084
00085 /*****************************************************************************
00086  *
00087  *  if Unicode is used then comment in HCC_UNICODE define
00088  *
00089  *****************************************************************************/
00090 /* #define HCC_UNICODE */
00091
00092 typedef unsigned short wchar;
00093
00094 #ifdef HCC_UNICODE
00095 #define W_CHAR wchar
00096 #else
00097 #define W_CHAR char
00098 #endif
00099
00100 #if 1
00101 #define FS_SEPARATORCHAR '/'
00102 #else
00103 #define FS_SEPARATORCHAR '\\'
00104 #endif
00105
00106 #endif /* FS_CAPI_USED */
00107
00108 #ifdef __cplusplus
00109 }
00110 #endif
00111
00112 /*****************************************************************************
00113  *
00114  * end of udefs.h
00115  *
00116  *****************************************************************************/
00117
00118 #endif /* _UDEFSSTD_H_ */
```

24.363 ftp.h File Reference

NetBurner FTP Client API.

`#include <nettypes.h>`

Macros

- #define **FTP_OK** (0)

  OK.
- #define **FTP_TIMEOUT** (-1)

  Timeout.
- #define **FTP_PASSWORDERROR** (-2)

  Password error.
- #define **FTP_CONNECTFAIL** (-3)

  Connection failed.
- #define **FTP_COMMANDFAIL** (-4)

  Command failed.
- #define **FTP_COMMANDERROR** (-4)

  Command error.
- #define **FTP_BADSESSION** (-5)

  Bad session.

- #define **FTP_NETWORKERROR** (-6)

  Network error.

Functions

- int FTP_InitializeSession (IPADDR4 server_address, uint16_t port, PCSTR UserName, PCSTR Password, uint32_t time_out)

  Initialize a FTP session with a FTP server.
- int FTP_CloseSession (int session)

  Close a FTP session.
- int FTPGetDir (int ftp_Session, char *dir_buf, int nbytes, uint16_t timeout)

  Get the current working directory.
- int FTPSetDir (int ftp_Session, const char *new_dir, uint16_t timeout)

  Set the current working directory.
- int FTPDeleteDir (int ftp_Session, const char *dir_to_delete, uint16_t timeout)

  Delete a directory.
- int FTPMakeDir (int ftp_Session, const char *dir_to_make, uint16_t timeout)

  Create a new directory.
- int FTPUpDir (int ftp_Session, uint16_t timeout)

  Move up one directory level.
- int FTPDeleteFile (int ftp_Session, const char *file_name, uint16_t timeout)

  Delete a file.
- int FTPRenameFile (int ftp_Session, const char *old_file_name, const char *new_file_name, uint16_t timeout)

  Rename a file.
- int FTPSendFile (int ftp_Session, const char *full_file_name, BOOL bBinaryMode, uint16_t timeout)

  Initialize the process to send a file to s FTP server.
- int FTPGetFile (int ftp_Session, const char *full_file_name, BOOL bBinaryMode, uint16_t timeout)

  Initialize the process to get a file from a FTP server.
- int FTPGetList (int ftp_Session, const char *full_dir_name, uint16_t timeout)

  Initialize the process to receive a directory listing from a FTP server.
- int FTPGetFileNames (int ftp_Session, const char *full_dir_name, uint16_t timeout)

  Initialize the process to receive just the file names in a directory from a FTP server.
- int FTPRawCommand (int ftp_Session, const char *cmd, char *cmd_buf, int nbytes, uint16_t timeout)

  Send a FTP command to the FTP server.
- int FTPGetCommandResult (int ftp_Session, char *cmd_buf, int nbytes, uint16_t timeout)

  Returns the result of the last FTP operation.
- int FTPRawStreamCommand (int ftp_Session, const char *cmd, int *pResult, char *cmd_buf, int nbytes, uint16_t timeout)

  Send a command and receive a response over a stream connection.
- void FTPActiveMode (int ftp_Session)

  Set mode to active.
- void FTPPassiveMode (int ftp_Session)

  Set mode to passive.

24.363.1 Detailed Description

NetBurner FTP Client API.

24.364 ftp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef _NB_FTP_H
00016 #define _NB_FTP_H
00017
00018 #include <nettypes.h>
00019
00024 #define FTP_OK (0)
00025 #define FTP_TIMEOUT (-1)
00026 #define FTP_PASSWORDERROR (-2)
00027 #define FTP_CONNECTFAIL (-3)
00028 #define FTP_COMMANDFAIL (-4)
00029 #define FTP_COMMANDERROR (-4)
00030 #define FTP_BADSESSION (-5)
00031 #define FTP_NETWORKERROR (-6)
00051 int FTP_InitializeSession(IPADDR4 server_address, uint16_t port, PCSTR UserName, PCSTR Password,
      uint32_t time_out);
00052
00063 int FTP_CloseSession(int session);
00064
00076 int FTPGetDir(int ftp_Session, char *dir_buf, int nbytes, uint16_t timeout);
00077
00088 int FTPSetDir(int ftp_Session, const char *new_dir, uint16_t timeout);
00089
00100 int FTPDeleteDir(int ftp_Session, const char *dir_to_delete, uint16_t timeout);
00101
00112 int FTPMakeDir(int ftp_Session, const char *dir_to_make, uint16_t timeout);
00113
00123 int FTPUpDir(int ftp_Session, uint16_t timeout);
00124
00135 int FTPDeleteFile(int ftp_Session, const char *file_name, uint16_t timeout);
00136
00148 int FTPRenameFile(int ftp_Session, const char *old_file_name, const char *new_file_name, uint16_t
      timeout);
00149
00190 int FTPSendFile(int ftp_Session, const char *full_file_name, BOOL bBinaryMode, uint16_t timeout);
00191
00230 int FTPGetFile(int ftp_Session, const char *full_file_name, BOOL bBinaryMode, uint16_t timeout);
00231
00250 int FTPGetList(int ftp_Session, const char *full_dir_name, uint16_t timeout);
00251
00270 int FTPGetFileNames(int ftp_Session, const char *full_dir_name, uint16_t timeout);
00271
00284 int FTPRawCommand(int ftp_Session, const char *cmd, char *cmd_buf, int nbytes, uint16_t timeout);
00285
00305 int FTPGetCommandResult(int ftp_Session, char *cmd_buf, int nbytes, uint16_t timeout);
00306
00325 int FTPRawStreamCommand(int ftp_Session, const char *cmd, int *pResult, char *cmd_buf, int nbytes,
      uint16_t timeout);
00326
00333 void FTPActiveMode(int ftp_Session);
00334
00341 void FTPPassiveMode(int ftp_Session);
00342
00343 #endif
00344
```

24.365 ftpd.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_FTPD_H
00006 #define _NB_FTPD_H
00007
00008 #include <nettypes.h>
00009
00010 /* Jan 16, 2008. These #defines were removed and all server functions (denoted as FTPD)
00011  * were modified to use FTPD_OK and FTPD_FAIL. This was necessary so that the FTP
00012  * client and server header files could be used at the same time. If you have any
00013  * errors in your application code, you can salefly change all server functions from
00014  * FTP_OK/FAIL to FTPD_OK/FAIL.
00015  */
00016 //#define FTP_FAIL        (0)
00017 //#define FTP_OK          (1)
00018
00019 #define FTPD_FAIL (0)
00020 #define FTPD_OK (1)
```

```
00021 #define FTPD_RUNNING (2)
00022 #define FTPD_NOT_RUNNING (3)
00023 #define FTPD_LISTEN_ERR (4)
00024
00025 /* FTP Server Code */
00026 /*Module: FTP Server
00027 Implementing an FTP server in an embedded system without a built-in
00028 file system is not a trivial undertaking. Most embedded applications
00029 do not require a file system, and a file system is not part of the
00030 standard NetBurner package. If a file system is required for a specific
00031 application, it is the responsibility of the programmer to implement
00032 the required features. A file system could be the trivially simple
00033 example of a single file, or it could be quite complex.
00034 Using the FTP Server requires that you, the programmer, write the
00035 functions defined in the FTP documentation. These functions are
00036 "callback" functions that allow you to customize the FTP server
00037 actions to suit your particular application. All callback functions
00038 required for your application must be implemented by you.
00039 The NetBurner examples directory has two FTP server sample applications:
00040 1. examples\ftpd_trivial - A simple system that reads and writes a single file.
00041 2. examples\ftpd_expose_hml - A more complex example that exposes all of the
00042    HTML served files to the  FTP server. This example also shows how to upgrade
00043    the NetBurner firmware and reset the system using FTP.
00044 */
00045 /*Functions:*/
00046 /*Group:FTP Server Initialization Functions */
00047 /*Initialize the FTP Server */
00048 /*
00049 This call starts the FTP Server task, which listens for incoming connections. Only one instance of the
      FTPD is allowed
00050 Parameters:
00051    uint16_t port The TCP port to listen to for incoming FTP requests
00052    uint8_t server_priority The uC/OS task priority for the FTP Server
00053 Returns:
00054    FTPD_RUNNING - The FTPD is already running
00055    FTPD_LISTEN_ERR - The listen socket was not able to be opened
00056    FTPD_OK - The FTPD was succesfully started
00057    FTPD_FAIL - The FTPD task could not be created
00058 See Also:
00059    void FTPDStopReq();
00060 */
00061 #ifdef FTPD_SSL_SUPPORT
00062 int FTPDStartSSL(uint16_t port, uint8_t server_priority, bool enableFTPS = false, bool encryptData =
      false);
00063 #endif
00064
00065 int FTPDStart(uint16_t port, uint8_t server_priority);
00066
00067 /*
00068 Group: FTP Server stop function
00069 This function sends a stop request to the currently running FTPD.
00070 Parameters:
00071   none
00072 Returns:
00073 FTPD_RUNNING - The FTPD is still running
00074 FTPD_NOT_RUNNING - The FTPD is no longer running
00075 See Also:
00076   int FTPDStart( uint16_t port, uint8_t server_priority );
00077 */
00078 int FTPDStopReq();
00079
00080 /*Group: FTP Session callback typedef */
00081 /*typedef for all directory reporting callbacks */
00082 /*
00083 This callback type definition is used by the directory reporting functions.
00084 Parameters:
00085    int handle The handle passed into the listing function
00086    const char * name_to_report The file name to report for use in a directory listing
00087 Returns:
00088    FTPD_RUNNING - The FTPD is already running
00089    FTPD_LISTEN_ERR - The listen socket was not able to be opened
00090    FTPD_OK - The FTPD was succesfully started
00091    FTPD_FAIL - The FTPD task could not be created
00092
00093 See Also:
00094    int FTPD_ListSubDirectories(const char * current_directory, void * pSession,
      FTPDCallBackReportFunct * pFunc, int handle);
00095    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00096    */
00097
00098 typedef void(FTPDCallBackReportFunct)(int handle, const char *name_to_report);
00099
00100 /*Group: FTP Session callback functions that must be implemented by the programmer */
00101 /*Functions:*/
00102 /* Function called to indicate the start of a User Session. */
00103 /*
```

```
00104 This function is called following the creation of a new FTP session. The function needs to determine
00105 the validity of the user/password pair. The returned void pointer will be passed to all access
      functions,
00106 which will be asked to determine the validity of the operation based on the permissions
00107 associated with the return value.
00108 Parameters:
00109    const char * user The name of the user attempting to establish an FTP session
00110    const char * passwd The password of the user attempting to establish an FTP session
00111    const IPADDR4 hi_ip The IP address of the server trying to establish this connection
00112 Returns:
00113    NULL The user name/password set is invalid
00114    (obj) A non-null void pointer to an object that will be associated with this login session
00115 See Also:
00116    void   FTPDSessionEnd(void * pSession);
00117 */
00118 void *FTPDSessionStart(const char *user, const char *passwd, const IPADDR4 hi_ip);
00119
00120 /*Called to indicate a user session will be terminated. */
00121 /*
00122 This callback function gives the user program the opportunity to clean up any storage
00123 associated with the void pointer returned from the FTPBSessionStart( ) call.
00124 Parameters:
00125    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00126 Returns:
00127     Nothing This is a void function
00128 See Also:
00129    void * FTPBSessionStart(const char * user, const char * passwd, const IPADDR4 hi_ip);
00130 */
00131 void FTPDSessionEnd(void *pSession);
00132
00133 /*Group: FTP Directory callback functions that must be implemented by the programmer */
00134 /*Functions:*/
00135 /* Function called by the FTP Server to test for the existence of a directory. */
00136 /*
00137 Called by the FTP Server as the result of an attempt to change to a new directory. This function can
      also
00138 be used to validate the permissions of the session.  This function must be implemented by
00139 the programmer.
00140 Parameters:
00141    const char * full_directory Name of the new directory to test
00142    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00143 Returns:
00144    FTPD_OK The requested directory exists
00145    FTPD_FAIL The requested directory does not exist, or access is not permitted for the user
00146 See Also:
00147    int FTPD_CreateSubDirectory(const char * current_directory, const char * new_dir, void * pSession);
00148    int FTPD_DeleteSubDirectory(const char * current_directory, const char * sub_dir, void * pSession);
00149 */
00150 int FTPD_DirectoryExists(const char *full_directory, void *pSession);
00151
00152 /* Function called by the FTP Server to create a directory. */
00153 /*
00154 Called by the FTP Server as the result of an attempt to create a new directory. This function can
00155 also be used to validate the permissions of the session.  This function must be implemented
00156 by the programmer.
00157 Parameters:
00158       const char * current_directory The current value of the session directory
00159    const char * new_dir The directory to create under the current_directory
00160    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00161 Returns:
00162    FTPD_OK The requested directory was created
00163    FTPD_FAIL The requested directory could not be created
00164 See Also:
00165    int FTPD_DirectoryExists(const char * full_directory, void * pSession);
00166    int FTPD_DeleteSubDirectory(const char * current_directory, const char * sub_dir, void * pSession);
00167 */
00168 int FTPD_CreateSubDirectory(const char *current_directory, const char *new_dir, void *pSession);
00169
00170 /* Function called by the FTP Server to delete a directory. */
00171 /*
00172    Called by the FTP Server as the result of an attempt to delete a subdirectory. This function call
      can be used
00173    to validate the permissions of this session.  This function must be implemented by the programmer.
00174 Parameters:
00175       const char * current_directory The current value of the session current directory
00176    const char * sub_dir The directory to delete under the current_directory
00177    void * pSession The void * object returned from the FTPBSessionStart( ) function call at the
      beginning of the session
00178 Returns:
00179    FTPD_OK The requested directory was deleted
00180    FTPD_FAIL The requested directory could not be deleted
00181 See Also:
00182    int FTPD_DirectoryExists(const char * full_directory, void * pSession);
00183    int FTPD_CreateSubDirectory(const char * current_directory, const char * new_dir, void * pSession);
00184 */
00185 int FTPD_DeleteSubDirectory(const char *current_directory, const char *sub_dir, void *pSession);
00186
```

```
00187 /* Function called by the FTP Server to list all subdirectories under the current directory. */
00188 /*
00189    Called by the FTP Server as the result of a client's attempt to list the contents of a directory.
00190    This function must be implemented by the programmer.
00191 Parameters:
00192       const char * current_directory The current value of the session current directory
00193    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00194    FTPDCallBackReportFunct * pFunc The pointer to the callback function to be called for each
      subdirectory
00195    int handle The handle value to be passed back into the pFunc
00196 Returns:
00197    FTPD_OK The requested listing was successfully delivered
00198    FTPD_FAIL The requested directory could not be listed
00199 See Also:
00200    int FTPD_DirectoryExists(const char * full_directory, void * pSession);
00201    int FTPD_DeleteSubDirectory(const char * current_directory, const char * sub_dir, void * pSession);
00202    int FTPD_CreateSubDirectory(const char * current_directory, const char * new_dir, void * pSession);
00203 Example:
00204 Everything inside the callback function stub must be supplied by the programmer. The FTP server will
00205 automatically call this function and provide values for the function variables. It is the programmer's
00206 responsibility to execute pFunc( ) with the provided handle and a pointer to the string representing
00207 the subdirectory name. pFunc( ) must be executed once for each subdirectory name.
00208 In this example, the variables number_of_directories and DirectoryName[] must be declared and
00209 initialized elsewhere in the application program.
00210 int FTPD_ListSubDirectories(const char * current_directory, void * pSession, FTPDCallBackReportFunct *
      pFunc, int handle);
00211 {
00212    for (int n = 0; n < number_of_dir; n++)
00213       pFunc(handle, DirectoryName[n]);
00214    return FTPD_OK;
00215 }
00216 */
00217 int FTPD_ListSubDirectories(const char *current_directory, void *pSession, FTPDCallBackReportFunct
      *pFunc, int handle);
00218
00219 /*Group: FTP file callback functions that must be implemented by the programmer */
00220 /*Functions:*/
00221 /*Function to report on the whether or not a file exists */
00222 /*
00223 Check for the existence of a file, usually just before an attempt is made to download the file.
00224 This function must be implemented by the programmer.
00225 Parameters:
00226       const char * full_directory The current value of the session directory
00227    const char * file_name The name of the file to check
00228    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00229
00230 Returns:
00231    FTPD_OK The requested file exists
00232    FTPD_FAIL The requested file does not exist
00233 See Also:
00234    int FTPD_FileExists(const char * full_directory, const char * file_name, void * pSession);
00235    int FTPD_SendFileToClient(const char * full_directory, const char * file_name, void * pSession, int
      fd);
00236    int FTPD_AbleToCreateFile(const char * full_directory, const char * file_name, void * pSession);
00237    int FTPD_GetFileFromClient(const char * full_directory, const char * file_name, void * pSession,
      int fd);
00238    int FTPD_DeleteFile(const char * current_directory, const char * file_name, void * pSession);
00239    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00240 */
00241 int FTPD_FileExists(const char *full_directory, const char *file_name, void *pSession);
00242
00243 /* returns the size of a specific file.
00244 return -1 no such file.
00245 return 0 file size unknown
00246 return file size
00247 */
00248 #define FTPD_FILE_SIZE_NOSUCH_FILE (-1)
00249 #define FTPD_FILE_SIZE_UNKNOWN (0)
00250 int FTPD_GetFileSize(const char *full_directory, const char *file_name);
00251
00252 /*Function to send the contents of a file to a file descriptor */
00253 /*
00254 Send a file to a FTP client. This function must be implemented by the programmer.
00255 Parameters:
00256       const char * full_directory. The current value of the session directory
00257    const char * file_name The name of the file to send
00258    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00259    int fd  The file descriptor to send to
00260 Returns:
00261    FTPD_OK The requested file was sent.
00262    FTPD_FAIL The requested file was not sent.
00263 See Also:
00264    int FTPD_FileExists(const char * full_directory, const char * file_name, void * pSession);
00265    int FTPD_AbleToCreateFile(const char * full_directory, const char * file_name, void * pSession);
00266    int FTPD_GetFileFromClient(const char * full_directory, const char * file_name, void * pSession,
      int fd);
```

```
00267    int FTPD_DeleteFile(const char * current_directory, const char * file_name, void * pSession);
00268    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00269 */
00270 int FTPD_SendFileToClient(const char *full_directory, const char *file_name, void *pSession, int fd);
00271
00272 /*Function to report on the ability to create/receive a file. */
00273 /*
00274 Determine if a file can be created.  This function must be implemented by the programmer.
00275 Parameters:
00276       const char * full_directory The current value of the session directory
00277    const char * file_name The name of the file to create
00278    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00279 Returns:
00280    FTPD_OK The requested file can be written/created
00281    FTPD_FAIL The requested file could not be created
00282 See Also:
00283    int FTPD_FileExists(const char * full_directory, const char * file_name, void * pSession);
00284    int FTPD_SendFileToClient(const char * full_directory, const char * file_name, void * pSession, int
      fd);
00285    int FTPD_GetFileFromClient(const char * full_directory, const char * file_name, void * pSession,
      int fd);
00286    int FTPD_DeleteFile(const char * current_directory, const char * file_name, void * pSession);
00287    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00288 */
00289 int FTPD_AbleToCreateFile(const char *full_directory, const char *file_name, void *pSession);
00290
00291 /*Function to create/get a file */
00292 /*
00293 Receive a file from the FTP client.  This function must be implemented by the programmer.
00294 Parameters:
00295       const char * full_directory The current value of the session directory
00296    const char * file_name The name of the file to create
00297    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00298    int fd The file descriptor that will be used to receive the file
00299 Returns:
00300    FTPD_OK The requested file was written/created
00301    FTPD_FAIL The requested file was not created
00302 See Also:
00303    int FTPD_FileExists(const char * full_directory, const char * file_name, void * pSession);
00304    int FTPD_SendFileToClient(const char * full_directory, const char * file_name, void * pSession, int
      fd);
00305    int FTPD_AbleToCreateFile(const char * full_directory, const char * file_name, void * pSession);
00306    int FTPD_DeleteFile(const char * current_directory, const char * file_name, void * pSession);
00307    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00308 */
00309 int FTPD_GetFileFromClient(const char *full_directory, const char *file_name, void *pSession, int fd);
00310
00311 /*User written function to delete a file */
00312 /*
00313 Delete a file.  This function must be implemented by the programmer.
00314 Parameters:
00315       const char * full_directory The current value of the session directory
00316    const char * file_name The name of the file to delete
00317    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00318 Returns:
00319    FTPD_OK The requested file was deleted
00320    FTPD_FAIL The requested file could not be deleted
00321 See Also:
00322    int FTPD_FileExists(const char * full_directory, const char * file_name, void * pSession);
00323    int FTPD_SendFileToClient(const char * full_directory, const char * file_name, void * pSession, int
      fd);
00324    int FTPD_AbleToCreateFile(const char * full_directory, const char * file_name, void * pSession);
00325    int FTPD_GetFileFromClient(const char * full_directory, const char * file_name, void * pSession,
      int fd);
00326    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00327 */
00328 int FTPD_DeleteFile(const char *current_directory, const char *file_name, void *pSession);
00329
00330 /*User written function to rename a file */
00331 /*
00332 Rename a file.  This function must be implemented by the programmer.
00333 Parameters:
00334    const char * full_directory The current value of the session directory
00335    const char * cur_file_name The name of the file to rename
00336    const char * new_file_name
00337    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00338 Returns:
00339    FTPD_OK The requested file was deleted
00340    FTPD_FAIL The requested file could not be renamed
00341 See Also:
00342    int FTPD_FileExists(const char * full_directory, const char * file_name, void * pSession);
00343    int FTPD_SendFileToClient(const char * full_directory, const char * file_name, void * pSession, int
      fd);
```

```
00344    int FTPD_AbleToCreateFile(const char * full_directory, const char * file_name, void * pSession);
00345    int FTPD_GetFileFromClient(const char * full_directory, const char * file_name, void * pSession,
      int fd);
00346    int FTPD_ListFile(const char * current_directory, void * pSession, FTPDCallBackReportFunct * pFunc,
      int handle);
00347 */
00348 int FTPD_Rename(const char *current_directory, const char *cur_file_name, const char *new_file_name,
      void *pSession);
00349
00350 /*User supplied function that lists all files in the current directory */
00351 /*
00352 List all files in the current directory.  This function must be implemented by the programmer.
00353 Parameters:
00354       const char * current_directory The current value of the session directory
00355    void * pSession The void * object returned from the FTPBSessionStart( ) function call
00356    FTPDCallBackReportFunct * pFunc The pointer to the callback function to be called for each file
      name. This is a callback function
00357 provided and used by the NetBurner internal FTP code. int handle The handle value to be passed back
      into the pFunc. This is a handle
00358 provided and used by the NetBurner internal FTP code. Returns: FTPD_OK The requested files were listed
      FTPD_FAIL The requested file was not
00359 listed See Also: int FTPD_FileExists(const char * full_directory, const char * file_name, void *
      pSession); int FTPD_SendFileToClient(const
00360 char * full_directory, const char * file_name, void * pSession, int fd); int
      FTPD_AbleToCreateFile(const char * full_directory, const char *
00361 file_name, void * pSession); int FTPD_GetFileFromClient(const char * full_directory, const char *
      file_name, void * pSession, int fd); int
00362 FTPD_DeleteFile(const char * current_directory, const char * file_name, void * pSession); Example:
      Everything inside the callback function
00363 stub must be supplied by the programmer. The FTP server will automatically call this function and
      provide values for the function variables.
00364 It is the programmer's responsibility to execute pFunc( ) with the provided handle and a pointer to
      the string representing the file
00365 name. pFunc( ) must be executed once for each file name. In this example, the variables
      number_of_directories and FileNames[] must be
00366 declared and initialized elsewhere in the application program. int FTPD_ListFile(const char *
      current_directory, void * pSession,
00367 FTPDCallBackReportFunct * pFunc, int handle);
00368 {
00369    for (int n = 0; n < numberof_files; n++)
00370       pFunc(handle, FileNames[n]);
00371    return FTPD_OK;
00372 }
00373 */
00374 int FTPD_ListFile(const char *current_directory, void *pSession, FTPDCallBackReportFunct *pFunc, int
      handle);
00375
00376 #endif
```

24.366 gdbstub.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _GDB_MON_H
00006 #define _GDB_MON_H
00007
00008 /* GDB Stub functions */
00009 /*Module: GDB Stub
00010 This Module provides code for operating a serial GDB stub.
00011 This is used to debug programs running on a NetBurner system.
00012 More information can be fount in ..nburn/docs/gdb/GDB-Gettingstarted
00013 */
00014
00015 /*Functions:*/
00016 /*Group:Stub Initialization functions*/
00017 /* Start the GDB stub and then immediately stop at a break point */
00018 /*
00019 This function will start the GDB stub, and will stop immediately at
00020 a breakpoint. This function is best used during development to
00021 stop the system from executing questionable code at startup.
00022 It is also useful to control the system execution to debug startup
00023 and initialization code.
00024
00025 Parameters:
00026    int port The serial port to use this can be 0 or 1 coresponding to uart 0 or uart 1
00027    int baudrate The baud rate to use on the port.
00028
00029
00030 */
00031 /*See Also:
00032    void InitGDBStubNoBreak(int port, int baudrate);
00033 */
00034 void InitGDBStub(int port, int baudrate);
```

```
00035
00036 /* Start the GDB stub and continue without stopping */
00037 /*
00038 This function will start the GDB stub, and continue with normal
00039 program execution. This function is best used to debug a program that
00040 is operating correctly and has intermitant bugs or anomallies.
00041 The program will run as normal until you connect the debugger.
00042 Thus you can operate the program until one needs to debug it.
00043
00044 Parameters:
00045    int port The serial port to use this can be 0 or 1 coresponding to uart 0 or uart 1
00046    int baudrate The baud rate to use on the port.
00047
00048
00049 */
00050 /*See Also:
00051    void InitGDBStub(int port, int baudrate);
00052 */
00053 void InitGDBStubNoBreak(int port, int baudrate);
00054
00055 #endif
```

24.367 hal.h File Reference

NetBurner Hardware Abstraction Layer (HAL)

```
#include <stdint.h>
#include <crypto/ssl.h>
#include <crypto/wolfssl/ssl.h>
```

Functions

- void HardwareSetup ()

  Initializes the system hardware such as the timer, cache and clock speed.
- void PostConfigHardwareInit ()

  Initializes the system hardware that depend on config variables such as the watchdog.
- void ForceReboot (bool bFromException=false)

  Forces the system hardware to perform a soft reset.
- int HalStorage_Save (uint8_t area, void *pData, int len, int offset=0)

  Save a blob to a specific persistent storage area as defined by the platform. This routine will perform all storage maintenance routines necessary to ensure a valid write.
- int HalStorage_SavePartial (uint8_t area, void *pData, int len, int offset)

  Save a blob to a specific persistent storage area as defined by the platform. This routine requires that any maintenance required for a valid write be explicitly performed prior to being called.
- int HalStorage_Prepare (uint8_t area, int len, int offset=0)

  Prepare a storage area for writing new data. For platforms with direct flash mapping this function is an alias for HalStorage_Erase.
- int HalStorage_Finalize (uint8_t area)

  Finalize a storage area after writing new data. This will perform any final completion or cleanup routines required to persist the previously saved data from HalStorage_SavePartial calls.
- int HalStorage_Erase (uint8_t area, int len=-1, int offset=0)

  Erase all or part of a storage area. Note: due to physical storage granularities, the total area erased may extend beyond the requested area.
- int HalStorage_GetAllocated (uint8_t area)

  Obtain the total size allocated to the given persistent storage area.
- int HalStorage_GetMaxAllocation (uint8_t area)

  Obtain the maximum size that may be allocated to the given persistent storage area. For direct flash mapped platforms, this is aliased to HalStorage_GetAllocated.
- int HalStorage_RemainingSpace (uint8_t area)

  Obtain the number of remaining bytes available to be written in the given persistent storage area.
- int HalStorage_WriteOffset (uint8_t area)

Get the offset of the next byte that can be written in the given persistent storage area.

- int HalStorage_AddressOffset (uint8_t area, void *pWhere)

  Get the offset within a persistent storage area of an address. A platform may map any address to any offset of its choosing for a given storage area, and my modify its behavior on an area by area basis.
- void FlashErase (void *pWhere, int len)

  Erases the flash memory.
- void FlashProgram (void *pWhere, void *pWhat, int len)

  Program flash memory.
- void FlashProgramAppImage (void *pWhere, void *pWhat, int len)

  Write an application image to flash memory.
- void DisableCache ()

  Disable the instruction and data cache.
- void EnableCache ()

  Enable the instruction and data cache.
- uint32_t spaceleft ()

  Report how much free unallocated space is left in dynamic memory.
- uint16_t HalGetTickFraction (void)

  Returns the fraction of the current system time tick.
- void StdioCheckIntc (void)

  Check STDIO interrupt sources.
- void SysLogCheckIntc (void)

  This is just like the StdioCheckIntc() function, except that the results are displayed via UDP.
- bool HalDeviceCertValid ()

  Determine if the stored certificate is valid.
- uint8_t * HalGetDeviceCert ()

  Get a pointer to the stored certificate.
- uint16_t HalGetDeviceCertLen ()

  Get the length of the stored certificate.
- uint8_t * HalGetDeviceKey ()

  Get a pointer to the stored certificate.
- uint16_t HalGetDeviceKeyLen ()

  Get the length of the stored key.
- bool HalSaveNewDeviceCert (const uint8_t *cert, uint16_t certlen, uint8_t format=SSL_FILETYPE_PEM)

  Save a device certificate in persistent storage.
- bool HalSaveNewDeviceKey (const uint8_t *key, uint16_t keylen, uint8_t format=SSL_FILETYPE_PEM)

  Save a device key in persistent storage.
- void HalEraseDeviceCertAndKey ()

  Clear the device certificate and key from persistent storage.

Variables

- void(* watchdog_service_function)(void)

  Watchdog callback service function.
- uint32_t HalTickMaxCount

  Rollover value for the system hardware tick timer.

24.367.1 Detailed Description

NetBurner Hardware Abstraction Layer (HAL)

24.368 hal.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00022 #ifndef _NB_HAL_H_
00023 #define _NB_HAL_H_
00024
00025 #include <stdint.h>
00026
00027 #if defined(NB_SSL_SUPPORTED) || defined(NB_SSH_SUPPORTED)
00028 // SSL For Hashing Certs
00029 #include <crypto/ssl.h>
00030 #include <crypto/wolfssl/ssl.h>
00031 #endif /* defined(NB_SSL_SUPPORTED) || defined(NB_SSH_SUPPORTED) */
00032
00033 enum HalStorage_t
00034 {
00035     HalStore_Config = 0x01,
00036     HalStore_Cert = 0x02,
00037     HalStore_UserParams = 0x03,
00038     HalStore_App = 0x04,
00039     HalStore_FileSys = 0xFC
00040 };
00041
00042 enum HalError_t : int
00043 {
00044     HALSTORE_NO_ERROR = 0,
00045     HALSTORE_ERR_UNKNOWN = -1,
00046     HALSTORE_BADARG = -2,
00047     HALSTORE_TOOBIG = -3,
00048     HALSTORE_NOTSUPPORTED = -4,
00049     HALSTORE_NOTAVAILABLE = -5,
00050     HALSTORE_NOTREADY = -6,
00051 };
00052
00057 void HardwareSetup();
00058
00063 void PostConfigHardwareInit();
00064
00069 void ForceReboot(bool bFromException = false) __attribute__((noreturn));
00070
00085 int HalStorage_Save(uint8_t area, void *pData, int len, int offset = 0);
00086
00101 int HalStorage_SavePartial(uint8_t area, void *pData, int len, int offset);
00102
00113 int HalStorage_Prepare(uint8_t area, int len, int offset = 0);
00114
00124 int HalStorage_Finalize(uint8_t area);
00125
00139 int HalStorage_Erase(uint8_t area, int len = -1, int offset = 0);
00140
00148 int HalStorage_GetAllocated(uint8_t area);
00149
00159 int HalStorage_GetMaxAllocation(uint8_t area);
00160
00169 int HalStorage_RemainingSpace(uint8_t area);
00170
00179 int HalStorage_WriteOffset(uint8_t area);
00180
00192 int HalStorage_AddressOffset(uint8_t area, void *pWhere);
00193
00201 void FlashErase(void *pWhere, int len);
00202
00211 void FlashProgram(void *pWhere, void *pWhat, int len);
00212
00225 void FlashProgramAppImage(void *pWhere, void *pWhat, int len);
00226
00231 void DisableCache();
00232
00237 void EnableCache();
00238
00244 uint32_t spaceleft();
00245
00257 extern void (*watchdog_service_function)(void);
00258
00265 extern uint16_t HalGetTickFraction(void);
00266
00267 #ifdef GATHER_RANDOM_USE_HW
00274 extern bool HalHWRandRdy(void);
00275
00281 extern uint32_t HalGetHWRand32(void);
00282 #endif
00283
```

```
00289 extern uint32_t HalTickMaxCount;
00290
00301 void StdioCheckIntc(void);
00302
00308 void SysLogCheckIntc(void);
00309
00310 #if defined(NB_SSL_SUPPORTED) || defined(NB_SSH_SUPPORTED)
00311
00323 bool HalDeviceCertValid();
00324
00325 // TODO: Should there be a HalDeviceKeyValid() ?
00326
00334 uint8_t *HalGetDeviceCert();
00335
00344 uint16_t HalGetDeviceCertLen();
00345
00353 uint8_t *HalGetDeviceKey();
00354
00363 uint16_t HalGetDeviceKeyLen();
00364
00365
00366 uint16_t HalGetDeviceFormat();
00367
00378 bool HalSaveNewDeviceCert(const uint8_t *cert, uint16_t certlen, uint8_t format = SSL_FILETYPE_PEM);
00379
00390 bool HalSaveNewDeviceKey(const uint8_t *key, uint16_t keylen, uint8_t format = SSL_FILETYPE_PEM);
00391
00398 void HalEraseDeviceCertAndKey();
00399 #endif /* defined(NB_SSL_SUPPORTED) || defined(NB_SSH_SUPPORTED) */
00400
00401 #endif /* _NB_HAL_H_ */
00402
```

24.369 hash.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __HASH_H
00006 #define __HASH_H
00007
00008 #include <stdio.h>
00009
00013 #define MD5_HASH_LENGTH (16)
00014 #define SHA1_HASH_LENGTH (20)
00015 #define SHA2_HASH_LENGTH (32)
00016 #define TLS1_HASH_LENGTH MD5_HASH_LENGTH + SHA1_HASH_LENGTH
00017 #define TLS1_2_HASH_LENGTH SHA1_HASH_LENGTH + SHA2_HASH_LENGTH
00018 #define MAX_HASH_LENGTH SHA2_HASH_LENGTH
00019 #define MAX_COMBINED_HASH_LENGTH TLS1_HASH_LENGTH
00020
00021 struct __vtable_HASH_CTX_t;
00022
00023 struct HASH_CTX
00024 {
00025     __vtable_HASH_CTX_t *__vtable;
00026
00027     void Init();
00028     void Update(const unsigned char *data, unsigned int len);
00029     void Final(unsigned char *digest);
00030     int GetDigestLen() const;
00031     int GetOIDLen() const;
00032     const unsigned char *GetOID() const;
00033 };
00034
00035 typedef void (HASH_CTX::*HashInitFn)();
00036 typedef void (HASH_CTX::*HashUpdateFn)(const unsigned char *, unsigned int);
00037 typedef void (HASH_CTX::*HashFinalFn)(unsigned char *);
00038 typedef int (HASH_CTX::*HashDigestLenFn)() const;
00039 typedef int (HASH_CTX::*HashOIDLenFn)() const;
00040 typedef const unsigned char *(HASH_CTX::*HashOIDFn)() const;
00041
00042 struct __vtable_HASH_CTX_t
00043 {
00044     HashInitFn _Init;
00045     HashUpdateFn _Update;
00046     HashFinalFn _Final;
00047     HashDigestLenFn _GetDigestLen;
00048     HashOIDLenFn _GetOIDLen;
00049     HashOIDFn _GetOID;
00050 };
00051
00052 inline void HASH_CTX::Init()
00053 {
```

```
00054     (this->*(this->__vtable->_Init))();
00055 }
00056 inline void HASH_CTX::Update(const unsigned char *data, unsigned int len)
00057 {
00058     (this->*(this->__vtable->_Update))(data, len);
00059 }
00060
00061 inline void HASH_CTX::Final(unsigned char *digest)
00062 {
00063     (this->*(this->__vtable->_Final))(digest);
00064 }
00065 inline int HASH_CTX::GetDigestLen() const
00066 {
00067     return (this->*(this->__vtable->_GetDigestLen))();
00068 }
00069 inline int HASH_CTX::GetOIDLen() const
00070 {
00071     return (this->*(this->__vtable->_GetOIDLen))();
00072 }
00073 inline const unsigned char *HASH_CTX::GetOID() const
00074 {
00075     return (this->*(this->__vtable->_GetOID))();
00076 }
00077
00078 #endif /* ----- #ifndef __HASH_H  ----- */
```

24.370 HiResDelay.h File Reference

NetBurner High resolution delay Timer.

```
#include <constants.h>
#include <nbrtos.h>
```

Classes

- class DelayObject

  Microsecond Delay Class.

24.370.1 Detailed Description

NetBurner High resolution delay Timer.

24.371 HiResDelay.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00033 #ifndef _NB_DELAY_TIMER_H
00034 #define _NB_DELAY_TIMER_H
00035
00036 #include <constants.h>
00037 #include <nbrtos.h>
00058 class DelayObject
00059 {
00060    private:
00061     OS_SEM WakeObject;        // Semaphore for internal ISR use
00062     int TimerNumber;          // Timer number used by the delay object
00063     void *MiscHardwarePtr;    // Pointer to timer hardwae registers
00064
00065    public:
00072     DelayObject(int Timer = FIRST_UNUSED_TIMER);
00073     ~DelayObject();
00074
00080     void DelayUsec(uint32_t usec);
00081
00088     bool valid() { return TimerNumber >= 0; }
00089
00090     // Internal use only for ISR processing.
00091     inline void _wake() { WakeObject.Post(); }
00092 };
00093
00094 #endif
00095
```

24.372 htmlfiles.h File Reference

NetBurner HTTP Web Server File Handling.

```
#include <basictypes.h>
#include <nettypes.h>
#include <ipv6/ipv6_addr.h>
```

Functions

- int SendEmailResponse (int sock, const char *name, const char *attachment)

  Send an email with HTML formatting.
- int SendFullResponse (char const *name, int fd)

  Send a file with the proper HTTP header, file specified by file name.
- int SendFullResponse (HTML_FILE_RECORD *fr, int fd, const char *pUrl)

  Send a file with the proper HTTP header, file specified by HTML_FILE_RECORD.
- int SendHeaderResponse (char const *name, int fd)

  Send a HTTP header response for the specified file type.
- int SendHeaderResponse (HTML_FILE_RECORD *fr, int fd)

  Send a HTTP header response for the specified HTML_FILE_RECORD type.
- int32_t SendFileFragment (char const *name, int32_t fd, PCSTR url=NULL)

  Send a file fragment without a header.
- HTML_FILE_RECORD * GetRecordFromName (char const *name)

  Returns a pointer to a HTML_FILE_RECORD for the specified file name.
- CONFIG_RENDER_OBJ ConfigRenderFunc (int mode, const char *pobj, int len=20, const char *extra=0)

  Render a configuration object as HTML for the specified configuration object.
- CONFIG_RENDER_OBJ ConfigRenderFunc (int mode, config_leaf &cl, int len=20, const char *extra=0)

  Render a configuration object as HTML for the specified configuration leaf object.
- void WriteHtmlVariable (int fd, CONFIG_RENDER_OBJ co)

  Send a CONFIG_RENDER_OBJ to the client.

24.372.1 Detailed Description

NetBurner HTTP Web Server File Handling.

24.373 htmlfiles.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00022 /*-------------------------------------------------------------------------
00023  *   These functions are satisfied by the code generated by the utility
00024  *   "comphtml.exe".
00025  *-------------------------------------------------------------------------*/
00026 #ifndef NB_HTMLFILES_H
00027 #define NB_HTMLFILES_H
00028
00029 #include <basictypes.h>
00030 #include <nettypes.h>
00031
00032 // File compression types
00033 enum comp_type
00034 {
00035     eNoCompression,   // No compression, no dynamic HTML tags
00036     eHuffman,         // Huffman encoded without dynamic content HTML tags
00037     eHuffmanHtml,     // Huffman encoded with dynamic HTML tags such as CPPCALL and VARIABLE
00038     eUncompHtml,      // No compression, file contains dynamic HTML tags such as CPPCALL and VARIABLE
00039     ePrevCompGZip     // File has been compressed with gzip
00040 };
00041
00042 // File type
```

```
00043 enum file_type
00044 {
00045     eTypeText,
00046     eTypeHtml,
00047     eTypeJpg,
00048     eTypeGIF,
00049     eTypeClass,
00050     eTypePNG,
00051     eTypeJar,
00052     eTypeOther,
00053     eTypeMPEG,
00054     eTypeCSS,
00055     eTypeXML,
00056     eTypeWBMP,
00057     eTypeJS
00058 };
00059
00060 // Each file has a record in htmldata.cpp
00061 struct HTML_FILE_RECORD
00062 {
00063     const char *fname;                  // name of file
00064     const unsigned char *file_pointer;  // pointer to file in array
00065     unsigned long siz;                  // size of file in bytes
00066     comp_type cType;                    // file compression type
00067     short fType;                        // file type
00068     short has_calls;                    // file has function call tags, such as: CPPCALL or VARIABLE
00069     int access_group;                   // security access group
00070     bool no_direct;                     // files to be included as part of an outer file, and are not
      directly accessed.
00071                                         // For example, A page with multiple repeated subelements
      could include the
00072                                         // same no_direct page multiple times, but with slightly
      different settings,
00073                                         // thereby eliminating the need to have the whole page as a
      single entry.
00074 };
00075
00076 // Variables declared in auto-generated htmldata.cpp
00077 extern const unsigned n_file_record;
00078 extern const unsigned short huffman_table[];
00079 extern const HTML_FILE_RECORD file_record[];
00080 extern const char *MIME_table[];
00081 extern const char *html_table[];
00082
00094 int SendEmailResponse(int sock, const char *name, const char *attachment);
00095
00113 int SendFullResponse(char const *name, int fd);
00114
00135 int SendFullResponse(HTML_FILE_RECORD *fr, int fd, const char *pUrl);
00136
00152 int SendHeaderResponse(char const *name, int fd);
00153
00169 int SendHeaderResponse(HTML_FILE_RECORD *fr, int fd);
00170
00190 int32_t SendFileFragment(char const *name, int32_t fd, PCSTR url = NULL);
00191
00201 HTML_FILE_RECORD *GetRecordFromName(char const *name);
00202
00203 void SendData(HTML_FILE_RECORD *fr, int sock, PCSTR url = NULL);
00204
00205 void WriteHtmlVariable(int fd, char c);
00206 void WriteHtmlVariable(int fd, int i);
00207 void WriteHtmlVariable(int fd, short i);
00208 void WriteHtmlVariable(int fd, long i);
00209 void WriteHtmlVariable(int fd, uint8_t b);
00210 void WriteHtmlVariable(int fd, uint16_t w);
00211 void WriteHtmlVariable(int fd, unsigned long dw);
00212 void WriteHtmlVariable(int fd, const char *);
00213
00214 void WriteHtmlVariable(int fd, MACADR ip);
00215
00216 #ifdef IPV6
00217 #include <ipv6/ipv6_addr.h>
00218
00219 class IPADDR_WCLASS
00220 {
00221    public:
00222     IPADDR the_addr;
00223     IPADDR_WCLASS(IPADDR ip) { the_addr = ip; }
00224     IPADDR_WCLASS(IPADDR4 ip4) { the_addr = ip4; }
00225 };
00226
00227 IPADDR_WCLASS IPCAST(IPADDR ip);
00228
00229 #else
00230 class IPADDR_WCLASS
00231 {
```

```
00232    public:
00233     IPADDR the_addr;
00234     IPADDR_WCLASS(IPADDR ip) { the_addr = ip; }
00235 };
00236
00237 #endif
00238 IPADDR_WCLASS IPCAST(IPADDR ip);
00239
00240 void WriteHtmlVariable(int fd, IPADDR_WCLASS ipa);
00241
00242 class config_leaf;
00243
00244 /*
00245  * eMode values:
00246  *  1 = CONFIGVALUE     Value of the configuration object. For example an IP value would be a string
      "192.168.1.10".
00247  *  2 = CONFIGINPUT     HTML input element with name and value that can be used as an input for form
      processing. An input
00248  *                      element can be a checkbox (boolean), text edit field, or chooser (drop down
      box).
00249  *  3 = CONFIGFULL      HTML input name, element, and hint all as one HTML string.
00250  *  4 = CONFIGTABLE     HTML input name, element, and hint all as one HTML string with <td> and </td>
      HTML tags. Note: <table>
00251  *                      and </table> tags must be provided separately.
00252  *  5 = CONFIGHINT      Text value of hint associated with element.
00253  *
00254  */
00255 struct CONFIG_RENDER_OBJ
00256 {
00257     config_leaf *pObj;
00258     int eMode;
00259     int len;
00260     const char *extra;
00261 };
00262
00275 CONFIG_RENDER_OBJ ConfigRenderFunc(int mode, const char *pobj, int len = 20, const char *extra = 0);
00276
00288 CONFIG_RENDER_OBJ ConfigRenderFunc(int mode, config_leaf &cl, int len = 20, const char *extra = 0);
00289
00298 void WriteHtmlVariable(int fd, CONFIG_RENDER_OBJ co);
00299
00301 #endif
```

24.374 http.h File Reference

NetBurner HTTP Web Server Header File.

```
#include <nettypes.h>
#include <stddef.h>
#include <config_obj.h>
```

Classes

- struct HTTP_Request

  HTTP Request Structure.

- class HtmlPageHandler

  Base class for all GET handlers. To handle GET requests for a specific URL in your application, build a GET handler object for that specifif URL. A NULL name will be a catch all for all GET requests.

- class CallBackFunctionPageHandler

  Implements the HtmlPageHandler class as a function pointer callback for GET requests.

Typedefs

- typedef int() http_gethandlerfunc(int sock, HTTP_Request &pd)

  Implements the HtmlPageHandler class as a function pointer callback for GET requests.

Enumerations

- enum HTTP_RequestTypes { tUnknown , tGet , tPost , tHead }

  HTTP request types for HTTP page handler callback functions.

- enum HTTP_ACCESS

  HTTP page access return values.

Functions

- HTTP_Request * **GetActiveHttpRequest** ()

  Get thc curent active running http request. Only valid fromwithing http get or post handeling. Also valid during page fill in operations.
- HTTP_ACCESS CheckHttpAccess (int sock, int access_level, HTTP_Request &Req)

  All HTTP requests go though this function.
- void StartHttp (uint16_t port, bool RunConfigMirror)

  Start the HTTP web server. Further documentation in the Initialization section Initialization - System Initialization Functions.
- void StopHttp ()

  Stop the HTTP web server.
- void SendHTMLHeader (int sock)

  Send a HTML response header.
- void SendHTMLHeaderWCookie (int sock, char *cookie)

  Send a HTML response header and cookie.
- void SendTextHeader (int sock)

  Send a HTML plain text header.
- void SendGifHeader (int sock)

  Send a HTML GIF header.
- void EmptyResponse (int sock)

  Send an empty response back.
- void NoContentResponse (int sock)

  Send a no content response back.
- void RedirectResponse (int sock, PCSTR new_page)

  Redirect a HTTP request to a different page.
- void NotFoundResponse (int sock, PCSTR new_page)

  Send a page not found response.
- void ForbiddenResponse (int sock, PCSTR new_page)

  Send a page is forbidden response.
- void NotAvailableResponse (int sock, PCSTR new_page)

  Send a response indicating that the requested resource is not available at this time.
- void BadRequestResponse (int sock, PCSTR url, PCSTR data)

  Send a response indicating that the client request itself is faulty in some manner.
- int writeallsafestring (int fd, PCSTR str)

  Send a string and escape all special characters.
- int writesafestring (int fd, PCSTR str, size_t strLength)

  Send a string with a specified length and escape all special characters.
- int httpstricmp (PCSTR str1, PCSTR strIsUpper2)

  Special string compare. Returns 1 if the strings match until one string ends with a null (0).

24.374.1 Detailed Description

NetBurner HTTP Web Server Header File.

24.375 http.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00020 #ifndef _NB_HTTP_H
00021 #define _NB_HTTP_H
00022
00023 #include <nettypes.h>
00024 #include <stddef.h>
00025 #include <config_obj.h>
00026
00027 const int CONFIG_ACCESS_GROUP = 99;
00028
00033 enum HTTP_RequestTypes
00034 {
00035     tUnknown,
00036     tGet,
00037     tPost,
00038     tHead
00039 };
00040
00044 enum HTTP_ACCESS
00045 {
00046     HTTP_OK_TO_SERVE,       //< Ok to serve the page
00047     HTTP_NEED_PASSWORD,     //< Either the password in the HTTP request is wrong or missing
00048     HTTP_NOTFOUND,          //< Report the page is not found
00049     HTTP_FORBIDEN,          //< Report the page forbiden
00050     HTTP_RESPONSE_HANDLED   //< This means this function handled the full response, including closing
      the socket.
00051 };
00052
00053 enum WebsocketFlags
00054 {
00055     WS_UPGRADE = 0x1,
00056     WS_CONNECT = 0x2,
00057     WS_VER_13 = 0x4,
00058     WS_KEY_CONFIRM = 0x8
00059 };
00060
00067 struct HTTP_Request
00068 {
00069     PSTR pURL;
00070     PSTR pAuthorization;
00071     PSTR pFirstCookie;
00072     PSTR pData;
00073     PSTR pSep;
00074     PSTR pHost;
00075     PSTR last_datarx;
00076     PSTR wsKey;
00077     PSTR wsProtocol;
00078     uint16_t sep_len;
00079     uint32_t content_length;
00080     uint32_t rx_length;
00081     IPADDR client_IPaddr;
00082     uint8_t websocketFlags;
00083     HTTP_RequestTypes req;
00084
00085     void ProcessLine(PSTR startofline);
00086     int Respond(int socket);
00087     bool ExtractAuthentication(char **pPassword, char **pUser);
00088 } __attribute__((packed));
00089
00090
00095 HTTP_Request * GetActiveHttpRequest();
00096
00097
00098
00099
00100 struct HTTP_Socket
00101 {
00102     int FileDescriptor;
00103     uint32_t StartTime;
00104     IPADDR client_IPaddr;
00105 } __attribute__((packed));
00106
00107 typedef int(http_wshandler)(HTTP_Request *req, int sock, PSTR url, PSTR rxb);
00108
00109 enum eWEB_ERROR
00110 {
00111     eTCP_CONN_OPEN_NO_DATA,    // WEB client open TCP port#80 conn, but no data request was sent
00112     eTCP_CONN_OPEN_TO_LONG,    // WEB client open TCP connect and send endless data ( for example -
      never ending POST).
00113 };
```

```
00114
00120 class HtmlPageHandler
00121 {
00122    protected:
00123     HtmlPageHandler *m_pNextHandler;
00124     const char *m_pUrlName;
00125     int m_access_group;
00126     HTTP_RequestTypes m_requestTypes;
00127
00128     void InsertSort(HtmlPageHandler *&ph);
00129     int SortValue(HtmlPageHandler *pv);
00130     void Remove();
00131     virtual bool Match(HTTP_Request &req);
00132
00133    public:
00142     HtmlPageHandler(const char *url, HTTP_RequestTypes rt = tGet, int accessGroup = 0, bool
      Before_Files = false);
00143     ~HtmlPageHandler();
00144
00145     static HtmlPageHandler *FindHandler(HTTP_Request &req, bool bBeforeFiles);
00146
00152     virtual int ProcessRaw(int sock, HTTP_Request &pd) = 0;
00153
00159     inline int GetGroup() { return m_access_group; };
00160 };
00161
00171 typedef int(http_gethandlerfunc)(int sock, HTTP_Request &pd);
00172
00173
00174 typedef bool(http_matchhandlerfunc)(HTTP_Request &pd);
00175
00180 class CallBackFunctionPageHandler : public HtmlPageHandler
00181 {
00182    protected:
00183     http_gethandlerfunc *m_pf;
00184     http_matchhandlerfunc *m_mhf;
00185
00186    public:
00187     inline virtual int ProcessRaw(int sock, HTTP_Request &pdt) { return m_pf(sock, pdt); };
00188
00201     inline CallBackFunctionPageHandler(const char *pUrl,
00202                                        http_gethandlerfunc *pFunction,
00203                                        HTTP_RequestTypes reqType = tGet,
00204                                        int accessGroup = 0,
00205                                        bool beforeFiles = false)
00206         : HtmlPageHandler(pUrl, reqType, accessGroup, beforeFiles), m_pf(pFunction)
00207     {
00208         m_mhf = 0;
00209     };
00210
00223     inline CallBackFunctionPageHandler(const char *pUrl,
00224                                        http_gethandlerfunc *pFunction,
00225                                        http_matchhandlerfunc *pMatchFunction,
00226                                        HTTP_RequestTypes reqType = tGet,
00227                                        int accessGroup = 0,
00228                                        bool beforeFiles = false)
00229         : HtmlPageHandler(pUrl, reqType, accessGroup, beforeFiles), m_pf(pFunction)
00230     {
00231         m_mhf = pMatchFunction;
00232     };
00233
00234     virtual bool Match(HTTP_Request &req)
00235     {
00236         if (m_mhf)
00237             return m_mhf(req);
00238         else
00239             return HtmlPageHandler::Match(req);
00240     }
00241 };
00242
00243
00244 class HtmlConfigExposer : public HtmlPageHandler
00245 {
00246 protected:
00247     config_leaf * pLeaf;
00248 public:
00249     inline HtmlConfigExposer(const char *pUrl, config_leaf & leaf,
00250                                        HTTP_RequestTypes reqType = tGet,
00251                                        int accessGroup = 0,
00252                                        bool beforeFiles = false
00253                                        )
00254         : HtmlPageHandler(pUrl, reqType, accessGroup, beforeFiles), pLeaf(&leaf)
00255     {
00256     }
00257
00258 virtual int ProcessRaw(int sock, HTTP_Request &pd);
00259 };
```

```
00260
00261
00262
00263
00264
00265
00266 extern const char *pHttpRealm;   // The realm used for password requests
00267
00280 HTTP_ACCESS CheckHttpAccess(int sock, int access_level, HTTP_Request &Req);
00281
00293 void StartHttp(uint16_t port /* = 80 */, bool RunConfigMirror /* = true */);
00294
00295 typedef int(http_errorhandler)(IPADDR ip, enum eWEB_ERROR Err, void *prm);
00296 typedef int(http_wshandler)(HTTP_Request *req, int sock, PSTR url, PSTR rxb);
00297
00298 /*Setup a custom WEB Error Report Handler */
00299 http_errorhandler *SetNewErrorHandler(http_errorhandler *newhandler);
00300
00301 /*Setup a custom Websocket Request Handler */
00302 http_wshandler *SetNewWSHandler(http_wshandler *newhandler);
00303
00309 void StopHttp();
00310
00320 void SendHTMLHeader(int sock);
00321
00332 void SendHTMLHeaderWCookie(int sock, char *cookie);
00333
00343 void SendTextHeader(int sock);
00344
00355 void SendGifHeader(int sock);
00356
00365 void EmptyResponse(int sock);
00366
00375 void NoContentResponse(int sock);
00376
00386 void RedirectResponse(int sock, PCSTR new_page);
00387
00397 void NotFoundResponse(int sock, PCSTR new_page);
00398
00408 void ForbiddenResponse(int sock, PCSTR new_page);
00409
00420 void NotAvailableResponse(int sock, PCSTR new_page);
00421
00433 void BadRequestResponse(int sock, PCSTR url, PCSTR data);
00434
00435 void writesafestring(int fd, PCSTR str);
00436
00449 int writeallsafestring(int fd, PCSTR str);
00450
00464 int writesafestring(int fd, PCSTR str, size_t strLength);
00465
00466 // Encode a uri component
00467 int decodeURI(char *str);
00468
00491 int httpstricmp(PCSTR str1, PCSTR strIsUpper2);
00492
00493 void append(char *&cpto, const char *cpfrm);
00494
00495 #endif
00496
```

24.376 httppass.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Note: internal use only. Programmers should use the CheckHttpAccess() function,
00006  *       which is a weak reference.
00007  *   @file httppass.h
00008  *   @brief NetBurner HTTP Web Server Password Header File
00009  */
00010
00011 /* @addtogroup httpGroup HTTP and HTML Functions
00012  *   @{
00013  */
00014
00015 #ifndef _NB_HTTPPASS_H
00016 #define _NB_HTTPPASS_H
00017 #include <basictypes.h>
00018 /*
00019  * @brief Reject the current HTTP password request, and send a new password request.
00020  *
00021  * Sends a 401 authentication request to the client.
00022  *
```

```
00023  * @param sock   The socket used to send the request
00024  * @param name   The name that will appear in the password request
00025  *
00026  * @return Nothing
00027  *
00028  * @sa CheckAuthentication()
00029  *
00030  */
00031 void RequestAuthentication(int sock, PCSTR name);
00032
00033 #endif
00034
00035 /* @} */
```

24.377 httppost.h File Reference

NetBurner HTTP Web Server Post handling Header File.

```
#include <http.h>
#include <json_lexer.h>
#include <nettypes.h>
#include <config_obj.h>
```

Classes

- class CallBackFunctionPostHandler

  Implements the HtmlPostHandler class as a function pointer callback for POST requests.
- class HtmlPostVariableListCallback

  Implements the HtmlPostVariableListHandler class as a function pointer callback for HTTP POST submissions.

Typedefs

- typedef int() http_posthandler(int sock, HTTP_Request &httpReqInfo)

  Type definition of the HtmlPostHandler callback for POST requests.

24.377.1 Detailed Description

NetBurner HTTP Web Server Post handling Header File.

24.378 httppost.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00012 #ifndef _NB_HTTPPOST_H
00013 #define _NB_HTTPPOST_H
00014
00015 #include <http.h>
00016 #include <json_lexer.h>
00017 #include <nettypes.h>
00018 #include <config_obj.h>
00019
00020
00039 typedef int(http_posthandler)(int sock, HTTP_Request &httpReqInfo);
00040
00041 // To handle posts for a specific URL, build a posthandler object for that URL.
00042 // This is the base class for all post handlers. A NULL name will be a catch all for all posts.
00043 class HtmlPostHandler : public HtmlPageHandler
00044 {
00045    public:
00046     HtmlPostHandler(const char *url, int accessGroup = 0) : HtmlPageHandler(url, tPost, accessGroup,
      false) {}
00047     // This class will do a callback with data for each post to the specified url
00048     virtual int ProcessRaw(int sock, HTTP_Request &pd) = 0;
00049 };
00050
00051 // This implements the above as a function pointer call back
```

```
00052
00057 class CallBackFunctionPostHandler : public HtmlPostHandler
00058 {
00059    protected:
00060     http_posthandler *m_pf;
00061
00062    public:
00063     inline virtual int ProcessRaw(int sock, HTTP_Request &pdt) { return m_pf(sock, pdt); };
00064     inline CallBackFunctionPostHandler(const char *pUrl, http_posthandler *pf, int accessGroup = 0) :
      HtmlPostHandler(pUrl, accessGroup)
00065     {
00066         m_pf = pf;
00067     };
00068 };
00069
00070 struct FilePostStruct
00071 {
00072     char FileText[5];        //< Has the text FILE\0
00073     int fd;                  //< File descriptor of the file, only valid during the duration of the
      callback
00074     const char *pFileName;   //< Name of file
00075     const char *pType;       //< Pointer to MIME type of file
00076 };
00077
00078 enum PostEvents
00079 {
00080     eStartingPost,   //< Occurs one time before variables are processed
00081     eVariable,       //< Occurs for each variable
00082     eFile,           //< Occurs if a file is being processed
00083     eEndOfPost       //< Occurs one time at the end of POST processing
00084 };
00085
00086 // This class will provide a virtual function call with a list of all variable values.
00087 class HtmlPostVariableListHandler : public HtmlPostHandler
00088 {
00089     HTTP_Request *m_pCurRequest;
00090
00091    public:
00092     HTTP_Request *GetCurRequest() { return m_pCurRequest; };
00093     virtual int ProcessRaw(int sock, HTTP_Request &pd);
00094
00095
00103     HtmlPostVariableListHandler(const char *pUrl, int accessGroup = 0) : HtmlPostHandler(pUrl,
      accessGroup){};
00104
00105     // Called back with each name/value pair.
00106     // Called back with name="Start" for start.
00107     // Called back with both name and value null for last post.
00108
00118     virtual void ProcessPostVariables(int sock, PostEvents event, const char *pNames, const char
      *pValues) = 0;
00119 };
00120
00121 // This class implements the above with a function call back.
00122 typedef void(postvarhandler)(int sock, PostEvents event, const char *pNames, const char *pValue);
00123
00128 class HtmlPostVariableListCallback : public HtmlPostVariableListHandler
00129 {
00130    protected:
00131     postvarhandler *m_pf;
00132
00133    public:
00134     inline void ProcessPostVariables(int sock, PostEvents event, const char *pName, const char
      *pValue)
00135     {
00136         m_pf(sock, event, pName, pValue);
00137     };
00138
00147     inline HtmlPostVariableListCallback(const char *pUrl, postvarhandler *pCallback, int accessGroup =
      0)
00148         : HtmlPostVariableListHandler(pUrl, accessGroup), m_pf(pCallback){};
00149 };
00150
00151 //---------------------------------------------------------------------------------------------------
00152 // JSON post handlers
00153 //---------------------------------------------------------------------------------------------------
00154
00155 // Handle JSON posts, virtual function call back...
00156 class JsonPostHandler : public HtmlPostHandler
00157 {
00158    protected:
00159     virtual void HandleJson(int sock, ParsedJsonDataSet &JsonSet) = 0;
00160
00161    public:
00162     JsonPostHandler(const char *pUrl, int accessGroup = 0) : HtmlPostHandler(pUrl, accessGroup){};
00163     virtual int ProcessRaw(int sock, HTTP_Request &pd);
00164 };
```

```
00165
00166 // Handle JSON posts function pointer call back...
00167 typedef void(jsonpostvarhandler)(int sock, ParsedJsonDataSet &JsonSet);
00168
00169 class JsonPostCallbackHandler : public JsonPostHandler
00170 {
00171     jsonpostvarhandler *m_pf;
00172
00173   public:
00174     inline void HandleJson(int sock, ParsedJsonDataSet &JsonSet)
00175     {
00176         m_pf(sock, JsonSet);
00177         ;
00178     };
00179     inline JsonPostCallbackHandler(const char *pUrl, jsonpostvarhandler *pCallback, int accessGroup =
      0)
00180         : JsonPostHandler(pUrl, accessGroup), m_pf(pCallback){};
00181 };
00182
00183 // Handle form posts that might contain config items in the form from CONFIGENTRY, FULL or CONFIGTABLE
      in the html.
00184 // Retuns true if it handled the event and no farther processing is required.
00185 bool HandleConfigFormEvent(PostEvents event, const char *pName, const char *pValue);
00186
00187 // This class will process post that ONLY have config variables in them.
00188 class HtmlPostConfigVariableHandler : public HtmlPostVariableListHandler
00189 {
00190     const char *m_pRedirect_url;
00191
00192   public:
00193     HtmlPostConfigVariableHandler(const char *pUrl, const char *pRedirect_Url = 0, int accessGroup =
      0)
00194         : HtmlPostVariableListHandler(pUrl, accessGroup)
00195     {
00196         m_pRedirect_url = pRedirect_Url;
00197     };
00198
00199     // Called back with each name/value pair.
00200     // Called back with name="Start" for start.
00201     // Called back with both name and value null for last post.
00202     virtual void ProcessPostVariables(int sock, PostEvents event, const char *pNames, const char
      *pValues);
00203 };
00204
00205
00206 class CustomConfigFormHandler
00207 {
00208 static CustomConfigFormHandler * pHead;
00209 CustomConfigFormHandler * pNext;
00210 const char * pTypeName;
00211 protected:
00212 CustomConfigFormHandler(const char * pTypeName);
00213 public:
00214 virtual  void RenderValue(int fd, config_leaf *pl, int len, const char *extra)=0;
00215 virtual  void RenderInput(int fd, config_leaf *pl, int len, const char *extra)=0;
00216 virtual  bool ProcessValue(const char * pValue,config_leaf * pl)=0;
00217 static  CustomConfigFormHandler * Find(const NBString &type_name);
00218 };
00219
00220
00221
00222 #endif
00223
00224
00225
00226
00227
```

24.379 https.h File Reference

NetBurner HTTPS Secure Web Server Header File.

```
#include <basictypes.h>
```

Functions

- void StartHttps (uint16_t ssl_port, uint16_t http_port)

  Start the HTTPS secure web server.

24.379.1 Detailed Description

NetBurner HTTPS Secure Web Server Header File.

24.380 https.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00013 #ifndef _NB_HTTPS_H_
00014 #define _NB_HTTPS_H_
00015 #include <basictypes.h>
00016
00032 void StartHttps(uint16_t ssl_port, uint16_t http_port);
00033
00034 #endif
00035
```

24.381 ieee802.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _IEEE802_H_
00006 #define _IEEE802_H_
00007 #include <nettypes.h>
00008
00009 /*
00010  ******************************************************************************
00011  *
00012  * All structures are in network order (big-endian)
00013  *
00014  ******************************************************************************
00015  */
00016
00017 /*
00018  ******************************************************************************
00019  *
00020  * Constant definitions
00021  *
00022  ******************************************************************************
00023  */
00024 /*  Destination service access point */
00025 #define IEEE802_DSAP_SNAP_PACKET_UNNUMBERED (0xAA)
00026
00027 /*  Source service access point */
00028 #define IEEE802_SSAP_SNAP_PACKET_UNNUMBERED (0xAA)
00029
00030 /*  Source service access point */
00031 #define IEEE802_CONTROL_SNAP_PACKET_UNNUMBERED (0x03)
00032
00033 /*
00034    OUI
00035       IEEE802_OUI_ETHERNET_TYPE  00-00-00
00036       IEEE801_OUI_DEC            00-00-F8
00037
00038    Notes:
00039       Only OUI byte 3 changes so far
00040
00041  */
00042 #define IEEE802_OUI_ETHERNET_TYPE (0x00)
00043 #define IEEE801_OUI_DEC (0xF8)
00044
00045 /* OUI length */
00046 #define IEEE802_P80211_OUI_LEN (3)
00047
00048 /*
00049    IEEE 802.11 Control Frame
00050       Type
00051          IEEE80211_TYPE_MANAGEMENT     - Management
00052          IEEE80211_TYPE_CONTROL        - Control
00053          IEEE80211_TYPE_DATA           - Data
00054
00055       Subtype
00056          IEEE80211_SUBTYPE_DATA        - Data
00057          IEEE80211_SUBTYPE_NULL        - No data
00058          IEEE80211_SUBTYPE_QOS         - QOS
00059
```

```
00060  */
00061 #define IEEE80211_TYPE_MANAGEMENT (0x0)
00062 #define IEEE80211_TYPE_CONTROL (0x1)
00063 #define IEEE80211_TYPE_DATA (0x2)
00064
00065 #define IEEE80211_SUBTYPE_DATA (0x0)
00066 #define IEEE80211_SUBTYPE_NULL (0x4)
00067 #define IEEE80211_SUBTYPE_QOS (0x8)
00068
00069 /*
00070  ****************************************************************************
00071  *
00072  * Structures
00073  *
00074  ****************************************************************************
00075  */
00076 /*
00077    IEEE 802.3 Standard Frame Media Access Control (MAC) Header
00078       destination          - Destination MAC-48
00079       source               - Source MAC-48 address
00080       typeOrLength         - Type or length
00081
00082 */
00083 typedef struct _Ieee802_3_Header
00084 {
00085     MACADDRESS_48 destination;
00086     MACADDRESS_48 source;
00087     beuint16_t typeOrLength;
00088
00089 } __attribute__((packed)) Ieee802_3_Header;
00090
00091 /*
00092    IEEE 802.2 Logical Link Control (LLC) +
00093          Subnetwork Access Protocol (SNAP) header
00094
00095       dsap                 - Destination service access point
00096       ssap                 - Source service access point
00097       control              - Control
00098       oui                  - Organizationally Unique Identfier
00099
00100
00101 */
00102 typedef struct _Ieee802_2_SnapHeader
00103 {
00104     /* LLC for SNAP Header 0xAA 0xAA 0x03 */
00105     uint8_t dsap;
00106     uint8_t ssap;
00107     uint8_t control;
00108
00109     /* Organizational Unique Identifier */
00110     uint8_t oui[IEEE802_P80211_OUI_LEN];
00111
00112     /* Typically EtherType IPV4 (0x0800) */
00113     beuint16_t protocolType;
00114
00115 } __attribute__((packed)) Ieee802_2_SnapHeader;
00116
00117 /*
00118  ****************************************************************************
00119  * IEEE 802.11
00120  ****************************************************************************
00121  */
00122 /*
00123    IEEE 802.11 Packet Header Frame Control Field
00124       order                - 1 frames strictly ordered
00125       wep                  - 1 yes, 0 no
00126       more                 - More data for destination
00127       power                - 1 power save, 0 no after transmit
00128       retry                - Retransmission of previous fragment
00129       fragmented           - 1 more fragment frames to follow, 0 last frame
00130       fromDs               - Incoming from distribution system
00131       toDs                 - Forward to distribution system
00132       subType              - Subtype
00133       type                 - Type
00134       version              - Currently 0
00135
00136  */
00137 typedef struct _FrameControl
00138 {
00139     uint16_t order : 1;
00140     uint16_t wep : 1;
00141     uint16_t more : 1;
00142     uint16_t power : 1;
00143     uint16_t retry : 1;
00144     uint16_t fragmented : 1;
00145     uint16_t fromDs : 1;
00146     uint16_t toDs : 1;
```

```
00147     uint16_t subType : 4;
00148     uint16_t type : 2;
00149     uint16_t version : 2;
00150
00151 } __attribute__((packed)) FrameControl;
00152
00153 /*
00154    IEEE 802.11 Packet Header Format
00155       frameControl         - Frame control
00156       durationId           - Data frame duration, control id of transmitter
00157       address[1|2|3|4}     - MAC-48 address based on control frame from/to DS
00158       sequence             - sequence control
00159
00160    Notes:
00161       Precedes data followed by 4 byte frame sequence check defined in P802.11
00162       Addresses
00163           ---------------------------------------------------------------
00164          |  To DS  | From DS | Address 1 | Address 2 | Address 3 | Address 4 |
00165           ---------------------------------------------------------------
00166          |    0    |    0    |    DA     |    SA     |   BSSID   |    N/A    |
00167          |    0    |    1    |    DA     |   BSSID   |    SA     |    N/A    |
00168          |    1    |    0    |   BSSID   |    SA     |    DA     |    N/A    |
00169          |    1    |    1    |    RA     |    TA     |    DA     |    SA     |
00170
00171       ONLY defined for 3 addresses, the 4 address header is used AP to AP
00172
00173  */
00174 typedef struct _Ieee802_11_Header
00175 {
00176     FrameControl frameControl;
00177     beuint16_t durationId;
00178     MACADDRESS_48 address1;
00179     MACADDRESS_48 address2;
00180     MACADDRESS_48 address3;
00181     beuint16_t sequence;
00182
00183 } __attribute__((packed)) Ieee802_11_Header;
00184
00185 #endif /* _IEEE802_H_ */
```

24.382 includes.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _INCLUDES_H_
00006 #define _INCLUDES_H_
00007
00008 // NB Definitions
00009 #include <predef.h>
00010
00011 // NB Libs
00012 #include <basictypes.h>
00013 #include <nbrtos.h>
00014 #include <nbrtoscpu.h>
00015
00016 #endif /* _INCLUDES_H_ */
```

24.383 init.h File Reference

NetBurner System Initialization Header File.

```
#include <predef.h>
#include <constants.h>
#include <basictypes.h>
#include <ctype.h>
#include <stdio.h>
```

Functions

- void init ()

  System initialization. Normally called at the beginning of all applications.

- void StartHttp (uint16_t port=80)

Start the HTTP web server.

- void StartHttps (uint16_t ssl_port=443, uint16_t http_port=80)

  Start the HTTPS secure web server.

- bool WaitForActiveNetwork (uint32_t ticks_to_wait=120 ∗TICKS_PER_SECOND, int interface=-1)

  Wait for an active network connection on at least one interface.

- void EnableSystemDiagnostics ()

  Turn on the diagnostic reports from the config page.

- void EnableSecureConfigServer (bool bSec_Only)

  Enable the minimal http config server to operate over TLS.

24.383.1 Detailed Description

NetBurner System Initialization Header File.

24.384 init.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00013 #ifndef _INIT_H
00014 #define _INIT_H
00015
00016 // NB Definitions
00017 #include <predef.h>
00018
00019 // NB Constants
00020 #include <constants.h>
00021
00022 // NB Libs
00023 #include <basictypes.h>
00024 #include <ctype.h>
00025 #include <stdio.h>
00026
00039 void init();
00040
00052 void StartHttp(uint16_t port = 80);
00053
00067 void StartHttps(uint16_t ssl_port = 443, uint16_t http_port = 80);
00068
00079 bool WaitForActiveNetwork(uint32_t ticks_to_wait = 120 * TICKS_PER_SECOND, int interface = -1);
00080
00081
00089 void EnableSystemDiagnostics();
00090
00091
00092
00093
00094 #ifdef NB_SSL_SUPPORTED
00109 void EnableSecureConfigServer(bool bSec_Only);
00110 #endif /* #ifdef NB_SSL_SUPPORTED */
00111 #endif
00112
```

24.385 IntervalTimer.h File Reference

NetBurner Interval Timer API.

```
#include <constants.h>
#include <nbrtos.h>
```

Functions

- int IntervalOSSem (OS_SEM ∗p_toSem, int num_per_sec, int timer=FIRST_UNUSED_TIMER)

  Posts to a semaphore at the requested interval.

- int IntervalOSFlag (OS_FLAGS ∗p_toFlag, uint32_t flag_value, int num_per_sec, int timer=FIRST_↩ UNUSED_TIMER)

Sets a flag at requested interval.

- int IntervalInterruptCallback (void(*p_toCallbackFunc)(), int num_per_sec, int timer=FIRST_UNUSED_↩ TIMER)

  Calls a function at requested interval. Note that the callback function is called from within the timer's interrupt handler so you should treat your callback function as an interrupt.

- void IntervalStop (int timer_number)

  Stops an existing Interval Timer and frees the resource.

24.385.1 Detailed Description

NetBurner Interval Timer API.

24.386 IntervalTimer.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00041 #ifndef _NB_INTERVAL_TIMER_H
00042 #define _NB_INTERVAL_TIMER_H
00043 #include <constants.h>
00044 #include <nbrtos.h>
00045
00055 int IntervalOSSem(OS_SEM *p_toSem, int num_per_sec, int timer = FIRST_UNUSED_TIMER);
00056
00067 int IntervalOSFlag(OS_FLAGS *p_toFlag, uint32_t flag_value, int num_per_sec, int timer =
      FIRST_UNUSED_TIMER);
00068
00080 int IntervalInterruptCallback(void (*p_toCallbackFunc)(), int num_per_sec, int timer =
      FIRST_UNUSED_TIMER);
00081
00088 void IntervalStop(int timer_number);
00089
00090 #endif
00091
```

24.387 iointernal.h File Reference

Extra File Descriptors.

Functions

- int GetExtraFD (void *extra_data, struct IoExpandStruct *pFuncs)

  Returns a file descriptor for the structure passed as the `IoExpandStruct`. FreeExtraFd( ) will release the fd back to the pool of available fds.

- void * GetExtraData (int fd)

  Returns the `extra` structure value from `IoExpandStruct` associated with the file descriptor.

- void FreeExtraFd (int fd)

  Free a file descriptor and associated resources.

- int GetFreeExtraFDCount ()

  Returns the number of free file descriptors.

- int GetFreeSocketCount (void)

  Returns the number of free sockets.

24.387.1 Detailed Description

Extra File Descriptors.

24.388 iointernal.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00023 #ifndef _NB_IOINTERNALS_H
00024 #define _NB_IOINTERNALS_H
00025
00026 /*
00027  ********************************************************************************
00028  *
00029  * Definitions
00030  *
00031  ********************************************************************************
00032  */
00033 #define IOINTERNALS_FRAMEWORK_TAG ((unsigned int)0x10127EB6)
00034
00035 /*
00036  ********************************************************************************
00037  *
00038  * Structures
00039  *
00040  ********************************************************************************
00041  */
00042 /*
00043  * Extra FD Framework
00044  *    tag                   - Unique tag (Must be IOINTERNALS_FRAMEWORK_TAG)
00045  *    ioSocketNumber        - I/O socket number
00046  *    extraSocketNumber     - Extra socket number
00047  *
00048  *    Notes:
00049  *       Must be at the very beginning of any extra data passed to GetExtraFD
00050  */
00051 typedef struct _IoFrameworkStruct
00052 {
00053     unsigned int tag;
00054     int ioSocketNumber;
00055     int extraSocketNumber;
00056
00057 } __attribute__((packed)) IoFrameworkStruct;
00058
00059 /*
00060  * I/O Expansion Structure
00061  *    read                  - read function
00062  *    write                 - write function
00063  *    close                 - close function
00064  *    extra                 - Control struct starts with IoFrameworkStruct
00065  *
00066  *    Notes:
00067  *       Expanded routines for library read, write and close
00068  */
00069 struct IoExpandStruct
00070 {
00071     int (*read)(int fd, char *buf, int nbytes);
00072     int (*write)(int fd, const char *buf, int nbytes);
00073     int (*close)(int fd);
00074     int (*peek)(int fd, char *buf);
00075     void *extra;
00076 } __attribute__((packed));
00077
00078 /*******************************************************************************
00079
00080    Acquire/Release/Access expansion file descriptor
00081
00082       Parameters:
00083          extra_data        - Control data structure beginning with
00084                              IoFrameworkStruct
00085          pFuncs            - Expanded I/O routines
00086
00087          fd                - Expanded file descriptor
00088
00089       Return:
00090          Acquired fd > EXTRA_IO_OFFSET OK, else -1 for none
00091
00092  *******************************************************************************/
00093
00104 int GetExtraFD(void *extra_data, struct IoExpandStruct *pFuncs);
00105
00113 void *GetExtraData(int fd);
00114
00121 void FreeExtraFd(int fd);
00122
00129 int GetFreeExtraFDCount();
00130
```

```
00137 int GetFreeSocketCount(void);
00138
00139 /*
00140   ***************************************************************************
00141
00142     I/O subsystem notification support for use by expanded I/O routines
00143
00144        Parameters:
00145           fd                - Expanded file descriptor
00146
00147        Return:
00148           None
00149
00150   ***************************************************************************
00151   */
00152 void SetDataAvail(int fd);
00153 void ClrDataAvail(int fd);
00154
00155 void SetWriteAvail(int fd);
00156 void ClrWriteAvail(int fd);
00157
00158 void SetHaveError(int fd);
00159 void ClrHaveError(int fd);
00160
00161 #endif
00162
```

24.389 iosys.h File Reference

NetBurner I/O System Library API.

```
#include <constants.h>
#include <stdio.h>
#include <basictypes.h>
```

Macros

- #define **IOCTL_TX_CHANGE_CRLF** (1)

  When set, transmitted char \n gets converted to \r\n
- #define **IOCTL_RX_CHANGE_CRLF** (2)

  When set, received \r\n get turned into \n
- #define **IOCTL_RX_PROCESS_EDITS** (4)

  When set, process backspace and do simple line editing.
- #define **IOCTL_RX_ECHO** (8)

  When set, echo chars received to tx.
- #define **IOCTL_TX_NO_BLOCK** (32)

  When set, stdout and stderr will drop output instead of blocking.
- #define **IOCTL_ALL_OPTIONS** (15)

  When set, turns on all options.
- #define **IOCTL_SET** (0x4000)

  Set an option.
- #define **IOCTL_CLR** (0x2000)

  Clear an option.

Typedefs

- typedef void FDCallBack(int fd, FDChangeType change, void *pData)

  Define the function signature for file descriptor notification callbacks.

Enumerations

- enum FDChangeType { eReadSet , eWriteSet , eErrorSet }

  The notifications that a registered FD monitor can receive.

Functions

- int close (int fd)

  Close the specified file descriptor and free the associated resources.
- int read (int fd, char *buf, int nbytes)

  Read data from a file descriptor (fd). This function will block forever until at least one byte is available to be read (as opposed to the ReadWithTimeout() function which reads data from a file descriptor with a specified time-out value). This function can be used to read from stdio, TCP sockets, or Serial ports.
- int peek (int fd, char *c)

  Peek at the data for the specified file descriptor (fd). Will block forever until at least one byte is available to be read. The byte returned is not removed from the fd buffer, it will be the first byte of data returned on any subsequent read operation.
- int write (int fd, const char *buf, int nbytes)

  Write data to the stream associated with a file descriptor (fd). Can be used to write data to stdio, a TCP socket, or a Serial port.
- int writestring (int fd, const char *str)

  Write a null terminated ascii string to the stream associated with a file descriptor (fd). Can be used to write data to stdio, a TCP socket, or a Serial port.
- int writeall (int fd, const char *buf, int nbytes=0)

  Write the specified number of bytes to a file descriptor. Will block until all bytes are sent, or a file descriptor error occurs (such as a TCP socket error). Can be used to write data to stdio, a TCP socket, or a Serial port.
- int ReadWithTimeout (int fd, char *buf, int nbytes, unsigned long timeout)

  Read data from a file descriptor(fd), or return if at least one byte is not read within the specified timeout. Can be used instead of the read() function, which will block forever until at least one byte is read. File descriptors include such things as stdio, TCP sockets, TLS sockets, and Serial ports.
- int ReadWithTickTimeout (int fd, char *buf, int nbytes, TickTimeout &timeout)

  Same as ReadWithTimeout(), except the timeout value parameter is of type TickTimeout instead of an unsigned long. The TickTimeout object is more robust in terms of sequential timeout calls and rollover protection.
- int ReadAllWithTimeout (int fd, char *buf, int nbytes, unsigned long timeout)

  Attempt to read the specified number of bytes from a file descriptor, or return with the number of bytes read if the timeout value has expired.
- int ReadAllWithTickTimeout (int fd, char *buf, int nbytes, TickTimeout &timeout)

  Same as ReadWithTimeout(), except the timeout value parameter is of type TickTimeout instead of an unsigned long. The TickTimeout object is more robust in terms of sequential timeout calls and rollover protection.
- int readall (int fd, char *buf, int nbytes)

  Read the specified number of bytes from a file descriptor (fd). This function will block until either the requested number of bytes have been read, or a file descriptor error occurs. It can be used to read from stdio, TCP sockets, or Serial ports.
- int PeekWithTimeout (int fd, char *c, unsigned long timeout)

  This function peeks at data from a file descriptor (fd), with a specified timeout value (as opposed to the peek function which will block forever until at least one byte is available to be read). This function can be used to peek from stdio, TCP sockets, or Serial ports.
- int dataavail (int fd)

  Check the specified file descriptor to detmeine if data is available to be read.
- int writeavail (int fd)

  Check the specified file descriptor to determine data can be written.
- int haserror (int fd)

  Check if a file descriptor has an error.
- int charavail ()

  Checks to see if data is available for read on stdin. By default, stdin is the boot/debug serial port.
- void RegisterFDCallBack (int fd, FDCallBack *fp, void *pData)

  Register a callback function to receive notification when an FD state changes. Register a NULL fp to remove the notification.
- void FD_ZERO (fd_set *pfds)

  Clear (set to 0) a fd_set (file descriptor set) so no file descriptors (fds) are selected.

- void FD_CLR (int fd, fd_set *pfds)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function clears or removes a specific file descriptor in an fd_set.
- void FD_SET (int fd, fd_set *pfds)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function sets or adds a specific file descriptor to an fd_set.
- int FD_ISSET (int fd, fd_set *pfds)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function checks whether or not a particular fd is set in the specified fd_set.
- int FD_OVERLAP (const fd_set *f1, const fd_set *f2)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function determines whether there exists a common file descriptor between the two sets.
- void FD_COPY (const fd_set *from, fd_set *to)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function copies one file descriptor set to another.
- void FD_SETFROMSET (const fd_set *from, fd_set *to)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function adds the file descriptors from one set to another.
- void FD_CLRFROMSET (const fd_set *from, fd_set *to)

  A fd_set (file descriptor set) holds a set of file descriptors (fds). This function remove the file descriptors in one set from another.
- int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, unsigned long timeout)

  Wait for events to occur on one or more I/O resources associated with a set of file descriptors (fds), such as data available to be read, a resource is available to write data, or an error has occurred.
- int ZeroWaitSelect (int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds)

  Returns whether events have occurred on one or more I/O resources associated with a set of file descriptors (fds), and returns immediately.
- int ioctl (int fd, int cmd)

  This function controls the selection of input/output control options for stdio: stdin = 0, stdout = 1 and stderr = 2.
- int ReplaceStdio (int stdio_fd, int new_fd)

  Maps stdio to any file descriptor (fd).
- int CurrentStdioFD (int stdio_fd)

  Returns the current file descriptor mapped to the stdio file descriptor.
- void **IrqStdio** ()

  Open the system default serial port in interrupt mode using the system default baud rate, and assign this serial port to stdin, stdout and stderr.

24.389.1 Detailed Description

NetBurner I/O System Library API.

24.390 iosys.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00018 #ifndef _NB_IOSYS_H
00019 #define _NB_IOSYS_H
00020
00021 #ifndef _CONSTANTS_H
00022 #include <constants.h>
00023 #endif
00024
00025 #include <stdio.h>
00026
00027 #ifndef _BASICTYPES_H_
00028 #include <basictypes.h>
00029 #endif
00030
```

```
00031 class TickTimeout;
00032 #ifdef __cplusplus
00033 extern "C"
00034 {
00035 #endif
00036
00050     int close(int fd);
00051
00074     int read(int fd, char *buf, int nbytes);
00075
00096     int peek(int fd, char *c);
00097
00124     int write(int fd, const char *buf, int nbytes);
00125
00146     int writestring(int fd, const char *str);
00147
00169     int writeall(int fd, const char *buf, int nbytes = 0);
00170
00206     int ReadWithTimeout(int fd, char *buf, int nbytes, unsigned long timeout);
00207
00242     int ReadWithTickTimeout(int fd, char *buf, int nbytes, TickTimeout &timeout);
00243
00278     int ReadAllWithTimeout(int fd, char *buf, int nbytes, unsigned long timeout);
00279
00315     int ReadAllWithTickTimeout(int fd, char *buf, int nbytes, TickTimeout &timeout);
00316
00340     int readall(int fd, char *buf, int nbytes);
00341
00367     int PeekWithTimeout(int fd, char *c, unsigned long timeout);
00368
00383     int dataavail(int fd);
00384
00398     int writeavail(int fd);
00399
00413     int haserror(int fd);
00414
00427     int charavail();
00428
00429 #ifdef fd_set
00430 #undef fd_set
00431 #endif
00432
00433 #ifdef FD_ZERO
00434 #undef FD_ZERO
00435 #endif
00436
00437 #ifdef FD_CLR
00438 #undef FD_CLR
00439 #endif
00440
00441 #ifdef FD_SET
00442 #undef FD_SET
00443 #endif
00444
00445 #ifdef FD_ISSET
00446 #undef FD_ISSET
00447 #endif
00448
00449 #ifdef FD_SETSIZE
00450 #undef FD_SETSIZE
00451 #endif
00452
00453 #define FD_SETSIZE (FDSET_ELEMENTS * 32)
00454
00455     class OS_SEM;
00456
00457     typedef struct
00458     {
00459         uint32_t fd_set_elements[FDSET_ELEMENTS];
00460     } __attribute__((packed)) fd_set;
00461
00470     enum FDChangeType
00471     {
00472         eReadSet,
00473         eWriteSet,
00474         eErrorSet
00475     };
00490     typedef void FDCallBack(int fd, FDChangeType change, void *pData);
00491
00505     void RegisterFDCallBack(int fd, FDCallBack *fp, void *pData);
00506
00518     void FD_ZERO(fd_set *pfds);
00519
00533     void FD_CLR(int fd, fd_set *pfds);
00534
00548     void FD_SET(int fd, fd_set *pfds);
00549
```

```
00575     int FD_ISSET(int fd, fd_set *pfds);
00576
00591     int FD_OVERLAP(const fd_set *f1, const fd_set *f2);
00592
00606     void FD_COPY(const fd_set *from, fd_set *to);
00607
00621     void FD_SETFROMSET(const fd_set *from, fd_set *to);
00622
00636     void FD_CLRFROMSET(const fd_set *from, fd_set *to);
00637
00638     // A Select that uses an external SEM so other things can wake it up.
00639     int extern_sem_select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, OS_SEM &sem,
      unsigned long timeout);
00640
00641     int textern_sem_select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, OS_SEM &sem,
      TickTimeout &timeout);
00642
00684     int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, unsigned long timeout);
00685
00686     int tselect(int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, TickTimeout &timeout);
00687
00710     int ZeroWaitSelect(int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds);
00711
00712 /*********************************************************************/
00713 /*   Define sets of flags to pass to the ioctl function              */
00714 /*********************************************************************/
00715
00720 #define IOCTL_TX_CHANGE_CRLF (1)
00721 #define IOCTL_RX_CHANGE_CRLF (2)
00722 #define IOCTL_RX_PROCESS_EDITS (4)
00723 #define IOCTL_RX_ECHO (8)
00724 #define IOCTL_TX_NO_BLOCK (32)
00725 #define IOCTL_ALL_OPTIONS (15)
00732 #define IOCTL_SET (0x4000)
00733 #define IOCTL_CLR (0x2000)
00758     int ioctl(int fd, int cmd);
00759
00777     int ReplaceStdio(int stdio_fd, int new_fd);
00778
00786     int CurrentStdioFD(int stdio_fd);
00787
00793     void IrqStdio();
00794
00795 #ifdef __cplusplus
00796 }
00797 #endif
00798
00799 #endif
00800    // groupIOSYS
```

24.391 ip.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Definitions for various IP definitions and structures. */
00006 #ifndef _NB_IP_H
00007 #define _NB_IP_H
00008
00009 // NB Definitions
00010 #include <predef.h>
00011
00012 #ifndef NB_NET_TYPES_H
00013 #include "nettypes.h"
00014 #endif
00015
00016 #ifndef _NB_BUFFERS_H
00017 #include "buffers.h"
00018 #endif
00019 /*
00020  ****************************************************************************
00021  *
00022  * Debugging
00023  *
00024  ****************************************************************************
00025  */
00026 // #define IP_DEBUG              ( 0 )
00027
00028 /*
00029  ****************************************************************************
00030  *
00031  * Definitions
00032  *
00033  ****************************************************************************
```

```
00034  */
00035 /*
00036  * IP Version
00037  */
00038 #define IP_VERSION_MASK (0xF0)
00039 #define IP_VERSION_IPv4 (0x40)
00040 #define IP_VERSION_IPv6 (0x60)
00041
00042 /*
00043  * Header Length
00044  */
00045 #define IP_HEADER_LENGTH_MASK (0x0F)
00046 #define IP_HEADER_LENGTH_NO_OPTIONS (0x05)
00047
00048 /*
00049  * Flags and Fragment Offset
00050  */
00051 #define IP_FLAGS_MASK (0xE000)
00052 #define IP_FRAGMENT_OFFSET_MASK (0x1FFF)
00053
00054 /*
00055  * IANA IP protocol numbers
00056  */
00057 #define IP_PROTOCOL_ICMP (0x01)
00058 #define IP_PROTOCOL_IGMP (0x02)
00059 #define IP_PROTOCOL_INTERNET (0x04)
00060 #define IP_PROTOCOL_TCP (0x06)
00061 #define IP_PROTOCOL_UDP (0x11)
00062
00063 /*
00064  * ICMP control message type(s) and code(s)
00065  */
00066 #define ICMP_TYPE_ECHO_REPLY (0)
00067 #define ICMP_CODE_ECHO_REPLY (0)
00068 #define ICMP_TYPE_DEST_UNREACHABLE (3)
00069 #define ICMP_CODE_PORT_UNREACHABLE (3)
00070 #define ICMP_TYPE_SOURCE_QUENCH (4)
00071 #define ICMP_CODE_SOURCE_QUENCH (0)
00072 #define ICMP_TYPE_REDIRECT_MESSAGE (5)
00073 #define ICMP_TYPE_ECHO_REQUEST (8)
00074 #define ICMP_CODE_ECHO_REQUEST (0)
00075
00076 /*
00077  * IANA Well Known Ports
00078  */
00079 #define IANA_ECHO_PORT (7)
00080 #define IANA_DISCARD_PORT (9)
00081 #define IANA_SSH_PORT (22)
00082 #define IANA_TELNET_PORT (23)
00083 //#define IANA_TFTP_PORT (69)
00084 #define IANA_NETBIOS_NAME_SERVICE_PORT (137)
00085
00086 /*
00087  * Data size for ICMP echo request and replay ("ping")
00088  */
00089 #define ICMP_PING_DATA_SIZE (32)
00090
00091 /*
00092  ******************************************************************************
00093  *
00094  * Data Structures
00095  *
00096  ******************************************************************************
00097  */
00098
00099 /*
00100  * IP Header (IPv4)                                    <Internal header members>
00101  *
00102  *    versionNLength       - Version and header length nibbles    <bVerHdrLen>
00103  *    diffServNEcn         - Service and congestion related       <bTos>
00104  *    totalLength          - Datagram size including IP header    <wLength>
00105  *    identification       - Fragment identification              <wpktId >
00106  *    flagsNFragmentOffset - Flags and fragment offset fields     <wFlags_Frag>
00107  *    timeToLive           - Usually hop count                    <bTTL>
00108  *    protocol             - IANA protocol number                 <proto>
00109  *    headerChecksum       - 16-bit one's complement of the one's <hCSum>
00110  *                           complement sum of all 16-bit words in the
00111  *                           header
00112  *    sourceAddress        - IP address of source 4 8-bit octets  <ipSrc>
00113  *    destinationAddress   - IP destination address 4 8-bit octets<ipDst>
00114  *
00115  * Note:
00116  *    Can contain options, determined by header length ( No options header
00117  *    length is 5 for 5 32 bit words ( 20 bytes )
00118  *    IETF RFC 791 September 1981, MIL-STD-1777
00119  *
00120  */
```

```
00121 typedef struct _IpHeaderIPv4
00122 {
00123     uint8_t versionNLength;
00124     uint8_t diffServNEcn;
00125     beuint16_t totalLength;
00126     beuint16_t identification;
00127     beuint16_t flagsNFragmentOffset;
00128     uint8_t timeToLive;
00129     uint8_t protocol;
00130     beuint16_t headerChecksum;
00131     beuint32_t sourceAddress;
00132     beuint32_t destinationAddress;
00133
00134 } __attribute__((packed)) IpHeaderIPv4;
00135
00136 /*
00137  * Internal IPv4 Header
00138  */
00139 typedef struct
00140 {
00141     uint8_t bVerHdrLen;
00142     uint8_t bTos;
00143     beuint16_t wLength;
00144     beuint16_t wpktId;
00145     beuint16_t wFlags_Frag;
00146     uint8_t bTTL;
00147     uint8_t proto;
00148     uint16_t hCSum;    // do not make a big endian var, checksumming is treated as native endian
00149     IPADDR4 ipSrc;
00150     IPADDR4 ipDst;
00151     uint8_t DATA[]; /*The data field is actually as long as the packet.... */
00152 } __attribute__((packed)) IPPKT;
00153
00154 typedef IPPKT *PIPPKT;
00155
00156 /*
00157  * UDP header
00158  *
00159  *    srcPort        - Source port number
00160  *    dstPort        - Destination port number
00161  *    UdpLen         - Datagram length including header
00162  *    UdpCSum        - Checksum (zero in IPv4)
00163  */
00164 typedef struct
00165 {
00166     beuint16_t srcPort;
00167     beuint16_t dstPort;
00168     beuint16_t UdpLen;
00169     uint16_t UdpCSum;    // do not make big endian, see IPPKT
00170     uint8_t DATA[];     /*The data field is actually as long as the packet.... */
00171 } __attribute__((packed)) UDPPKT;
00172 typedef UDPPKT *PUDPPKT;
00173
00174 /*
00175  * UDP IPv4 Pseudo-header segment (bits 0 through 95)
00176  *
00177  *    proto          - Protocol (UDP 0x11)
00178  *    len            - UDP header and data
00179  *    srcip          - Source IP address
00180  *    dstip          - Destination IP address
00181  */
00182
00183 /*******************WARNING ************************
00184
00185 This structure has the elements in a differnt order than usuallyt defined.
00186 The IPChecksum is order invariant on 16 bit values.
00187 The struucture is this way so it can MAP over a section of UDP and TCP IP packets without messing them
      up.
00188 Dont change the order
00189 *****************WARNING ***************************/
00190
00191 typedef struct
00192 {
00193     beuint16_t proto;
00194     beuint16_t len;
00195     IPADDR4 srcip;
00196     IPADDR4 dstip;
00197
00198 } PsudeoHeader;
00199 typedef PsudeoHeader *PPSUH;
00200
00201 /*
00202  ********************************************************************************
00203  *
00204  * Global Data
00205  *
00206  ********************************************************************************
```

```
00207  */
00208 /* Default number of hops (time to live) */
00209 extern uint8_t bTTL_Default;
00210
00211 /* ARP lifetime in seconds */
00212 extern uint16_t wArpLifetime;
00213
00214 /* Quiet start */
00215 extern BOOL bQuietStart;
00216
00217 /*
00218  ****************************************************************************
00219  *
00220  * "C" Internal Functions
00221  *
00222  ****************************************************************************
00223  */
00224
00225 /* IP packet insertion for processing */
00226 void IpProcessEthernetPacket(PoolPtr poolPtr, uint16_t packetSizeInBytes);
00227
00228 extern "C"
00229 {
00230     /* Checksum */
00231     uint16_t GetSum(puint16_t addr, uint16_t count);
00232     /* Checksum */
00233     uint16_t GetSum20(puint32_t addr);
00234     /* Checksum using pseudo-header */
00235     uint16_t GetSumHdr(PsudeoHeader &hdr, puint16_t addr, uint16_t count);
00236 }
00237
00238 /* Is this my IP address? */
00239 BOOL IsMyIp4(IPADDR4 ip, int ifc = -1);
00240
00241 /* Get source IP address for this destination */
00242 IPADDR4 GetSrcIp4(IPADDR4 dst);
00243
00244 IPADDR4 GetSrcIpwIf4(int ifn, const IPADDR4 dst);
00245
00246 #ifdef IPV6
00247 /* Is this my IP address? */
00248 BOOL IsMyIp6(const IPADDR &ip, int ifc = -1);
00249 inline BOOL IsMyIp(const IPADDR &ip, int ifc = -1)
00250 {
00251     return IsMyIp6(ip, ifc);
00252 };
00253
00254 /* Get source IP address for this destination */
00255 IPADDR GetSrcIp6(const IPADDR &dst);
00256 inline IPADDR GetSrcIp(const IPADDR &dst)
00257 {
00258     return GetSrcIp6(dst);
00259 };
00260
00261 IPADDR GetSrcIp6wIf(int ifn, const IPADDR &dst);
00262 inline IPADDR GetSrcIpwIf(int ifn, const IPADDR &dst)
00263 {
00264     return GetSrcIp6wIf(ifn, dst);
00265 };
00266
00267 #else
00268 inline BOOL IsMyIp(IPADDR4 ip, int ifc = -1)
00269 {
00270     return IsMyIp4(ip, ifc);
00271 };
00272 inline IPADDR4 GetSrcIp(IPADDR4 dst)
00273 {
00274     return GetSrcIp4(dst);
00275 };
00276
00277 inline IPADDR4 GetSrcIpwIf(int ifn, IPADDR4 dst)
00278 {
00279     return GetSrcIpwIf4(ifn, dst);
00280 };
00281
00282 #endif
00283
00284 /*
00285  ****************************************************************************
00286  *
00287  * "C++" Internal Functions
00288  *
00289  ****************************************************************************
00290  */
00291 /* Send ICMP error */
00292 void SendICMPError(PoolPtr pBadPacket, uint8_t type, uint8_t code);
00293
```

```
00294 /* Complete header and send on primary network interface */
00295 void FixHeaderAndSend(PoolPtr p, PIPPKT pIp);
00296
00297 /* Complete header and send on an interface */
00298 void FixHeaderAndSendViaInterface(PoolPtr p, PIPPKT pIp, int Interface);
00299
00300 /* Get IP Packet pointer from network buffer pool buffer */
00301 inline PIPPKT GetIpPkt(PoolPtr p)
00302 {
00303     return (p == NULL) ? NULL : (PIPPKT)(p->pData + 14);
00304 };
00305
00306 /* Get IP packet pointer from pointer to frame */
00307 inline PIPPKT GetIpPkt(PEFRAME pFrame)
00308 {
00309     return (PIPPKT)(pFrame->pData);
00310 };
00311
00312 inline PIPPKT GetIpPkt(PVLEFRAME pFrame)
00313 {
00314     return (PIPPKT)(pFrame->pData);
00315 };
00316
00317 /*
00318  * WARNING WARNING WARNING
00319  *
00320  *    If you use these functions on an uninitialized buffer you will get bogus
00321  *    values for the pointer as the header length field in the IP packet is not
00322  *    yet set up!
00323  */
00324 /* Get UPD packet pointer from IP packet pointer */
00325
00326 inline PUDPPKT GetUdpPkt(PIPPKT pIp)
00327 {
00328     if (pIp == NULL) { return NULL; }
00329     if (pIp->bVerHdrLen == 0x45) { return (PUDPPKT)pIp->DATA; }
00330     return (PUDPPKT)(pIp->DATA + (((pIp->bVerHdrLen & 0XF) * 4) - 20));
00331 }
00332
00333 /*
00334  ******************************************************************************
00335  *
00336  * "C++" Runtime Library Functions
00337  *
00338  ******************************************************************************
00339  */
00340 /*
00341  ******************************************************************************
00342  *
00343  * Initializes the IP stack.
00344  *
00345  *    Parameters:
00346  *       ipaddr         - Address of primary network interface
00347  *       ipMask         - Subnetwork mask
00348  *       ipGate         - Gateway address.
00349  *       dns            - Domain name system (DNS)server address
00350  *
00351  *    Return:
00352  *       >=0   Ticks wait for the response
00353  *       -1    Timeout
00354  *
00355  *    Notes:
00356  *       Should be called once and only once before UserMain resets it priority.
00357  *       The default settings cause the configuration record settings to be used.
00358  *       Add addresses and masks are IPv4
00359  *       This provides the network infrastructure for additional network interfaces
00360  *
00361  ******************************************************************************
00362  */
00363 void InitializeStack();
00364
00365 /*
00366  ******************************************************************************
00367  *
00368  * UPD fragment packet pointer for user defined function.
00369  *
00370  *    Parameters:
00371  *       pp             - Packet pointer from network buffer pool
00372  *
00373  *    Return:
00374  *       None
00375  *
00376  *    Notes:
00377  *       This is only called for packets with the fragmentation flag set
00378  *
00379  ******************************************************************************
00380  */
```

```
00381 typedef void(FragmentProcessFunction)(PoolPtr pp);
00382 extern FragmentProcessFunction *pFragFunc;
00383
00384 /*
00385  ****************************************************************************
00386  *
00387  * Promiscuous packet pointer for user defined function.
00388  *
00389  *    Parameters:
00390  *       pp             - Packet pointer from network buffer pool
00391  *
00392  *    Return:
00393  *       None
00394  *
00395  *    Notes:
00396  *       This is only called for packets that fail the IP address == my address.
00397  *       This function must free the buffers it receives!
00398  *
00399  ****************************************************************************
00400  */
00401 typedef void(PromisciousPacketFunc)(PoolPtr pp);
00402 extern PromisciousPacketFunc *pPromisciousPacketFunc;
00403
00404 /*
00405  ****************************************************************************
00406  *
00407  * UDP task packet function pointer
00408  *
00409  *    Parameters:
00410  *       pkt            - UDP packet
00411  *       ifn            - Interface number
00412  *
00413  *    Return:
00414  *       None
00415  *
00416  *    Notes:
00417  *       None
00418  *
00419  ****************************************************************************
00420  */
00421 class UDPPacket;
00422 typedef int(TaskPacketFunc)(UDPPacket &pkt, int ifn);
00423 extern TaskPacketFunc *pTaskPacketFunc;
00424
00425 /*
00426  ****************************************************************************
00427  *
00428  * Display buffer for diagnostic purposes
00429  *
00430  *    Parameters:
00431  *       rp             - Packet pointer from network buffer pool
00432  *
00433  *    Return:
00434  *       None
00435  *
00436  *    Notes:
00437  *       Uses iprintf
00438  *
00439  ****************************************************************************
00440  */
00441 void ShowIPBuffer(PoolPtr rp);
00442
00443 /*
00444  ****************************************************************************
00445  *
00446  * Sends a "ping" packet and waits the specified timeout for a response.
00447  *
00448  *    Ping uses the primary network interface (usually the Ethernet)
00449  *    PingViaInterface allows the selection of network interface
00450  *    SendPing does not wait for a response.
00451  *
00452  *    Parameters:
00453  *       to             - Destination address
00454  *       id             - Identifier
00455  *       seq            - Sequence number
00456  *       maxwaitticks   - Wait time in ticks
00457  *
00458  *    Return:
00459  *       >=0   Ticks wait for the response
00460  *       -1    Timeout
00461  *       -2    Error other than timeout.
00462  *    Notes:
00463  *       Sends ICMP echo request with specified identifier, sequence number and
00464  *       32 bytes of ASCII data ('a' (0x61) through 0xFF ).
00465  *       Expects echo reply with same data
00466  *
00467  ****************************************************************************
```

```
00468  */
00469 int Ping4(IPADDR4 to, uint16_t id, uint16_t seq, uint16_t maxwaitticks);
00470 int Ping4ViaInterface(IPADDR4 to, uint16_t id, uint16_t seq, uint16_t wait, int interface);
00471
00472 #ifdef IPV6
00473 int Ping6(const IPADDR &to, uint16_t id, uint16_t seq, uint16_t maxwaitticks, int size = 32);
00474 int Ping6ViaInterface(const IPADDR &to, uint16_t id, uint16_t seq, uint16_t wait, int interface, int
      size = 32);
00475 inline int Ping(const IPADDR &to, uint16_t id, uint16_t seq, uint16_t maxwaitticks, int size = 32)
00476 {
00477     return Ping6(to, id, seq, maxwaitticks, size);
00478 };
00479 inline int PingViaInterface(const IPADDR &to, uint16_t id, uint16_t seq, uint16_t wait, int interface,
      int size = 32)
00480 {
00481     return Ping6ViaInterface(to, id, seq, wait, interface, size);
00482 };
00483 #else
00484 inline int Ping(IPADDR4 to, uint16_t id, uint16_t seq, uint16_t maxwaitticks)
00485 {
00486     return Ping4(to, id, seq, maxwaitticks);
00487 };
00488 inline int PingViaInterface(IPADDR4 to, uint16_t id, uint16_t seq, uint16_t wait, int interface)
00489 {
00490     return Ping4ViaInterface(to, id, seq, wait, interface);
00491 };
00492 #endif
00493
00494 BOOL IsLocal4(IPADDR4 ip, int ifc);
00495 int GetProperInterface4(IPADDR4 dst);
00496
00497 #if defined MULTIHOME || defined AUTOIP
00498 int AddInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, int root_if);
00499 int GetMultiHomeInterface(IPADDR4 ipa, int norgif);
00500 int GetLocalIPInterface4(IPADDR4 ipa);
00501 #ifdef IPV6
00502 int GetLocalIPInterface6(const IPADDR &ipa);
00503 inline int GetLocalIPInterface(const IPADDR &ipa)
00504 {
00505     return GetLocalIPInterface6(ipa);
00506 };
00507 #else
00508 inline int GetLocalIPInterface(IPADDR4 ipa)
00509 {
00510     return GetLocalIPInterface4(ipa);
00511 };
00512 #endif
00513 #endif
00514
00515     /*
00516      ********************************************************************************
00517      *
00518      * Deprecated and Backward Compatibility Runtime Library Functions
00519      *
00520      ********************************************************************************
00521      */
00522
00523 #ifndef IPV6
00524 #ifndef IPV4ONLY
00525 #error Got to pick an IP version
00526 #endif
00527 #endif
00528
00529 // Get a random port value for random return port values
00530 uint16_t GetEphemeralPort();
00531
00532 #endif /* #ifndef _NB_IP_H */
```

24.392 ip_negotiation.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NB_IP_NEGOTIATE_H
00006 #define NB_IP_NEGOTIATE_H
00007 #include <dhcpclient.h>
00008
00009 #define IPC_STATE_STARTUP 1
00010 #define IPC_STATE_STATIC 2
00011 #define IPC_STATE_DCHPCLIENT 3
00012 #define IPC_STATE_FALLBACK 4
00013 #define IPC_STATE_AUTOIP 5
00014 #define IPC_STATE_DHCPSERVER 6
00015 #define IPC_STATE_DHCPCLIENT_OVERRIDE 7
```

```
00016 #define IPC_STATE_AUTOIP_OVERRIDE 8
00017
00018 class IPClient
00019 {
00020    private:
00021     bool useDHCP;
00022     bool dhcpClientRetry;
00023     int IPState;
00024     int clientInterface;
00025
00026     IPADDR4 IPClientIP;
00027     IPADDR4 IPClientMask;
00028     IPADDR4 gatewayIP;
00029     IPADDR4 DNSIP;
00030
00031     DhcpObject dhcpClient;
00032
00033    public:
00034     IPClient(int interface = 0, bool useDhcp = true, bool retry = true);
00035     ~IPClient();
00036     void start();
00037     void stop();
00038     void restart();
00039
00040     IPADDR4 getClientIP();
00041     IPADDR4 getClientMask();
00042     IPADDR4 getGatewayIP();
00043     IPADDR4 getDNSIP();
00044
00045     int getState();
00046 };
00047
00048 #endif
```

24.393 ipshow.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _IPSHOW_H_
00006 #define _IPSHOW_H_
00007
00008 /*-------------------------------------------------------------------------
00009  * Web page function callback to display links to IPv4 and IPv6 addresses.
00010  * Note that IPv6 prefix addresses may take a few seconds to be assigned.
00011  *-------------------------------------------------------------------------*/
00012 extern "C"   // Required for web page callback linking
00013 {
00014     void webShowAddress(int socket, char *url);
00015 }
00016
00017 /*-------------------------------------------------------------------------
00018  * Print IP addresses to serial port
00019  *-------------------------------------------------------------------------*/
00020 void showIpAddresses();
00021
00022 #endif
```

24.394 dhcpv6_const.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __DHCPV6_CONST_H
00006 #define __DHCPV6_CONST_H
00007
00008 #include <constants.h>
00009 #include <nettypes.h>
00010 #include <utils.h>
00011 #include <ipv6/ipv6_addr.h>
00012
00013 #define ADDR_ALL_DHCP_RELAY_AND_SERVERS IPADDR6::AsciiToIp6("FF02::1:2")
00014 #define ADDR_ALL_DHCP_SERVERS IPADDR6::AsciiToIp6("FF05::1:3")
00015
00016 #define UDP_DHCPV6_CLIENT_PORT (546)
00017 #define UDP_DHCPV6_RELAY_AND_SERVER_PORT (547)
00018
00019 #define SOL_MAX_DELAY (1 * TICKS_PER_SECOND)
00020 #define SOL_TIMEOUT (1 * TICKS_PER_SECOND)
00021 #define SOL_MAX_RT (120 * TICKS_PER_SECOND)
```

```
00022 #define REQ_TIMEOUT (1 * TICKS_PER_SECOND)
00023 #define REQ_MAX_RT (30 * TICKS_PER_SECOND)
00024 #define REQ_MAX_RC (10)
00025 #define CNF_MAX_DELAY (1 * TICKS_PER_SECOND)
00026 #define CNF_TIMEOUT (1 * TICKS_PER_SECOND)
00027 #define CNF_MAX_RT (4 * TICKS_PER_SECOND)
00028 #define CNF_MAX_RD (10 * TICKS_PER_SECOND)
00029 #define REN_TIMEOUT (10 * TICKS_PER_SECOND)
00030 #define REN_MAX_RT (600 * TICKS_PER_SECOND)
00031 #define REB_TIMEOUT (10 * TICKS_PER_SECOND)
00032 #define REB_MAX_RT (600 * TICKS_PER_SECOND)
00033 #define INF_MAX_DELAY (1 * TICKS_PER_SECOND)
00034 #define INF_TIMEOUT (1 * TICKS_PER_SECOND)
00035 #define INF_MAX_RT (120 * TICKS_PER_SECOND)
00036 #define REL_TIMEOUT (1 * TICKS_PER_SECOND)
00037 #define REL_MAX_RC (5)
00038 #define DEC_TIMEOUT (1 * TICKS_PER_SECOND)
00039 #define DEC_MAX_RC (5)
00040 #define REC_TIMEOUT (2 * TICKS_PER_SECOND)
00041 #define REC_MAX_RC (8)
00042 #define HOP_COUNT_LIMIT (32)
00043
00044 // Per RFC 4242 Sec 3.1
00045 #define INFO_REQ_MAX_CHECK_DLY (86400 * TICKS_PER_SECOND)
00046 #define INFO_REQ_MIN_CHECK_DLY (600 * TICKS_PER_SECOND)
00047
00048 // NB defined arbitrary Renew And Rebind values, per REFC 3315, Sec 18.1.3
00049 #define DEFAULT_RENEW_PERIOD (3600 * TICKS_PER_SECOND)
00050 #define DEFAULT_REBIND_PERIOD (4800 * TICKS_PER_SECOND)
00051
00052 #endif /* ----- #ifndef __DHCPV6_CONST_H  ----- */
```

24.395 dhcpv6_internal.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __DHCPV6_INTERNAL_H
00006 #define __DHCPV6_INTERNAL_H
00007
00008 #include <constants.h>
00009 #include <nettypes.h>
00010 #include <udp.h>
00011 //#include <ipv6/ipv6_interface.h>
00012 #include <ipv6/dhcpv6_const.h>
00013 #include <ipv6/dhcpv6_msg.h>
00014
00015 #define CLIENT_LONG_SERVID 18
00016
00017 //#define DHCPv6_DEBUG
00018
00019 #ifdef DHCPv6_DEBUG
00020 #define DHCPv6_DBSHOW(msg) msg.ShowMsg()
00021 #define DHCPv6_DBIPRINTF(...)                                                                    \
00022     {                                                                                            \
00023         iprintf("\n[T:%d-%ld]%s:%s:%d::", OSTaskID(), TimeTick, __FILE__, __FUNCTION__, __LINE__); \
00024         iprintf(__VA_ARGS__);                                                                    \
00025     }
00026 #else
00027 #define DHCPv6_DBSHOW(msg) ((void)0)
00028 #define DHCPv6_DBIPRINTF(...) ((void)0)
00029 #endif
00030
00031 class IPv6Interface;
00032 struct IPV6_DHCPD;
00033
00034 namespace NB
00035 {
00036 namespace V6
00037 {
00038 namespace DHCPv6
00039 {
00040 class DHCPv6Message
00041 {
00042    private:
00043     UDPPacket Msg;
00044     DHCP_Message *dhcp_msg;
00045     uint16_t msgLen;
00046     bool txNRx;
00047
00048     uint32_t opts_present[4];   // 128-bit bitfield for option numbers
00049
00050    public:
00051     static const IPADDR6 ALL_RELAY_AND_SERVERS;
```

```
00052     static const IPADDR6 ALL_SERVERS;
00053
00054     DHCPv6Message();
00055     DHCPv6Message(DHCP_Message *msg, uint16_t len);
00056     DHCPv6Message(UDPPacket *msg);
00057     ~DHCPv6Message();
00058
00059 #ifdef DHCPv6_DEBUG
00060     void ShowMsg();   // Display DHCPv6 message for debug
00061 #endif
00062
00063     PoolPtr ReleaseBuffer();
00064
00065     inline void ClrMsg()
00066     {
00067         Msg.ResetData();
00068         dhcp_msg = (DHCP_Message *)Msg.GetDataBuffer();
00069         txNRx = true;
00070     }
00071     inline void SetTypeAndXid(MessageType type, uint32_t xid)
00072     {
00073         ClrMsg();
00074         xid &= 0xFFFFFF;   // 24 bit mask
00075         xid |= type « 24;
00076         beuint32_t tXid = xid;   // needed for Little Endian hosts
00077         Msg.AddData((uint8_t *)&tXid, 4);
00078         msgLen += 4;
00079     }
00080
00081     void AddOption(Option &opt, uint16_t overrideLen = 0);
00082
00083     void AddClientID();
00084     void AddORO(Option_ID ids[], uint16_t len, int ifNum);
00085     void AddElapsedTime(uint16_t elapsedTime);
00086     void AddIA_NA(uint32_t IA_ID, uint32_t T1, uint32_t T2, uint16_t optLen = 0);
00087     void AddFQDN(const char *name);
00088
00089     inline MessageType GetType() { return (MessageType)((dhcp_msg->typeAndXid) » 24); }
00090     inline uint32_t GetXid() { return (dhcp_msg->typeAndXid) & 0xFFFFFF; }
00091
00092     Option *GetOption(Option_ID optID, Option *lastOpt = NULL);
00093     inline IPADDR6 GetSourceAddress() { return Msg.GetSourceAddress(); }
00094
00095     void SendMsg(const IPADDR6 &ServerIP = ALL_RELAY_AND_SERVERS, int via_interface = -1);
00096
00097     static void SendSolicit(uint32_t ifNum, uint32_t xid, uint32_t timeElapsed);
00098     static void SendRequest(uint32_t ifNum, uint32_t xid, uint32_t timeElapsed, Opt::ClientServerID
      *serverID, Opt::IA_NA *ia_na);
00099     static void SendInfoReq(uint32_t ifNum, uint32_t xid);
00100 };
00101
00102 class DHCPClient
00103 {
00104   public:
00105     enum ClientState
00106     {
00107         STATE_SOLICIT_START = 0,
00108         STATE_SOLICIT_DLYED = 1,
00109         STATE_SOLICIT_MSG_REC = 2,
00110         STATE_REQUEST_SENDING = 3,
00111         STATE_CONFIRMED = 4,
00112         STATE_RENEWING = 5,
00113         STATE_REBINDING = 6,
00114         STATE_INFO_REQ_SENDING = 7,
00115         STATE_INFO_REP_REC = 8,
00116         STATE_STOPPED = 9
00117     };
00118
00119   private:
00120     ClientState state;
00121     IPv6Interface *myInterface;
00122     uint32_t nextActionTick;
00123     uint32_t currentDelay;
00124     uint32_t retransmitCount;
00125     uint32_t startTick;
00126     uint32_t xid;
00127     uint8_t maxPrefSeen;
00128     bool hasDns;
00129     IPV6_DHCPD *activeDHCPD;
00130     uint8_t activeServerID[CLIENT_LONG_SERVID];   // Space to contain the ServerID of the leasing
      server
00131
00132     PoolPtr actingMessage;   // DHCP advert/reply to act on
00133
00134     void ClearInfo();
00135     inline void SetState(ClientState newState)
00136     {
```

```
00137         DHCPv6_DBIPRINTF("Setting state: %s", GetStateString(newState));
00138         state = newState;
00139     }
00140     static char const *GetStateString(ClientState state);
00141
00142     void SetCurrentDelay(uint32_t delayBase);
00143     void IncrementCurrentDelay(uint32_t maxDelay);
00144     uint32_t GetNewXid();
00145
00146     void ProcessReply(DHCPv6Message &msg);
00147     void ProcessReply_Info(DHCPv6Message &msg);
00148     void ProcessReply_Solicit(DHCPv6Message &msg, bool needServerMatch);
00149
00150     void ProcessReply_DNSOpt(IPV6_DHCPD *server, Opt::DNS_Servers *dnsOpt, uint32_t time);
00151
00152     void ProcessAdvert(DHCPv6Message &msg);
00153     void ProcessReconfig(DHCPv6Message &msg);
00154
00155     void SendSolicit();
00156     void SendRequest();
00157     void SendRenewRebind(bool renew);
00158
00159   public:
00160     DHCPClient(IPv6Interface *intf);
00161     inline uint32_t GetNextActionTick() { return nextActionTick; }
00162     void Tick();
00163     void ProcessMsg(DHCPv6Message &msg);
00164     void StartInfoReq();
00165     void StartSolicit();
00166     inline ClientState GetState() { return state; }
00167
00168     void ShowInfo();
00169
00170     static void ProcessDHCPAvail(IPv6Interface *intf, uint8_t raFlags);
00171     static void ProcessDHCPMsg(UDPPacket *pkt);
00172 };
00173
00174 // these are in the NB::V6::DHCPv6 namespace
00175 extern Option_ID *(*pAddOROCB)(int ifNum);
00176 extern void (*pRequestOptionCB)(int ifNum, DHCPv6Message &msg);
00177 extern void (*pReplyCB)(int ifNum, DHCPv6Message &msg);
00178
00179 }   // namespace DHCPv6
00180 }   // namespace V6
00181 }   // namespace NB
00182
00183 #endif /* ----- #ifndef __DHCPV6_INTERNAL_H  ----- */
```

24.396 dhcpv6_msg.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __DHCPV6_MSG_H
00006 #define __DHCPV6_MSG_H
00007
00008 #include <constants.h>
00009 #include <nettypes.h>
00010
00011 // See RFC 3315 for all struct definitions
00012
00013 namespace NB
00014 {
00015 namespace V6
00016 {
00017 namespace DHCPv6
00018 {
00019 typedef enum
00020 {
00021     MSG_SOLICIT = 1,
00022     MSG_ADVERTISE = 2,
00023     MSG_REQUEST = 3,
00024     MSG_CONFIRM = 4,
00025     MSG_RENEW = 5,
00026     MSG_REBIND = 6,
00027     MSG_REPLY = 7,
00028     MSG_RELEASE = 8,
00029     MSG_DECLINE = 9,
00030     MSG_RECONF = 10,
00031     MSG_INFO_REQ = 11,
00032     MSG_RELAY_FORW = 12,
00033     MSG_RELAY_REPL = 13,
00034     MSG_LEASEQUERY = 14,
00035     MSG_LEASEQUERY_REPLY = 15,
```

```
00036     MSG_LEASEQUERY_DONE = 16,
00037     MSG_LEASEQUERY_DATA = 17,
00038     MSG_RECONFIGURE_REQUEST = 18,
00039     MSG_RECONFIGURE_REPLY = 19,
00040     MSG_DHCPV4_QUERY = 20,
00041     MSG_DHCPV4_RESPONSE = 21,
00042     MSG_FORCE_16BIT = 0xFFFF
00043 } MessageType;
00044
00045 typedef enum
00046 {
00047     OPT_NULL = 0,
00048     OPT_CLIENTID = 1,
00049     OPT_SERVERID = 2,
00050     OPT_IA_NA = 3,
00051     OPT_IA_TA = 4,
00052     OPT_IA_ADDR = 5,
00053     OPT_ORO = 6,
00054     OPT_PREFERENCE = 7,
00055     OPT_ELAPSED_TIME = 8,
00056     OPT_RELAY_MSG = 9,
00057     OPT_AUTH = 11,
00058     OPT_UNICAST = 12,
00059     OPT_STATUS_CODE = 13,
00060     OPT_RAPID_COMMIT = 14,
00061     OPT_USER_CLASS = 15,
00062     OPT_VENDOR_CLASS = 16,
00063     OPT_VENDOR_OPTS = 17,
00064     OPT_INTERFACEID = 18,
00065     OPT_RECONF_MSG = 19,
00066     OPT_RECONF_ACCEPT = 20,
00067     OPT_SIP_SERVER_D = 21,
00068     OPT_SIP_SERVER_A = 22,
00069     OPT_DNS_SERVERS = 23,
00070     OPT_DOMAIN_LIST = 24,
00071     OPT_IA_PD = 25,
00072     OPT_IA_PREFIX = 26,
00073     OPT_NIS_SERVERS = 27,
00074     OPT_NISP_SERVERS = 28,
00075     OPT_NIS_DOMAIN_NAME = 29,
00076     OPT_NISP_DOMAIN_NAME = 30,
00077     OPT_SNTP_SERVERS = 31,
00078     OPT_INFORMATION_REFRESH_TIME = 32,
00079     OPT_BCMCS_SERVER_D = 33,
00080     OPT_BCMCS_SERVER_A = 34,
00081     OPT_GEOCONF_CIVIC = 36,
00082     OPT_REMOTE_ID = 37,
00083     OPT_SUBSCRIBER_ID = 38,
00084     OPT_CLIENT_FQDN = 39,
00085     OPT_PANA_AGENT = 40,
00086     OPT_NEW_POSIX_TIMEZONE = 41,
00087     OPT_NEW_TZDB_TIMEZONE = 42,
00088     OPT_ERO = 43,
00089     OPT_LQ_QUERY = 44,
00090     OPT_CLIENT_DATA = 45,
00091     OPT_CLT_TIME = 46,
00092     OPT_LQ_RELAY_DATA = 47,
00093     OPT_LQ_CLIENT_LINK = 48,
00094     OPT_MIP6_HNIDF = 49,
00095     OPT_MIP6_VDINF = 50,
00096     OPT_V6_LOST = 51,
00097     OPT_CAPWAP_AC_V6 = 52,
00098     OPT_RELAY_ID = 53,
00099     OPT_IPv6_Address_MoS = 54,
00100     OPT_IPv6_FQDN_MoS = 55,
00101     OPT_NTP_SERVER = 56,
00102     OPT_V6_ACCESS_DOMAIN = 57,
00103     OPT_SIP_UA_CS_LIST = 58,
00104     OPT_BOOTFILE_URL = 59,
00105     OPT_BOOTFILE_PARAM = 60,
00106     OPT_CLIENT_ARCH_TYPE = 61,
00107     OPT_NII = 62,
00108     OPT_GEOLOCATION = 63,
00109     OPT_AFTR_NAME = 64,
00110     OPT_ERP_LOCAL_DOMAIN_NAME = 65,
00111     OPT_RSOO = 66,
00112     OPT_PD_EXCLUDE = 67,
00113     OPT_VSS = 68,
00114     OPT_MIP6_IDINF = 69,
00115     OPT_MIP6_UDINF = 70,
00116     OPT_MIP6_HNP = 71,
00117     OPT_MIP6_HAA = 72,
00118     OPT_MIP6_HAF = 73,
00119     OPT_RDNSS_SELECTION = 74,
00120     OPT_KRB_PRINCIPAL_NAME = 75,
00121     OPT_KRB_REALM_NAME = 76,
00122     OPT_KRB_DEFAULT_REALM_NAME = 77,
```

```
00123     OPT_KRB_KDC = 78,
00124     OPT_CLIENT_LINKLAYER_ADDR = 79,
00125     OPT_LINK_ADDRESS = 80,
00126     OPT_RADIUS = 81,
00127     OPT_SOL_MAX_RT = 82,
00128     OPT_INF_MAX_RT = 83,
00129     OPT_ADDRSEL = 84,
00130     OPT_ADDRSEL_TABLE = 85,
00131     OPT_V6_PCP_SERVER = 86,
00132     OPT_DHCPV4_MSG = 87,
00133     OPT_DHCP4_O_DHCP6_SERVER = 88,
00134     OPT_S46_RULE = 89,
00135     OPT_S46_BR = 90,
00136     OPT_S46_DMR = 91,
00137     OPT_S46_V4V6BIND = 92,
00138     OPT_S46_PORTPARAMS = 93,
00139     OPT_S46_CONT_MAPE = 94,
00140     OPT_S46_CONT_MAPT = 95,
00141     OPT_S46_CONT_LW = 96,
00142     OPT_4RD = 97,
00143     OPT_4RD_MAP_RULE = 98,
00144     OPT_4RD_NON_MAP_RULE = 99,
00145     OPT_LQ_BASE_TIME = 100,
00146     OPT_LQ_START_TIME = 101,
00147     OPT_LQ_END_TIME = 102,
00148     OPT_FORCE_16BIT = 0xFFFF
00149 } Option_ID;
00150
00151 typedef enum
00152 {
00153     STATUS_SUCCESS = 0,
00154     STATUS_UNSPEC_FAIL = 1,
00155     STATUS_NO_ADDRS_AVAIL = 2,
00156     STATUS_NO_BINDING = 3,
00157     STATUS_NOT_ON_LINK = 4,
00158     STATUS_USE_MULTICAST = 5,
00159     STATUS_NO_PREFIX_AVAIL = 6,
00160     STATUS_UNKNOWN_QUERY_TYPE = 7,
00161     STATUS_MALFORMED_QUERY = 8,
00162     STATUS_NOT_CONFIGURED = 9,
00163     STATUS_NOT_ALLOWED = 10,
00164     STATUS_QUERY_TERMINATED = 11,
00165     STATUS_FORCE_16_BIT = 0xFFFF
00166 } StatusCode;
00167
00168 struct Option
00169 {
00170     beuint16_t id;
00171     beuint16_t len;
00172
00173     inline Option *GetNext() { return (Option *)(((uint8_t *)this) + len + sizeof(Option)); }
00174 } __attribute__((packed));
00175
00176 typedef enum
00177 {
00178     DUID_TYPE_LLT = 1,
00179     DUID_TYPE_EN = 2,
00180     DUID_TYPE_LL = 3,
00181     DUID_TYPE_FORCE_16BIT = 0xFFFF
00182 } DUID_Types;
00183
00184 struct DUID
00185 {
00186     beuint16_t type;
00187 } __attribute__((packed));
00188
00189 struct DUID_LLT : public DUID
00190 {
00191     beuint16_t hardwareType;
00192     beuint32_t time;
00193     uint8_t llAddr[];
00194 } __attribute__((packed));
00195
00196 struct DUID_EN : public DUID
00197 {
00198     beuint32_t enterpriseNum;
00199     uint8_t ID[];
00200 } __attribute__((packed));
00201
00202 struct DUID_LL : public DUID
00203 {
00204     beuint16_t hardwareType;
00205     uint8_t llAddr[];
00206 } __attribute__((packed));
00207
00208 // RFC 3315.6 Client/Server messages, pg. 17
00209 struct DHCP_Message
```

```
00210 {
00211     beuint32_t typeAndXid;
00212     Option opts[];
00213 
00214     Option *GetOption(Option_ID optID, Option *lastOpt = NULL);
00215 } __attribute__((packed));
00216 
00217 namespace Opt
00218 {
00219 struct BaseOption : public Option
00220 {
00221     uint8_t data[];
00222 } __attribute__((packed));
00223 
00224 struct ClientServerID : public Option
00225 {
00226     DUID duid;
00227 } __attribute__((packed));
00228 
00229 struct IA_NA : public Option
00230 {
00231     beuint32_t IAID;
00232     beuint32_t T1;
00233     beuint32_t T2;
00234     Option opts[];
00235 
00236     Option *GetOption(Option_ID optID, Option *lastOpt = NULL);
00237 } __attribute__((packed));
00238 
00239 struct IA_TA : public Option
00240 {
00241     beuint32_t IAID;
00242     Option opts[];
00243 
00244     Option *GetOption(Option_ID optID, Option *lastOpt = NULL);
00245 } __attribute__((packed));
00246 
00247 struct IA_ADDR : public Option
00248 {
00249     IPADDR6 addr;
00250     beuint32_t preferred;
00251     beuint32_t valid;
00252     Option opts[];
00253 
00254     inline IA_ADDR(IPADDR6 ip, uint32_t pref = 0, uint32_t val = 0) : addr(ip), preferred(pref),
      valid(val)
00255     {
00256         id = OPT_IA_ADDR;
00257         len = sizeof(IA_ADDR) - sizeof(Option);
00258     }
00259 
00260     inline bool ServerIsInvalidating() { return (preferred == 0) && (valid == 0); }
00261 
00262     Option *GetOption(Option_ID optID, Option *lastOpt = NULL);
00263 } __attribute__((packed));
00264 
00265 struct ORO : public Option
00266 {
00267     beuint16_t ReqOptCode[];
00268 } __attribute__((packed));
00269 
00270 struct Pref : public Option
00271 {
00272     uint8_t prefVal;
00273 } __attribute__((packed));
00274 
00275 struct ElapsedTime : public Option
00276 {
00277     beuint16_t elapsed;
00278 } __attribute__((packed));
00279 
00280 struct RelayMsg : public Option
00281 {
00282     uint8_t relayMsg[];
00283 } __attribute__((packed));
00284 
00285 struct Auth : public Option
00286 {
00287     uint8_t protocol;
00288     uint8_t alg;
00289     uint8_t RDM;
00290     uint8_t replay;
00291     uint8_t authInfo[];
00292 } __attribute__((packed));
00293 
00294 struct Unicast : public Option
00295 {
```

```
00296     IPADDR6 ServerAddr;
00297 } __attribute__((packed));
00298
00299 struct StatusMsg : public Option
00300 {
00301     beuint16_t statusCode;
00302     uint8_t statusMsg[];   // UTF-8 string, NON NULL-TERMINATED!!!
00303 } __attribute__((packed));
00304
00305 struct RapidCommit : public Option
00306 {
00307     // empty
00308 } __attribute__((packed));
00309
00310 struct UserClass : public Option
00311 {
00312     struct ClassData
00313     {
00314         beuint16_t classLen;
00315         uint8_t data[];
00316     } __attribute__((packed));
00317     ClassData data[];
00318 } __attribute__((packed));
00319
00320 struct VendorClass : public Option
00321 {
00322     struct ClassData
00323     {
00324         beuint16_t classLen;
00325         uint8_t data[];
00326     } __attribute__((packed));
00327     beuint32_t enterpriseNum;
00328     ClassData data[];
00329 } __attribute__((packed));
00330
00331 struct VendorOpts : public Option
00332 {
00333     beuint32_t enterpriseNum;
00334     Option opts[];
00335 } __attribute__((packed));
00336
00337 struct InterfaceID : public Option
00338 {
00339     uint8_t interfaceID[];   // arbitrary len ID
00340 } __attribute__((packed));
00341
00342 struct ReconfMsg : public Option
00343 {
00344     beuint16_t msgType;
00345 } __attribute__((packed));
00346
00347 struct ReconfAccept : public Option
00348 {
00349     // empty
00350 } __attribute__((packed));
00351
00352 struct DNS_Servers : public Option
00353 {
00354     IPADDR6 nameServer[];
00355 } __attribute__((packed));
00356
00357 struct DomainList : public Option
00358 {
00359     uint8_t searchList[];
00360 } __attribute__((packed));
00361
00362 struct InfoRefreshTime : public Option
00363 {
00364     beuint32_t refreshTime;
00365 };
00366
00367 // FullyQualifiedDomainName RFC 4704
00368 struct FQDN : public Option
00369 {
00370     uint8_t flags;
00371     uint8_t namelen;
00372     char name[];
00373     //    inline FQDN(uint32_t namelen = 0)
00374     //        : flags(0x00)
00375     //    {id = OPT_CLIENT_FQDN; len = namelen + 1;}
00376 };
00377
00378 }   // namespace Opt
00379 void ShowOption(int lvl, Option *opt);
00380 }   // namespace DHCPv6
00381 }   // namespace V6
00382 }   // namespace NB
```

```
00383
00384 #endif /* ----- #ifndef __DHCPV6_MSG_H  ----- */
```

24.397 ipv6_addr.h File Reference

NetBurner IPADDR6 Class.

```
#include <basictypes.h>
#include <nbprintfinternal.h>
#include <nettypes.h>
```

Classes

- class IPADDR6

  Used to hold and manipulate IPv4 and IPv6 addresses in dual stack mode.

24.397.1 Detailed Description

NetBurner IPADDR6 Class.

24.398 ipv6_addr.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00023 #ifndef IPV6_ADDR_H
00024 #define IPV6_ADDR_H
00025
00026 struct IPADDR6;   // Forward
00027
00028 #include <basictypes.h>
00029 #include <nbprintfinternal.h>
00030 #include <nettypes.h>
00031
00038 class IPADDR6
00039 {
00040    protected:
00041     void Compactformat(char *cp) const;
00042     beuint32_t val[4];
00043
00044    public:
00045     // IPv6 operator = overload
00046     IPADDR6 &operator=(const IPADDR6 &v)
00047     {
00048         val[0] = v.val[0];
00049         val[1] = v.val[1];
00050         val[2] = v.val[2];
00051         val[3] = v.val[3];
00052         return *this;
00053     }
00054
00064     bool IsEmbeddedIPV4() const { return ((val[2] == 0xFFFF) && (val[0] == 0) && (val[1] == 0)); }
00065
00071     IPADDR4 Extract4() const
00072     {
00073         IPADDR4 i4;
00074         i4 = val[3];
00075         return i4;
00076     };
00077
00078     // Operator = overload
00079     IPADDR6 &operator=(const IPADDR4 v4)
00080     {
00081         val[0] = 0;
00082         val[1] = 0;
00083         val[2] = 0xFFFF;
00084         val[3] = v4;
00085         return *this;
00086     }
00087
00088     //-------------------------------------------------------------------------------------
00089     // Member functions to query IP address status
```

```
00090     //-------------------------------------------------------------------------------------------
00091
00099     bool IsNull() const { return ((val[0] | val[1] | val[3] | ((val[2] != 0xFFFF) && (val[2] != 0)))
      == 0); }
00100
00108     bool NotNull() const { return !IsNull(); }
00109
00117     inline bool IsLoopBack() const { if(IsEmbeddedIPV4())
00118                                         { return Extract4().IsLoopBack(); }
00119                                      else
00120                                         { return ((val[3] == 1) && (val[2] == 0) && (val[1] == 0) &&
      (val[0] == 0)); }
00121                                    };
00122
00130     inline bool IsMultiCast() const { return ((val[0] & 0xff000000) == 0xff000000); };
00131
00141     inline bool IsLinkLocal() const { return ((val[0] & 0xffc00000) == 0xfe800000); };
00142
00149     MACADR McastMac() const;
00150
00151     //-------------------------------------------------------------------------------------------
00152     // Member functions to display/print an IP address
00153     //-------------------------------------------------------------------------------------------
00154
00164     void print(bool bCompact = true, bool bShowV4Raw = false) const;
00165
00176     void fdprint(int fd, bool bCompact = true, bool bShowV4Raw = false) const;
00177
00191     int sprintf(char *cp, int maxl, bool bCompact = true, bool bShowV4Raw = false) const;
00192
00193     //-------------------------------------------------------------------------------------------
00194     // Static functions that return an IPADDR6 object
00195     //-------------------------------------------------------------------------------------------
00196
00207     static IPADDR6 AsciiToIp6(const char *cp, bool bembed_v4addresses = true);   // false only parses
      a v6
00208
00209     // Used internally to generate auto-configuration addresses, such as a prefix received from a
      router.
00210     static IPADDR6 FromIPMask(MACADR &ma, const IPADDR6 &g_root, int mask_len);
00211
00212     // Internal use only
00213     static IPADDR6 SolicitedNodeIP6(const IPADDR6 &ip6);
00214
00232     static IPADDR6 NullIP();
00233
00234     //-------------------------------------------------------------------------------------------
00235     // Member functions that correspond to the static functions
00236     //-------------------------------------------------------------------------------------------
00237
00246     void SetFromAscii(const char *cp, bool bembed_v4addresses = true);
00247
00248     void SetGlobal(MACADR &ma, const IPADDR6 &g_root);   // Internal use only
00249     void SetSolicitedNodeIP6(const IPADDR6 &ip6);        // Internal use only
00250
00251     // Used internally to generate auto-configuration addresses, such as a prefix received from a
      router.
00252     void SetFromIPMask(MACADR &ma, const IPADDR6 &g_root, int mask_len);
00253
00261     void SetFromIP4(IPADDR4 ip);
00262
00273     void SetFromUint32Shortcut(uint32_t w0, uint32_t w1, uint32_t w2, uint32_t w3)
00274     {
00275         val[0] = w0;
00276         val[1] = w1;
00277         val[2] = w2;
00278         val[3] = w3;
00279     }
00280
00286     void SetNull()
00287     {
00288         val[0] = 0;
00289         val[1] = 0;
00290         val[2] = 0;
00291         val[3] = 0;
00292     }
00293
00294     uint32_t csum();   // Calculate the inet pre csum for this
00295
00296     //-------------------------------------------------------------------------------------------
00297     // Constructors
00298     //-------------------------------------------------------------------------------------------
00299
00300     IPADDR6() { SetNull(); };
00301     IPADDR6(IPADDR4 ip4) { SetFromIP4(ip4); };
00302     IPADDR6(uint32_t w0, uint32_t w1, uint32_t w2, uint32_t w3) { SetFromUint32Shortcut(w0, w1, w2,
      w3); };
```

```
00303
00304     //-------------------------------------------------------------------------------------------
00305     // Internal use only, used by system printf functions
00306     //-------------------------------------------------------------------------------------------
00307
00308     int GetPrintLen(bool compact);
00309     int PrintHelper(PutCharsFunction *pf, void *data, bool compact);
00310
00311     // Access the uint32_t values
00312     beuint32_t inline GetInternalValue(int i) const { return val[i]; };
00313
00314     //-------------------------------------------------------------------------------------------
00315     // Friend functions for operator overloading
00316     //-------------------------------------------------------------------------------------------
00317
00318     friend bool operator==(const IPADDR6 i, const IPADDR6 j);
00319     friend bool operator!=(const IPADDR6 i, const IPADDR6 j);
00320     friend bool operator>(const IPADDR6 i, const IPADDR6 j);
00321     friend bool operator<(const IPADDR6 i, const IPADDR6 j);
00322
00323 } __attribute__((packed));
00324
00325 //-----------------------------------------------------------------------------------------------
00326 // friend function implementations for operator overload
00327 //-----------------------------------------------------------------------------------------------
00328
00329 inline bool operator==(const IPADDR6 i, const IPADDR6 j)
00330 {
00331     return ((i.val[3] == j.val[3]) && (i.val[2] == j.val[2]) && (i.val[1] == j.val[1]) && (i.val[0] ==
    j.val[0]));
00332 }
00333
00334 inline bool operator!=(const IPADDR6 i, const IPADDR6 j)
00335 {
00336     return !((i.val[3] == j.val[3]) && (i.val[2] == j.val[2]) && (i.val[1] == j.val[1]) && (i.val[0]
    == j.val[0]));
00337 }
00338
00339 inline bool operator>(const IPADDR6 i, const IPADDR6 j)
00340 {
00341     if (i.val[0] > j.val[0]) return true;
00342
00343     if (i.val[0] == j.val[0])
00344     {
00345         if (i.val[1] > j.val[1]) return true;
00346
00347         if (i.val[1] == j.val[1])
00348         {
00349             if (i.val[2] > j.val[2]) return true;
00350
00351             if (i.val[2] == j.val[2])
00352             {
00353                 if (i.val[3] > j.val[3]) return true;
00354             }
00355         }
00356     }
00357     return false;
00358 };
00359
00360 inline bool operator<(const IPADDR6 i, const IPADDR6 j)
00361 {
00362     if (i.val[0] < j.val[0]) return true;
00363
00364     if (i.val[0] == j.val[0])
00365     {
00366         if (i.val[1] < j.val[1]) return true;
00367         if (i.val[1] == j.val[1])
00368         {
00369             if (i.val[2] < j.val[2]) return true;
00370             if (i.val[2] == j.val[2])
00371             {
00372                 if (i.val[3] < j.val[3]) return true;
00373             }
00374         }
00375     }
00376     return false;
00377 };
00378
00379 #endif
00380
```

24.399 ipv6_constants.h

```
00001 /*NB_REVISION*/
00002
```

```
00003 /*NB_COPYRIGHT*/
00004 #ifndef NB_IPV6CONST_H
00005 #define NB_IPV6CONST_H
00006
00007 #define DEF_IPV6_ETHERNET_MTU (1500)
00008 #define DEF_IPV6_MIN_MTU (1280)
00009 #define DEF_IPV6_HOPS (64)
00010 #define DEF_IPV6_REACHABLE_TIME (30)
00011 #define IPV6_RETRANS_TIMER (1)
00012 #define IPV6_MAX_MULTICAST_SOLICIT (3)   // How many time before we give up.
00013 #define IPV6_MAX_UNICAST_SOLICIT (3)
00014 #define DELAY_FIRST_PROBE_TIME (5)
00015 #define IPV6_MLD_REFRESH_DELAY_SEC (30)
00016
00017 #define NUM_DEST_TIMEOUT_COUNT (32)   // How many destinations to keep around
00018 #define DEST_TIMEOUT_SECS (60)        // When over the limit how old does dest need to be before
      killing it
00019 #define MAX_PKTS_ON_NEIGHBOR (5)
00020 #define DAD_DELAY_TICKS (20)
00021
00022 #define IPV6_HDR_OPT_HOP_BY_HOP (0)
00023 #define IPV6_HDR_OPT_TCP (6)
00024 #define IPV6_HDR_OPT_UDP (17)
00025 #define IPV6_HDR_OPT_ENCAPV6 (41)
00026 #define IPV6_HDR_OPT_ROUTE (43)
00027 #define IPV6_HDR_OPT_FRAG (44)
00028 #define IPV6_HDR_OPT_SECU (50)
00029 #define IPV6_HDR_OPT_AUTH (51)
00030 #define IPV6_HDR_OPT_ICMP (58)
00031 #define IPV6_HDR_OPT_NONE (59)
00032 #define IPV6_HDR_OPT_DEST (60)
00033
00034 #define IPV6_OPT_TYPE_ROUTER_ALERT (5)
00035
00036 #define IPV6_ICMP_DEST_UNREACH (1)
00037 #define IPV6_ICMP_TOO_BIG (2)
00038 #define IPV6_ICMP_TIME_EXCEEDED (3)
00039 #define IPV6_ICMP_PARAM_PROB (4)
00040 #define IPV6_ICMP_ECHO_REQ (128)
00041 #define IPV6_ICMP_ECHO_REPLY (129)
00042 #define IPV6_ICMP_MC_LISTENER_QUERY (130)
00043 #define IPV6_ICMP_MC_LISTENER_REPORT (131)
00044 #define IPV6_ICMP_MC_LISTENER_DONE (132)
00045 #define IPV6_ICMP_ND_R_SOLICIT (133)
00046 #define IPV6_ICMP_ND_R_ADVERTISE (134)
00047 #define IPV6_ICMP_ND_N_SOLICIT (135)
00048 #define IPV6_ICMP_ND_N_ADVERTISE (136)
00049 #define IPV6_ICMP_ND_REDIRECT (137)
00050 #define IPV6_ICMP_I_ND_SOLICIT (141)     // Node info
00051 #define IPV6_ICMP_I_ND_ADVERTISE (142)   // Node info
00052 #define IPV6_ICMP_MCV2_LISTENER_RP (143)
00053
00054 #define IPV6_ICMP_MC_ROUTER_ADVERTISE (151)
00055 #define IPV6_ICMP_MC_ROUTER_SOLICIT (152)
00056 #define IPV6_ICMP_MC_ROUTER_TERMINATE (153)
00057
00058 #define IPV6_MCV2_REC_MODE_IS_INCLUDE (1)
00059 #define IPV6_MCV2_REC_MODE_IS_EXCLUDE (2)
00060 #define IPV6_MCV2_REC_CHANGE_INCLUDE (3)
00061 #define IPV6_MCV2_REC_CHANGE_EXCLUDE (4)
00062 #define IPV6_MCV2_REC_ALLOW_NEW_SOURCES (5)
00063 #define IPV6_MCV2_REC_BLOCK_OLD_SOURCES (6)
00064
00065 #define PARAM_ERROR_ERONIOUS_HEADER (0)
00066 #define PARAM_ERROR_UNREC_HEADER (1)
00067 #define PARAM_ERROR_UNREC_OPT (2)
00068 #endif
```

24.400 ipv6_diag.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef IPV6_DIAG_H
00006 #define IPV6_DIAG_H
00007 void ShowAllV6Info();
00008 void ShowIP6Counters();
00009 #endif
```

24.401 ipv6_frames.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef IPV6_FRAMES_H
00006 #define IPV6_FRAMES_H
00007
00008 #include <basictypes.h>
00009 #include <ipv6/ipv6_addr.h>
00010 #include <nettypes.h>
00011
00012 #define ND_OPTION_SL_ADDR (1)
00013 #define ND_OPTION_TL_ADDR (2)
00014 #define ND_OPTION_PREFIX (3)
00015 #define ND_OPTION_REDIR (4)
00016 #define ND_OPTION_MTU (5)
00017 #define ND_OPTION_AD_INTVL (7)
00018 #define ND_OPTION_HOME_AGENT (8)
00019 #define ND_OPTION_ROUTE_INFO (9)
00020 #define ND_OPTION_DNS (25)
00021
00022 #define ND_FLAG_ROUTER (0x80000000)
00023 #define ND_FLAG_SOLICITED (0x40000000)
00024 #define ND_FLAG_OVERRIDE (0x20000000)
00025
00026 #define RA_FLAG_MANAGED_IP (0x80)
00027 #define RA_FLAG_OTHER_AVAIL (0x40)
00028
00029 struct IP6FRAME
00030 {
00031     beuint16_t ver_ihl_flowmsb;
00032     beuint16_t flowlsb;
00033     beuint16_t length;
00034     uint8_t next_header;
00035     uint8_t hop_limit;
00036     IPADDR6 source_ip;
00037     IPADDR6 dest_ip;
00038     uint8_t pData[];
00039 } __attribute__((packed));
00040
00041 struct IP6_FRAG
00042 {
00043     uint8_t frag_next_header;
00044     uint8_t reserved;
00045     beuint16_t offset_and_flags;
00046     beuint32_t packetid;
00047     uint8_t pData[];
00048 } __attribute__((packed));
00049
00050 struct IP6_ROUTE_HEADER
00051 {
00052     uint8_t next_header;
00053     uint8_t ext_len;
00054     uint8_t route_type;
00055     uint8_t seg_left;
00056     uint8_t pData[];
00057 } __attribute__((packed));
00058
00059 struct ICMPV6_PSEUDO_HDR
00060 {
00061     IPADDR6 source_ip;
00062     IPADDR6 dest_ip;
00063     beuint32_t icmplength;
00064     uint8_t zero_0;
00065     uint8_t zero_1;
00066     uint8_t zero_2;
00067     uint8_t next_hdr;
00068 } __attribute__((packed));
00069
00070 struct ICMPV6_HDR
00071 {
00072     uint8_t icmp_type;
00073     uint8_t icmp_code;
00074     uint16_t checksum;   // checksum, leave as native
00075 } __attribute__((packed));
00076
00077 struct ICMPV6_ND_SOLICIT
00078 {
00079     uint8_t icmp_type;   // 135
00080     uint8_t icmp_code;   // 0
00081     uint16_t checksum;   // checksum, leave as native
00082     beuint32_t pad;
00083     IPADDR6 target_ip;
00084
00085 } __attribute__((packed));
```

```
00086
00087 struct ICMP6_ND_ADVERT_RSP
00088 {
00089     uint8_t icmp_type;    // 136
00090     uint8_t icmp_code;    // 0
00091     uint16_t checksum;    // checksum, leave as native
00092     beuint32_t flags;
00093     IPADDR6 target_ip;
00094     uint8_t option_type;   // 1
00095     uint8_t len_type;      // 1
00096     MACADR the_mac;
00097 } __attribute__((packed));
00098
00099 struct ICMP6_ND_ADVERT
00100 {
00101     uint8_t icmp_type;    // 136
00102     uint8_t icmp_code;    // 0
00103     uint16_t checksum;    // checksum, leave as native
00104     beuint32_t flags;
00105     IPADDR6 target_ip;
00106 } __attribute__((packed));
00107
00108 struct ICMPV6_PARAM_PROB
00109 {
00110     uint8_t icmp_type;    // 4
00111     uint8_t icmp_code;
00112     uint16_t checksum;    // checksum, leave as native
00113     beuint32_t pointer;
00114 } __attribute__((packed));
00115
00116 struct ICMPV6_TOOBIG
00117 {
00118     uint8_t icmp_type;    // 4
00119     uint8_t icmp_code;
00120     uint16_t checksum;    // checksum, leave as native
00121     beuint32_t mtu;
00122 } __attribute__((packed));
00123
00124 struct ICMPV6_DEST_UNREACH
00125 {
00126     uint8_t icmp_type;    // 4
00127     uint8_t icmp_code;
00128     uint16_t checksum;    // checksum, leave as native
00129     beuint32_t unused;
00130 } __attribute__((packed));
00131
00132 struct ICMP6_ERROR
00133 {
00134     uint8_t icmp_type;
00135     uint8_t icmp_code;
00136     uint16_t checksum;    // checksum, leave as native
00137     beuint32_t pointer;
00138     uint8_t pData[];
00139 } __attribute__((packed));
00140
00141 struct ICMPV6_TIME_EXCEEDED
00142 {
00143     uint8_t icmp_type;    // 4
00144     uint8_t icmp_code;
00145     uint16_t checksum;    // checksum, leave as native
00146     beuint32_t unused;
00147 } __attribute__((packed));
00148
00149 struct ICMP6_RS_PKT
00150 {
00151     uint8_t icmp_type;    // 133
00152     uint8_t icmp_code;    // 0
00153     uint16_t checksum;    // checksum, leave as native
00154     beuint32_t reserved;
00155     uint8_t pData[];   // Allow us to send link-layer option
00156 } __attribute__((packed));
00157
00158 struct ICMPV6_REDIR
00159 {
00160     uint8_t icmp_type;    // 137
00161     uint8_t icmp_code;    // 0
00162     uint16_t checksum;    // checksum, leave as native
00163     beuint32_t reserved;
00164     IPADDR6 better_target_ip;    // Better first hop
00165     IPADDR6 destination_ip;      // Whee the origional was headed
00166 } __attribute__((packed));
00167
00168 struct ICMPV6_RD_ADVERT
00169 {
00170     uint8_t icmp_type;    // 134
00171     uint8_t icmp_code;    // 0
00172     uint16_t checksum;    // checksum, leave as native
```

```
00173     uint8_t cur_hop_limit;
00174     uint8_t flags;
00175     beuint16_t Lifetime;
00176     beuint32_t ReachableTime;
00177     beuint32_t RetranstimeTime;
00178 } __attribute__((packed));
00179
00180 struct ICMPV6_RA_PREFIX
00181 {
00182     uint8_t option_type;
00183     uint8_t option_len;
00184     uint8_t prefix_len;
00185     uint8_t LA_Flags;
00186     beuint32_t ValidTime;
00187     beuint32_t PreferedLifeTime;
00188     beuint32_t Reserved;
00189     IPADDR6 prefix_ip;
00190 } __attribute__((packed));
00191
00192 struct ICMPV6_RA_MTU
00193 {
00194     uint8_t option_type;
00195     uint8_t option_len;
00196     uint16_t reserved;
00197     beuint32_t link_mtu;
00198 } __attribute__((packed));
00199
00200 struct ICMPV6_RA_DNS
00201 {
00202     uint8_t option_type;
00203     uint8_t option_len;
00204     uint16_t reserved;
00205     beuint32_t lifetime;
00206     IPADDR6 list[1];
00207 } __attribute__((packed));
00208
00209 struct ICMP6_ECHO
00210 {
00211     uint8_t icmp_type;   // 128 request 129 respond
00212     uint8_t icmp_code;   // 0
00213     uint16_t checksum;   // checksum, leave as native
00214     beuint16_t id;
00215     beuint16_t seq;
00216     uint8_t pData[];
00217 } __attribute__((packed));
00218
00219 struct ICMPV6_MCV2_REC
00220 {
00221     uint8_t rec_type;
00222     uint8_t aux_len;
00223     beuint16_t src_count;
00224     IPADDR6 mcast_addr;
00225     IPADDR6 src_addr[];
00226 } __attribute__((packed));
00227
00228 struct ICMPV6_MCV2_REP
00229 {
00230     uint8_t icmp_type;   // 143
00231     uint8_t icmp_code;   // 0
00232     uint16_t checksum;   // checksum, leave as native
00233     beuint16_t reserved;
00234     beuint16_t rec_count;
00235     ICMPV6_MCV2_REC pRec[];
00236 } __attribute__((packed));
00237
00238 struct UDPV6_HEADER
00239 {
00240     beuint16_t src_port;
00241     beuint16_t dst_port;
00242     beuint16_t length;
00243     uint16_t csum;   // checksum, leave as native
00244     uint8_t data[];
00245 } __attribute__((packed));
00246
00247 #endif
```

24.402 ipv6_interface.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef IPV6_INTERFACE_H
00006 #define IPV6_INTERFACE_H
00007
```

```
00008 #include <basictypes.h>
00009 #include <buffers.h>
00010 #include <ipv6/dhcpv6_internal.h>
00011 #include <ipv6/ipv6_addr.h>
00012 #include <ipv6/ipv6_frames.h>
00013 #include <nettimer.h>
00014 #include <nettypes.h>
00015 #include <utils.h>
00016
00017 #define ADDR_ALL_NODES_LINK_LOCAL IPADDR6::AsciiToIp6("FF02::1")
00018 #define ADDR_ALL_ROUTERS_LINK_LOCAL IPADDR6::AsciiToIp6("FF02::2")
00019 #define ADDR_ALL_NB_LINK_LOCAL IPADDR6::AsciiToIp6("FF02::4e42")
00020 #define ADDR_UNICAST_LINK_LOCAL_PREFIX IPADDR6::AsciiToIp6("FE80::")
00021 #define ADDR_MLDV2_REPORTS IPADDR6::AsciiToIp6("FF02::16")
00022
00023 #define IPV6_MULTICAST_NODE_LOCAL 0xFF01
00024 #define IPV6_MULTICAST_LINK_LOCAL 0xFF02
00025 #define IPV6_MULTICAST_REALM_LOCAL 0xFF03
00026 #define IPV6_MULTICAST_ADMIN_LOCAL 0xFF04
00027 #define IPV6_MULTICAST_SITE_LOCAL 0xFF05
00028 #define IPV6_MULTICAST_ORG_LOCAL 0xFF08
00029 #define IPV6_MULTICAST_GLOBAL 0xFF0E
00030
00031 struct IP6FRAME;
00032 struct ICMP6_ND_ADVERT;
00033
00034 struct IPv6FrameProcessingStruct
00035 {
00036     PoolPtr root_pp;
00037     IP6FRAME *pIPf;
00038     uint8_t *pData;
00039     uint16_t nDataLen;
00040     uint8_t *pLargeData;   // Used for fragments larger than 1500 bytes.
00041
00042     void free_proc_struct();
00043     void release();
00044     uint8_t WalkNextHeader(int *pError_offset = NULL);
00045     bool DestOptHasError(puint8_t pData, int *pError_offset);
00046     bool HopByHopParseHasError(puint8_t pData, int *pError_offset);
00047     bool RouterHeaderParseHasError(puint8_t pData, int *pError_offset);
00048     IPv6FrameProcessingStruct(PoolPtr pp);
00049     IPv6FrameProcessingStruct()
00050     {
00051         root_pp = NULL;
00052         pIPf = NULL;
00053         pData = NULL;
00054         nDataLen = 0;
00055         pLargeData = 0;
00056     }
00057     inline puint8_t GetDataPointer()
00058     {
00059         if (pLargeData) return pLargeData;
00060         if (root_pp) return root_pp->pData;
00061         return NULL;
00062     }
00063     inline EFRAME *GetEframe()
00064     {
00065         if (pLargeData) return (EFRAME *)pLargeData;
00066         if (root_pp) return (EFRAME *)root_pp->pData;
00067         return NULL;
00068     }
00069 };
00070
00071 // forward
00072 struct IPV6_ROOT_EL;
00073 struct IPV6_PREFIX;
00074 struct IPV6_ROUTER;
00075 struct IPV6_DNS;
00076 struct IPV6_NEIGHBOR;
00077 struct IPV6_DEST;
00078 struct IPV6_ROOT_EL_CONTAINER;
00079 class IPv6Interface;
00080 class UDPPacket;
00081 class TcpCarrierPacket;
00082 struct socket_struct;
00083 typedef socket_struct SOCKET;
00084 typedef SOCKET *PSOCKET;
00085
00086 enum IPV6_ROOT_EL_TYPE
00087 {
00088     IPV6_ROOT_TYPE_EL = 0,
00089     IPV6_ROOT_TYPE_EL_CONTAINER = 1,
00090     IPV6_ROOT_TYPE_ROUTER = 2,
00091     IPV6_ROOT_TYPE_DHCPD = 3,
00092     IPV6_ROOT_TYPE_PREFIX = 4,
00093     IPV6_ROOT_TYPE_DNS = 5,
00094     IPV6_ROOT_TYPE_NEIGHBOR = 6,
```

```
00095     IPV6_ROOT_TYPE_DEST = 7,
00096 };
00097
00098 struct IPV6_ROOT_EL
00099 {
00100     IPV6_ROOT_EL *m_pNext;
00101     IPV6_ROOT_EL *m_pPrev;
00102     IPV6_ROOT_EL_CONTAINER *m_pContainer;
00103
00104     IPADDR6 m_IPAddress;
00105
00106     virtual void ShowItem();
00107     virtual bool AgeStillValidTest();
00108     virtual void free_element();
00109     virtual ~IPV6_ROOT_EL();
00110 };
00111
00112 struct IPV6_ROOT_EL_CONTAINER
00113 {
00114     const char *m_name;
00115     IPV6_ROOT_EL *m_pHead;
00116     IPV6_ROOT_EL *m_pTail;
00117     IPv6Interface *m_pInterface;
00118     int m_nElements;
00119
00120     void Insert(IPV6_ROOT_EL *pEl);
00121     void InsertBack(IPV6_ROOT_EL *pEl);
00122     IPV6_ROOT_EL *Find(const IPADDR6 &ip, bool age);
00123     IPV6_ROOT_EL *First();
00124     IPV6_ROOT_EL_CONTAINER();
00125     void Attach(IPv6Interface &Interface, const char *name);
00126     void MoveToTop(IPV6_ROOT_EL *pEl);
00127     void Remove(IPV6_ROOT_EL *pEl);
00128
00129     void Show();
00130 };
00131
00132 struct IPV6_PREFIX;
00133
00134 struct IPV6_DHCPD : public IPV6_ROOT_EL
00135 {
00136     //    IPV6_ROOT_EL_CONTAINER addrs;
00137     uint8_t m_ServerID[CLIENT_LONG_SERVID];   // Space to contain the ServerID of the leasing server
00138     uint32_t m_IAID;
00139     uint32_t m_renewTick;
00140     uint32_t m_rebindTick;
00141
00142     void free_element();
00143     virtual bool AgeStillValidTest();
00144     virtual void ShowItem();
00145     IPV6_PREFIX *FindIA_Addr(IPV6_PREFIX *start);
00146     inline uint32_t GetDhcpRenewTime() { return m_renewTick; }
00147     inline uint32_t GetDhcpRebindTime() { return m_rebindTick; }
00148 };
00149
00150 struct IPV6_IA_ADDR : public IPV6_ROOT_EL
00151 {
00152     void free_element();
00153     virtual bool AgeStillValidTest();
00154     virtual void ShowItem();
00155 };
00156
00157 struct IPV6_ROUTER : public IPV6_ROOT_EL
00158 {
00159     uint16_t m_CheckSumCache;
00160     uint32_t m_SecsLastAdvertise;
00161     IPV6_NEIGHBOR *m_pNeighbor;
00162     uint32_t m_Life_In_Secs;
00163     uint16_t m_PathMTU;
00164     IPV6_ROUTER *GetNextValid();
00165     IPV6_ROUTER *GetNext();
00166     void CleanUpAndRemove();
00167     void RemoveFromDefault();
00168     void free_element();
00169     virtual bool AgeStillValidTest();
00170
00171     virtual void ShowItem();
00172 };
00173
00174 enum eMY_ADDR_STATE
00175 {
00176     eTenative,
00177     eValid_Preferred,
00178     eValid_Depricated,
00179     eInvalid
00180 };
00181
```

```
00182 enum ePrefixSource
00183 {
00184     eLinkLocal,
00185     eRouter,
00186     eDHCP,
00187     eStatic,
00188     eUnknown
00189 };
00190
00191 // Prefix holds both local addresses and possible router destinations....
00192 struct IPV6_PREFIX : public IPV6_ROOT_EL
00193 {
00194     eMY_ADDR_STATE m_state;
00195     uint32_t time_to_lose_prefered_in_secs_from_establish;
00196     uint32_t time_to_lose_valid_in_secs_from_establish;
00197     uint32_t time_established_in_secs;
00198     uint32_t max_valid_time_seen;   // Added to do IOL prefix cleanup if valid time less than 7200
00199     IPV6_ROUTER *pRouter;           // The router that established me...
00200     IPV6_DHCPD *pDHCPD;             // Or the DHCP server that established me...
00201     uint8_t PrefixLen;
00202     bool bValidForInterface;
00203     bool bOnLink;
00204
00205     IPV6_PREFIX *GetNext()
00206     {
00207         return (IPV6_PREFIX*)m_pNext;
00208     }
00209     IPV6_PREFIX();
00210     void CleanUpAndRemove();
00211     void free_element();
00212     bool OnLink(const IPADDR6 &ip);
00213     bool Prefered();
00214     eMY_ADDR_STATE GetState() { return m_state; };
00215     virtual void ShowItem();
00216
00217     inline ePrefixSource Source()
00218     {
00219         if (pDHCPD) return eDHCP;
00220         if (pRouter) return eRouter;
00221         if (m_IPAddress.IsLinkLocal()) return eLinkLocal;
00222         if ((bValidForInterface) && (bOnLink)) return eStatic;
00223         return eUnknown;
00224     }
00225     inline uint32_t RemainingValidTime()
00226     {
00227         uint32_t lastt = time_established_in_secs;
00228         if (pRouter) lastt = pRouter->m_SecsLastAdvertise;
00229         if (time_to_lose_valid_in_secs_from_establish == 0xffffffff) return 0xffffffff;
00230         return time_to_lose_valid_in_secs_from_establish - (Secs - lastt);
00231     };
00232     inline uint32_t GetRemainingDhcpLeaseTime() { return time_to_lose_valid_in_secs_from_establish +
      time_established_in_secs - Secs; }
00233     inline uint32_t GetRemainingDhcpLeasePreferredTime()
00234     {
00235         return time_to_lose_prefered_in_secs_from_establish + time_established_in_secs - Secs;
00236     }
00237     virtual bool AgeStillValidTest();
00238     bool AgeStillPreferred();
00239     void CheckTenative();
00240 };
00241
00242 struct IPV6_DNS : public IPV6_ROOT_EL
00243 {
00244     uint32_t m_nSecsToBeValid;
00245     IPV6_ROUTER *pRouter;
00246     IPV6_DHCPD *pDHCPD;
00247     IPV6_DNS *GetNext()
00248     {
00249         return (IPV6_DNS *)m_pNext;
00250     }
00251     virtual bool AgeStillValidTest();
00252     virtual void ShowItem();
00253     void CleanUpAndRemove();
00254     void free_element();
00255 };
00256
00257 enum eRouteOutResult
00258 {
00259     eSent,
00260     eDoingND,
00261     eNoRoute
00262 };
00263 enum eMY_NEIGHBOR_STATE
00264 {
00265     eIncomplete,
00266     eReachable,
00267     eStale,
```

```
00268     eDelay,
00269     eProbe
00270 };
00271
00272 inline bool ThisCheck(void * p) {return p!=0; }
00273
00274
00275 struct IPV6_NEIGHBOR : public IPV6_ROOT_EL
00276 {
00277     MACADR m_Macaddr;
00278     eMY_NEIGHBOR_STATE m_NeighborState;
00279     uint8_t m_SentNDCount;
00280     uint32_t m_TickTimeOfNextAction;
00281     bool m_bActiveTimer;   // Should we timer service this.
00282
00283     IPV6_ROUTER *m_pRouter;
00284     PoolPtr m_pOutBound1;
00285     PoolPtr m_pOutBound2;
00286
00287     IPV6_NEIGHBOR();
00288     IPV6_NEIGHBOR *GetNext()
00289     {
00290         if (ThisCheck(this))
00291             return (IPV6_NEIGHBOR *)m_pNext;
00292         else
00293             return NULL;
00294     }
00295
00296     void free_element();
00297     void Discard();
00298     bool ProcessAdvert(IPv6FrameProcessingStruct &p6proc, ICMP6_ND_ADVERT *pRsp);
00299     void ProcessTick();
00300     void Send_ND_Solicit(bool multicast);
00301     eRouteOutResult Send(IPv6FrameProcessingStruct &p6proc, uint16_t mtu);
00302     virtual bool AgeStillValidTest();
00303     bool StillValidToSend();
00304     void SetState(eMY_NEIGHBOR_STATE s);
00305     eMY_NEIGHBOR_STATE GetState() { return m_NeighborState; };
00306     void CoreSendPend(PoolPtr pp);
00307     bool SendPending();
00308     virtual void ShowItem();
00309 };
00310
00311 struct IPV6_DEST : public IPV6_ROOT_EL
00312 {
00313     IPV6_NEIGHBOR *m_pNeighbor;
00314     IPV6_DEST *GetNext()
00315     {
00316         if (ThisCheck(this))
00317             return (IPV6_DEST *)m_pNext;
00318         else
00319             return NULL;
00320     }
00321     uint16_t m_PathMTU;
00322     uint32_t m_nSecsLastUsed;
00323     int m_nLockCount;   // Used to keep it from being destroyed if some one is using this
00324     bool m_bStaticRoute;
00325     void free_element();
00326     virtual bool AgeStillValidTest();
00327     virtual void ShowItem();
00328 };
00329
00330 // namespace NB {
00331 // namespace V6 {
00332 // namespace DHCPv6 {
00333 // class DHCPClient;
00334 //}
00335 //}
00336 //}
00337
00338 class IPv6Interface : public TimeOutElement
00339 {
00340     int m_ifnum;
00341     MACADR m_myMac;
00342     uint32_t m_LastFragCheckSec;
00343     uint32_t m_LastRouterSolSec;
00344     uint32_t m_RouterSolCount;
00345     uint32_t m_LastMLDRepSec;
00346     bool m_solicitSendLinkLayer;
00347     NB::V6::DHCPv6::DHCPClient *m_pDhcpClient;
00348
00349     OS_CRIT NDCrit;
00350     IPV6_ROOT_EL_CONTAINER Prefixes;
00351     IPV6_ROOT_EL_CONTAINER Destinations;
00352     IPV6_ROOT_EL_CONTAINER Neighbors;
00353     IPV6_ROOT_EL_CONTAINER Routers;
00354     IPV6_ROOT_EL_CONTAINER DnsList;
```

```
00355     IPV6_ROOT_EL_CONTAINER DHCPServers;
00356
00357     volatile PoolPtr m_pp_FragmentParts;
00358     virtual void TimeElementEvent();
00359     OS_FIFO m_PingFifo;
00360
00361     IPV6_PREFIX *PrefixAlloc(const IPADDR6 &ip);
00362     IPV6_DEST *DestAlloc(const IPADDR6 &ip);
00363     IPV6_ROUTER *RouterAlloc(const IPADDR6 &ip);
00364     IPV6_NEIGHBOR *NeighborAlloc(const IPADDR6 &ip);
00365     IPV6_DNS *DnsAlloc(const IPADDR6 &ip, bool front = false);
00366     IPV6_DHCPD *DHCPDAlloc(const IPADDR6 &ip);
00367
00368     inline IPV6_PREFIX *FindPrefix(const IPADDR6 &ip, bool age = false) { return (IPV6_PREFIX
      *)Prefixes.Find(ip, age); };
00369     inline IPV6_DEST *FindDest(const IPADDR6 &ip, bool age = false) { return (IPV6_DEST
      *)Destinations.Find(ip, age); };
00370     inline IPV6_ROUTER *FindRouter(const IPADDR6 &ip, bool age = false) { return (IPV6_ROUTER
      *)Routers.Find(ip, age); };
00371     inline IPV6_DHCPD *FindDHCPD(const IPADDR6 &ip, bool age = false) { return (IPV6_DHCPD
      *)DHCPServers.Find(ip, age); };
00372     inline IPV6_NEIGHBOR *FindNeighbor(const IPADDR6 &ip, bool age = false) { return (IPV6_NEIGHBOR
      *)Neighbors.Find(ip, age); };
00373     IPV6_NEIGHBOR *FindCreateNeighbor(const IPADDR6 &ip);
00374     inline IPV6_PREFIX *FindIA_Addr(IPV6_DHCPD *dhcpd, IPV6_PREFIX *prefix = NULL)
00375     {
00376         if (prefix == NULL) { prefix = FirstPrefix(); }
00377         return dhcpd->FindIA_Addr(prefix);
00378     }
00379
00380     IPV6_DNS *FindDNS(const IPADDR6 &ip, bool age = false, IPV6_DHCPD *pDhcp = NULL);   //
      {return(IPV6_DNS *)DnsList.Find(ip,age);};
00381     IPV6_DNS *FindDNSByDHCPD(IPV6_DHCPD *pDhcp, IPV6_DNS *pEl);
00382
00383     // Interface constants....
00384     uint16_t CurHopLimit;
00385     uint32_t BaseReachableTime_Ticks;   // Base value for random reachable
00386     uint32_t ReachableTime_Ticks;       // Randomized version of base reachable
00387     uint32_t ND_RetranmistTimer_Ticks;
00388
00389     uint32_t CalcRandomReachable();   // Takes ticks and returns ticks
00390
00391     IPADDR6 RoundRobinRouterIP;
00392     eRouteOutResult RouteOut(IPv6FrameProcessingStruct &p6proc, IPV6_DEST *dest = NULL);
00393
00394     uint16_t m_MultiCastMtu;
00395     uint16_t m_DefMTU;
00396     uint8_t m_DefHopCount;
00397     volatile bool bHadLink;
00398     volatile int m_bStill_Need_To_Process_DupDiscovery_Ticks;
00399     volatile bool m_bNeighborTicks;
00400
00401     /* used to make a walkable list of all ipv6 interfaces */
00402     static IPv6Interface *gifList;
00403     IPv6Interface *m_pNext;
00404     static IPADDR6 NetBurnerMultiCast;
00405
00406   private:
00407     void StartND();
00408
00409     void SumIcmp(IPv6FrameProcessingStruct &p6proc);
00410
00411     void RootErrors(uint8_t typev, uint8_t code, uint32_t ptr, IPv6FrameProcessingStruct &p6proc);
00412
00413     void SendParameterProblem(IPv6FrameProcessingStruct &p6proc, int prob, int offset);
00414
00415     void SendTimeExceeded(PoolPtr pp);
00416
00417     void SendUnreachable(IPv6FrameProcessingStruct &p6proc);
00418
00419     void Tick();
00420
00421     bool Process_UDP(IPv6FrameProcessingStruct &p6proc);
00422
00423     bool Process_TCP(IPv6FrameProcessingStruct &p6proc);
00424
00425     bool Process_Fragment(IPv6FrameProcessingStruct &p6proc);
00426
00427     bool ProcessND_N_Solicit(IPv6FrameProcessingStruct &p6proc);
00428
00429     bool ProcessND_N_Advertise(IPv6FrameProcessingStruct &p6proc);
00430
00431     bool ProcessND_R_Advertise(IPv6FrameProcessingStruct &p6proc);
00432
00433     bool ProcessRouterOptions(IPV6_ROUTER *pRouter, IPv6FrameProcessingStruct &p6proc, int &rem,
      uint8_t *&pD);
00434
```

```
00435     bool ProcessND_Redirect(IPv6FrameProcessingStruct &p6proc);
00436
00437     bool ProcessPingRequest(IPv6FrameProcessingStruct &p6proc);
00438
00439     bool ProcessPingReply(IPv6FrameProcessingStruct &p6proc);
00440
00441     // All of the icmp error packets
00442     bool ProcessIcmpUnreach(IPv6FrameProcessingStruct &p6proc);
00443     bool ProcessTooBig(IPv6FrameProcessingStruct &p6proc);
00444     bool ProcessTimeExceeded(IPv6FrameProcessingStruct &p6proc);
00445     bool ProcessParamProb(IPv6FrameProcessingStruct &p6proc);
00446
00447     bool ProcessIcmpV6(IPv6FrameProcessingStruct &p6proc);
00448
00449     bool ProcessV6(IPv6FrameProcessingStruct &p6proc);
00450
00451     void AddDefAddress(const IPADDR6 &ip);
00452
00453     void AddPrefix(const IPADDR6 &ip, int len, int expire);
00454
00455     bool ValidateIcmpPacket(IPv6FrameProcessingStruct &p6proc);
00456
00457     void SendRouterSolicit(bool sendLinkLayer);
00458
00459     void SendDUP_Discover(const IPADDR6 &ip);
00460
00461     IPADDR6 MyLinkLocalAddress()
00462     {
00463         if (m_pMyLinkLocal) return m_pMyLinkLocal->m_IPAddress;
00464         return IPADDR6::NullIP();
00465     };
00466
00467     friend struct IPV6_PREFIX;
00468     friend struct IPV6_DEST;
00469     friend struct IPV6_ROUTER;
00470     friend struct IPV6_NEIGHBOR;
00471     friend struct IPV6_DNS;
00472     friend struct IPV6_DHCPD;
00473     friend class UDPPacket;
00474     friend class NB::V6::DHCPv6::DHCPClient;
00475     friend void RetransmitV6Packet(PoolPtr pp, PSOCKET ps);
00476     friend void TcpSendwSum6(PSOCKET ps, const IPADDR &IPto, TcpCarrierPacket &pkt, BOOL keep,
      uint32_t data_sum);
00477     friend void TcpSendwSumFrom6(const IPADDR &IPto, const IPADDR &IPfrom, TcpCarrierPacket &pkt, BOOL
      keep, uint32_t data_sum);
00478     friend void InitIPv6(int ifnum);
00479     friend int Ping6(const IPADDR6 &to, uint16_t id, uint16_t seq, uint16_t maxwaitticks, int size);
00480     friend int Ping6ViaInterface(IPADDR6 &to, uint16_t id, uint16_t seq, uint16_t wait, int interface,
      int size);
00481
00482     static void RootProcessV6(PoolPtr pp);
00483
00484     void MLDTick();
00485
00486     // Returns true if it used the p6proc, false otherwise.
00487     bool ProcessICMPExtension(IPv6FrameProcessingStruct &p6proc);
00488     void JoinMulticastGroup(const IPADDR6 &addr);
00489     void LeaveMulticastGroup(const IPADDR6 &addr);
00490     void ReadyMLDBase(IPv6FrameProcessingStruct &p6proc, const IPADDR6 &recAddr);
00491     void LinkLocalIsNowValid();
00492
00493     void ClearDHCPDInfo(IPV6_DHCPD *pDhcpd = NULL);
00494
00495   public:
00496     inline IPV6_PREFIX *FirstPrefix() { return (IPV6_PREFIX *)Prefixes.First(); };
00497     inline IPV6_DEST *FirstDest() { return (IPV6_DEST *)Destinations.First(); };
00498     inline IPV6_ROUTER *FirstRouter() { return (IPV6_ROUTER *)Routers.First(); };
00499     IPV6_ROUTER *FirstDefRouter();
00500     IPV6_DHCPD *FirstDHCPD() { return (IPV6_DHCPD *)DHCPServers.First(); };
00501     inline IPV6_NEIGHBOR *FirstNeighbor() { return (IPV6_NEIGHBOR *)Neighbors.First(); };
00502     inline IPV6_DNS *FirstDNS() { return (IPV6_DNS *)DnsList.First(); };
00503
00504     IPV6_PREFIX *m_pMyLinkLocal;
00505
00506     IPADDR6 m_StaticDNS;
00507
00508     uint16_t GetDestinationMTU(const IPADDR &ip);
00509     static IPv6Interface *GetInterfaceForDestination(const IPADDR6 &ip);
00510     static IPv6Interface *GetInterfaceForSource(const IPADDR6 &ip);
00511     static IPv6Interface *GetInterfaceN(int n);
00512     static IPv6Interface *GetFirst_IP6_Interface();
00513     IPv6Interface *GetNext_IP6_Interface();
00514
00515     IPADDR6 MySourceAddress(const IPADDR6 &ip);
00516     int GetInterfaceNumber() { return m_ifnum; };
00517
00518     IPADDR6 MyDNSAddress();
```

```
00519     inline bool HadLink() { return bHadLink; }
00520
00521     IPV6_PREFIX *AddStaticAddress(const IPADDR6 &ip, int PrefixLen);
00522     IPV6_ROUTER *AddDefaultGateway(const IPADDR6 &ip);
00523     IPV6_DNS *AddStaticDNS(const IPADDR6 &ip);
00524
00525     // The following functions return true if they have successfully found and removed the indicated
      address.
00526     bool RemoveStaticAddress(const IPADDR6 &ip);
00527     bool RemoveDefaultGateway(const IPADDR6 &ip);
00528     bool RemoveStaticDNS(const IPADDR6 &ip);
00529
00530     void SetStaticDNS(IPADDR6 dns) { m_StaticDNS = dns; };
00531     inline void StartDHCP_Solicit() { NB::V6::DHCPv6::DHCPClient::ProcessDHCPAvail(this,
      (uint8_t)RA_FLAG_MANAGED_IP); }
00532     inline void StartDHCP_InfoReq() { NB::V6::DHCPv6::DHCPClient::ProcessDHCPAvail(this,
      (uint8_t)RA_FLAG_OTHER_AVAIL); }
00533
00534     bool IsMyAddress(const IPADDR6 &ip6, bool bMustBePrefered);
00535     bool OnLink(const IPADDR6 &ip6);
00536     bool HasRoute(const IPADDR6 &ip6);
00537     void NotifyReachable(const IPADDR6 &ip6);
00538     void NotifyUnreachable(const IPADDR6 &ip);
00539
00540     IPADDR6 GetMyFirstAddress()
00541     {
00542         IPV6_PREFIX *pPrefix = FirstPrefix();
00543         if (pPrefix) return pPrefix->m_IPAddress;
00544         return IPADDR6::NullIP();
00545     };
00546
00547     IPADDR6 GetMyNextAddress(const IPADDR6 &ip)
00548     {
00549         IPV6_PREFIX *pPrefix = FindPrefix(ip);
00550         if (pPrefix)
00551         {
00552             pPrefix = pPrefix->GetNext();
00553             if (pPrefix) return pPrefix->m_IPAddress;
00554         }
00555         return IPADDR6::NullIP();
00556     };
00557
00558     IPv6Interface(int ifnum);
00559     int ping(const IPADDR6 &ip, uint16_t id, uint16_t seq, uint16_t wait, int siz);
00560     int SendMLDRegistration(const IPADDR &regAddr, bool joinNotLeave = true);
00561
00562     void ShowInfo();             // IP info for a single interface
00563     static void ShowAllInfo();   // IP infor for all interfaces
00564 };
00565
00566 // Dump counters to stdout
00567 void ShowIP6Counters();
00568
00569 #endif
```

24.403 ipv6_intf.h File Reference

NetBurner DHCPv6 API.

```
#include <basictypes.h>
#include <nettypes.h>
```

Functions

- void StartDHCPv6_Solicit (int ifnum=-1)

  Manually starts the DHCPv6 Client in Full Solicitation mode.
- void StartDHCPv6_InfoReq (int ifnum=-1)

  Manually starts the DHCPv6 Client in Full Solicitation mode.
- void StartDHCPv6 (int ifnum=-1)

  Manually starts the DHCPv6 Client in Information Request mode.
- bool AddStaticIPv6Address (const IPADDR6 &ip, int ifnum=-1)

  Add a static IPv6 address to an interface.
- bool RemoveStaticIPv6Address (const IPADDR6 &ip, int ifnum=-1)

  Add a static IPv6 address to an interface.

24.403.1 Detailed Description

NetBurner DHCPv6 API.
The DHCPv6 Library allows for full DCHPv6 negotiations as a client. It operates in one of two modes: Information Request or Full Solicitation. In the Information Request mode, it will only perform Information Requests, seeking Additional Information from DHCPv6 Servers. This includes things such as DNS Servers, NTP Servers, TFTP Boot Servers, etc, basically everything that isn't assigning an address to the interface requesting the information. In Full Solicitation, the client will perform a full address solicitation and attempt to have assigned at least one address from a DHCPv6 Server. It will also obtain the Additional Information as well.
The DHCPv6 client is automagically started whenever a Router Advertisement (RFC 4861) is detected that has either the Other or Managed bits set. It is also possible to manually launch the DHCPv6 Client by calling an API function.
Finally, as a way to allow user access and extension to the DHCPv6 negotiations, there are a set of Callback functions that will allow access to add requested options and examine responses.

24.404 ipv6_intf.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00028 #ifndef IPV6_INTF_H
00029 #define IPV6_INTF_H
00030
00031 #include <basictypes.h>
00032 #include <nettypes.h>
00033
00034 void InitIPv6(int ifc = 0);
00035 void ShowAllV6Info();
00036
00045 void StartDHCPv6_Solicit(int ifnum = -1);
00046
00053 void StartDHCPv6_InfoReq(int ifnum = -1);
00054
00063 inline void StartDHCPv6(int ifnum = -1)
00064 {
00065     StartDHCPv6_Solicit(ifnum);
00066 }
00067
00080 bool AddStaticIPv6Address(const IPADDR6 &ip, int ifnum = -1);
00081
00094 bool RemoveStaticIPv6Address(const IPADDR6 &ip, int ifnum = -1);
00095
00096 #endif
00097
```

24.405 json_lexer.h File Reference

NetBurner JSON Lexer. See the JSON Lexer page for complete documentation.

```
#include <buffers.h>
#include <math.h>
#include <nettypes.h>
#include <webclient/web_buffers.h>
#include <nbstring.h>
```

Classes

- struct JsonAllocString

  A list of large strings that are created with malloc.

- class JsonRef

  Represents a positional reference (pointer) of a location inside a ParsedJsonDataSet object

- class ParsedJsonDataSet

  A class to create, read, and modify a JSON object.

Typedefs

- typedef void() CharOutputFn(const char *chars, int len, void *blob)

  Helper function typedef for print functions.

Enumerations

- enum json_primitive_type {
  UNDEFINED , BEGIN_ARRAY , BEGIN_OBJECT , END_ARRAY ,
  END_OBJECT , NAME , STRING , VALUE_SEPERATOR ,
  NUMBER , FALSE_EL , TRUE_EL , NULL_EL ,
  STRING_TOO_BIG , ALLOC_STRING , NOTFOUND , EOF_EL }

  The following types define the basic building blocks that make up a JSON data set. These are the values that will be returned from the functions used to parse the data set. Member functions include operators to return specific data type, as well as type validity checks.

24.405.1 Detailed Description

NetBurner JSON Lexer. See the JSON Lexer page for complete documentation.

24.406 json_lexer.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00045 #ifndef _JSON_LEXER_H
00046 #define _JSON_LEXER_H
00047
00048 #include <buffers.h>
00049 #include <math.h>
00050 #include <nettypes.h>
00051 #include <webclient/web_buffers.h>
00052 #include <nbstring.h>
00053
00058 enum json_primitive_type
00059 {
00060     UNDEFINED,
00061     BEGIN_ARRAY,
00062     BEGIN_OBJECT,
00063     END_ARRAY,
00064     END_OBJECT,
00065     NAME,
00066     STRING,
00067     VALUE_SEPERATOR,
00068     NUMBER,
00069     FALSE_EL,
00070     TRUE_EL,
00071     NULL_EL,
00072     STRING_TOO_BIG,
00073     ALLOC_STRING,
00074     NOTFOUND,
00075     EOF_EL
00076 };
00077
00078 const char *GetTypeName(json_primitive_type t);
00079
00080
00081 typedef json_primitive_type json_primative_type; // Added for backwards compatibility
00082
00090 typedef void(CharOutputFn)(const char *chars, int len, void *blob);
00091
00096 struct JsonAllocString
00097 {
00098     JsonAllocString * pNext;
00099     char data[];
00100 };
00101
00102 class ParsedJsonDataSet;        // forward declaration
00103
00111 class JsonRef {
00112     const ParsedJsonDataSet *parser;
00113     PoolPtr m_pCurrent_Pool;
00114     int m_nCurrent_Position;
```

```
00115
00116     JsonRef() {parser = 0; m_pCurrent_Pool = 0; m_nCurrent_Position = 0;}
00117     friend class ParsedJsonDataSet;
00118         // Used to print objects
00119     int printhelper(CharOutputFn *pf, void *p, bool pretty);
00120
00121
00122
00123     public:
00124     JsonRef(const JsonRef &pos);
00125     json_primitive_type GetCurrent() const;
00126     json_primitive_type GetRawCurrent() const;          // Get current position element including non
      public types
00127     json_primitive_type GetNextName();
00128     json_primitive_type GetNext();
00129     json_primitive_type GetFirst();
00130     json_primitive_type GetNextNameInCurrentArray();
00131     json_primitive_type GetNextNameInCurrentObject();   // Retrieve only name elements inside current
      object
00132     json_primitive_type FindFullAtName(const char *name);
00133     json_primitive_type SkipCurrentValue();             // Skips over the current value
00134     json_primitive_type FindFullName(const char *name);
00135     json_primitive_type FindElementAfterName(const char *name);
00136     json_primitive_type FindElementAfterNameInCurrentObject(const char *name);
00137     json_primitive_type FindGlobalElementAfterName(const char *name);
00138     json_primitive_type GetIndexInCurrentArray(int i);
00139     json_primitive_type FindElementAfterNameInCurrentArray(const char *name);
00140
00141
00142     const char *CurrentName() const;
00143     const char *CurrentString() const;
00144     double CurrentNumber() const;
00145     bool CurrentBool() const;
00146     bool PermissiveCurrentBool() const;
00147
00168     JsonRef operator [](int i);
00169
00179     JsonRef object(const char *name)
00180     {
00181         JsonRef prf = *this;
00182         if (prf.Valid())
00183         {
00184             if (prf.GetCurrent() == BEGIN_OBJECT)
00185                 prf.next();
00186
00187             while (1)
00188             {
00189                 json_primitive_type t = prf.FindElementAfterNameInCurrentObject(name);
00190                 if ((t != NOTFOUND) && (t != UNDEFINED) && (t != EOF_EL))
00191                     return prf;
00192
00193                 if (t == NOTFOUND)
00194                 {
00195                     prf.MakeInvalid();
00196                     return prf;
00197                 };
00198
00199                 if (t == BEGIN_OBJECT)
00200                 {
00201                     prf.MakeInvalid();
00202                     return prf;
00203                 };
00204             }
00205         }
00206         else
00207         {
00208             return prf;
00209         }
00210     };
00211
00212
00224     JsonRef name(const char *name)
00225     {
00226         JsonRef prf = *this;
00227         if (prf.Valid())
00228         {
00229             if (prf.GetCurrent() == BEGIN_OBJECT)
00230                 prf.next();
00231
00232             if (prf.FindElementAfterNameInCurrentObject(name) == NOTFOUND)
00233                 prf.MakeInvalid();
00234         }
00235         return prf;
00236     };
00237
00249     JsonRef operator()(const char *name)
00250     {
```

```
00251         JsonRef prf = *this;
00252         if (prf.Valid())
00253         {
00254             if (prf.GetCurrent() == BEGIN_OBJECT)
00255                 prf.next();
00256 
00257             if (prf.FindElementAfterNameInCurrentObject(name) == NOTFOUND)
00258                 prf.MakeInvalid();
00259         }
00260         return prf;
00261     };
00279     operator bool ()    const {return PermissiveCurrentBool(); };
00280 
00284     operator float ()   const {return (float)CurrentNumber(); };
00285 
00289     operator double ()  const {return CurrentNumber(); };
00290 
00294     operator uint8_t()  const {return  (uint8_t)CurrentNumber(); };
00295 
00299     operator int()      const {return  (int)CurrentNumber(); };
00300 
00304     operator uint16_t() const {return (uint16_t)CurrentNumber(); };
00305 
00309     operator uint32_t() const {return (uint32_t)CurrentNumber(); };
00310 
00314     operator int8_t()   const {return   (int8_t)CurrentNumber(); };
00315 
00319     operator int16_t()  const {return (int16_t)CurrentNumber(); };
00320 
00324     operator int32_t()  const {return (uint32_t)CurrentNumber(); };
00325 
00329     operator time_t()   const {return (time_t)CurrentNumber(); };
00330 
00334     operator const char *() const {return CurrentString(); };
00335 
00339     operator NBString() const {return (NBString)CurrentString(); };
00340 
00350     inline bool IsNumber() {return (Valid() && GetCurrent() == NUMBER); };
00351 
00355     inline bool IsObject() {return (Valid() && GetCurrent() == BEGIN_OBJECT); } ;
00356 
00361     inline bool IsString() {return (Valid() &&  ((GetCurrent() == STRING)||(GetCurrent() ==
      ALLOC_STRING))); } ;
00362 
00366     inline bool IsBool()   {return (Valid() && ((GetCurrent() == TRUE_EL) || (GetCurrent() ==
      FALSE_EL))); } ;
00367 
00371     inline bool IsNull()   {return (Valid() && GetCurrent() == NULL_EL); } ;
00372 
00376     inline bool IsArray()  {return (Valid() && GetCurrent() == BEGIN_ARRAY); } ;
00377 
00381     inline bool Valid()const {return ((m_pCurrent_Pool) && (parser));};
00382 
00394     inline JsonRef start()
00395     {
00396         JsonRef prf = *this;
00397         if (prf.Valid())
00398             prf.ResetPosition();
00399         return prf;
00400     };
00401 
00406     JsonRef next() const  {JsonRef prf=*this;  if(prf.Valid()) prf.IncPosition();  return prf; };
00407 
00412     bool IncPosition();
00413 
00418     bool JumpPosition(int siz);
00419 
00424     void ResetPosition();
00425 
00430     void SkipElement();
00431 
00436     void SkipArray();
00437 
00442     void SkipObject();
00443 
00455     int PrintChildren(bool pretty = false);
00456 
00461     int PrintChildrenToFd(int fd, bool pretty = false);
00462 
00467     int PrintChildrenToBuffer(char *buffer, int maxlen, bool pretty = false);
00468 
00476     int GetChildPrintSize(bool pretty=false);
00477 
00482     void DiagDump(const char * lab);
00483 
00484 
00489     void MakeInvalid(){m_pCurrent_Pool=0; };
```

```
00490
00492 };
00493
00494
00502 class ParsedJsonDataSet : public buffer_object {
00503     PoolPtr m_pStorage_List_Head;
00504     JsonRef m_ref;
00505     JsonAllocString * pStringList;
00506     int m_nMax_Position;
00507     int m_nPool_Count;
00508     bool bEnableLargeStrings;
00509
00510     // Parser state vars used when receiving text data
00511     int m_state;           // Parser state flag
00512     unsigned char m_sep;   // Current string separator
00513     unsigned short m_tv;
00514     unsigned char *m_pPrevStringStart;
00515     int m_nStrLen;
00516     bool bNeedComma;
00517
00518     unsigned char *PutChar(unsigned char c);   // Used to process incoming text
00519     void CheckForAndFixBrokenString();
00520     void ProcessChar(unsigned char c);
00521     void ZeroNextString();
00522     void ZeroNextNumber(char c);
00523     void AddStringChar(char c);
00524     void AddNumberChar(char c);
00525     void EmitNumberBuffer();
00526     bool CheckStringEmit(json_primitive_type t);
00527     void Emit(json_primitive_type t);
00528     void ChangeString(json_primitive_type t);
00529     inline bool IncPosition() { return m_ref.IncPosition(); };
00530     inline bool JumpPosition(int siz) {return m_ref.JumpPosition(siz); };
00531     bool Built();
00532
00533     virtual void Error(const char *err);   // Error handler
00534
00535    private:
00536     void InsertName(const char *name);
00537     void InsertString(const char *s);
00538     void Insert(short i);
00539     void Insert(int i);
00540     void Insert(long i);
00541     void Insert(unsigned short i);
00542     void Insert(unsigned int i);
00543     void Insert(unsigned long i);
00544     void Insert(bool b);
00545     void Insert(double d);
00546     void Insert(const IPADDR &ip);
00547
00548
00549    public:
00550    friend class JsonRef;
00551
00552
00563     void EnableLargeStrings(bool b) {bEnableLargeStrings=b; };
00564
00573     virtual int WriteData(const unsigned char *pCopyFrom, int numBytes);
00574
00582     virtual int ReadFrom(int fd);
00583
00590     bool CopyObject(ParsedJsonDataSet & src_set);
00591
00596     void ClearObject();
00619     inline JsonRef operator [](int i){return m_ref[i]; };
00620
00632     inline JsonRef operator()(const char * name){return m_ref.name(name);}
00633
00645     inline JsonRef name(const char * name){return m_ref.name(name);}
00646
00656     inline JsonRef object(const char * name){return m_ref.object(name);}
00657
00669     inline JsonRef start(){return m_ref.start();};
00670
00675     inline JsonRef next(){return m_ref.next();};
00676
00690     inline json_primitive_type GetFirst() {return m_ref.GetFirst(); };
00691
00705    inline json_primitive_type GetNext() {return m_ref.GetNext();};
00706
00720     inline json_primitive_type GetCurrent() {return m_ref.GetCurrent();};
00721
00735     // Get current position element including non public types
00736     inline json_primitive_type GetRawCurrent() {return m_ref.GetRawCurrent(); };
00737
00738     // Retrieve only name elements
00739     // returns NOTFOUND if its not there
```

```
00740
00750     inline json_primitive_type GetNextName() {return m_ref.GetNextName(); };
00751
00761     json_primitive_type GetNextObject();
00762
00772     json_primitive_type GetNextArray();
00773
00783     inline json_primitive_type GetNextNameInCurrentObject(){return m_ref.GetNextNameInCurrentObject();
      };
00784
00794     inline json_primitive_type GetNextNameInCurrentArray() {return m_ref.GetNextNameInCurrentArray();
      };
00795
00804     json_primitive_type GetNextNumberInCurrentArray();
00805
00814     json_primitive_type GetNextStringInCurrentArray();
00815
00825     json_primitive_type GetNextBoolInCurrentArray();
00826
00836     json_primitive_type GetNextObjectInCurrentArray();
00851     inline json_primitive_type SkipCurrentValue() {return m_ref.SkipCurrentValue(); }; // Skips over
      the current value
00852
00853     // Get/Restore parser position
00863     inline void ResetPosition() {m_ref.ResetPosition(); };
00864
00870     JsonRef GetParsePosition();
00871
00879     JsonRef SetParsePosition(JsonRef pos);
00880
00944    inline json_primitive_type FindFullName(const char *name){ return m_ref.FindFullName(name); };
00945
00955     inline json_primitive_type FindFullAtName(const char *name) {return m_ref.FindFullAtName(name); };
00965    inline json_primitive_type FindElementAfterName(const char *name) {return
      m_ref.FindElementAfterName(name); };
00966
00976     inline json_primitive_type FindGlobalElementAfterName(const char *name) {return
      m_ref.FindGlobalElementAfterName(name); };
00977
00986     inline json_primitive_type FindElementAfterNameInCurrentObject(const char *name) {return
      m_ref.FindElementAfterNameInCurrentObject(name);  };
00995     inline json_primitive_type FindElementAfterNameInCurrentArray(const char *name)
00996     {return m_ref.FindElementAfterNameInCurrentArray(name);};
00997
01004     inline bool CurrentBool() {return m_ref.CurrentBool(); };
01005
01012     inline bool PermissiveCurrentBool() {return m_ref.PermissiveCurrentBool(); };
01013
01020     inline double CurrentNumber() {return m_ref.CurrentNumber(); };
01021
01028     inline const char *CurrentString() {return m_ref.CurrentString(); };
01029
01035     inline const char *CurrentName() {return m_ref.CurrentName(); };
01036
01045     bool FindFullNameBoolean(const char *name)
01046     {
01047         if (FindFullName(name) == TRUE_EL) return true;
01048         return false;
01049     };
01050
01060     bool FindGlobalBoolean(const char *name)
01061     {
01062         if (FindGlobalElementAfterName(name) == TRUE_EL) return true;
01063         return false;
01064     };
01065
01074     bool FindBoolean(const char *name)
01075     {
01076         if (FindElementAfterName(name) == TRUE_EL) return true;
01077         return false;
01078     };
01079
01088     bool FindBooleanInCurentObject(const char *name)
01089     {
01090         if (FindElementAfterNameInCurrentObject(name) == TRUE_EL) return true;
01091         return false;
01092     };
01093
01102     bool FindFullNamePermissiveBoolean(const char *name)
01103     {
01104         if (FindFullName(name) != NOTFOUND) return PermissiveCurrentBool();
01105         return false;
01106     };
01107
01117     bool FindGlobalPermissiveBoolean(const char *name)
01118     {
01119         if (FindGlobalElementAfterName(name) != NOTFOUND) return PermissiveCurrentBool();
```

```
01120         return false;
01121     };
01122
01131     bool FindPermissiveBoolean(const char *name)
01132     {
01133         if (FindElementAfterName(name) != NOTFOUND) return PermissiveCurrentBool();
01134         return false;
01135     };
01136
01145     bool FindPermissiveBooleanInCurentObject(const char *name)
01146     {
01147         if (FindElementAfterNameInCurrentObject(name) != NOTFOUND) return PermissiveCurrentBool();
01148         return false;
01149     };
01150
01151     // If item not found returns null
01159     const char *FindFullNameString(const char *name)
01160     {
01161         if (FindFullName(name) == STRING) return CurrentString();
01162         return 0;
01163     };
01164
01173     const char *FindGlobalString(const char *name)
01174     {
01175         if (FindGlobalElementAfterName(name) == STRING) return CurrentString();
01176         return 0;
01177     };
01178
01186     const char *FindString(const char *name)
01187     {
01188         if (FindElementAfterName(name) == STRING) return CurrentString();
01189         return 0;
01190     };
01191
01199     const char *FindStringInCurentObject(const char *name)
01200     {
01201         if (FindElementAfterNameInCurrentObject(name) == STRING) return CurrentString();
01202         return 0;
01203     };
01204
01205     // if item is not found returns nan
01213     double FindFullNameNumber(const char *name)
01214     {
01215         if (FindFullName(name) == NUMBER) return CurrentNumber();
01216         return nan("quiet");
01217     };
01218
01227     double FindGlobalNumber(const char *name)
01228     {
01229         if (FindGlobalElementAfterName(name) == NUMBER) return CurrentNumber();
01230         return nan("quiet");
01231     };
01232
01240     double FindNumber(const char *name)
01241     {
01242         if (FindElementAfterName(name) == NUMBER) return CurrentNumber();
01243         return nan("quiet");
01244     };
01245
01253     double FindNumberInCurentObject(const char *name)
01254     {
01255         if (FindElementAfterNameInCurrentObject(name) == NUMBER) return CurrentNumber();
01256         return nan("quiet");
01257     };
01258
01259     // If item is not found returns false
01269     bool FindGlobalObject(const char *name)
01270     {
01271         if (FindGlobalElementAfterName(name) == BEGIN_OBJECT) return true;
01272         return false;
01273     };
01274
01283     bool FindObject(const char *name)
01284     {
01285         if (FindElementAfterName(name) == BEGIN_OBJECT) return true;
01286         return false;
01287     };
01288
01297     bool FindObjectInCurentObject(const char *name)
01298     {
01299         if (FindElementAfterNameInCurrentObject(name) == BEGIN_OBJECT) return true;
01300         return false;
01301     };
01305     // Constructors
01306     // Build emty
01307     ParsedJsonDataSet();
01308     ParsedJsonDataSet(const char *pData, int len = 0, bool bBigStrings=false);   // 0 len mean use
```

```
      strlen
01309     // Destructor
01310     virtual ~ParsedJsonDataSet(); //Needs to be virtual for derived classes....
01311
01321     void DumpState();
01322
01328     inline int PrintObject(bool pretty = false){return m_ref.start().PrintChildren(pretty);}
01329
01339     inline int PrintObjectToBuffer(char *buffer, int maxlen, bool pretty = false) {return
      m_ref.start().PrintChildrenToBuffer(buffer,maxlen,pretty);}
01340
01348     inline int PrintObjectToFd(int fd, bool pretty = false){return
      m_ref.start().PrintChildrenToFd(fd,pretty);};
01349
01354     inline int PrintChildren(bool pretty = false) {return m_ref.PrintChildren(pretty); }
01355
01360     int PrintChildrenToFd(int fd, bool pretty = false) {return m_ref.PrintChildrenToFd(fd,pretty); }
01361
01366     int PrintChildrenToBuffer(char *buffer, int maxlen, bool pretty = false){return
      m_ref.PrintChildrenToBuffer(buffer,maxlen,pretty); }
01367
01375     inline int GetPrintSize(bool pretty = false){return m_ref.start().GetChildPrintSize(pretty);};
01387     void StartBuilding();
01388
01394     void AddObjectStart(const char *name);
01395
01401     void AddMyMac(const char *name);
01402
01409     void Add(const char *name, int i);
01410
01417     void Add(const char *name, short i);
01418
01425     void Add(const char *name, long i);
01426
01433     void Add(const char *name, unsigned int i);
01434
01441     void Add(const char *name, unsigned short i);
01442
01449     void Add(const char *name, unsigned long i);
01450
01457     void Add(const char *name, double d);
01458
01465     void Add(const char *name, const char *str);
01466
01473     void Add(const char *name, bool b);
01474
01481     void Add(const char *name, IPADDR4 i4);
01482
01489     void Add(const char *name, const IPADDR &i);
01490
01497     void Add(const char *name, const MACADR &ma);
01498
01504     void AddNull(const char *name);
01505
01511     void AddArrayStart(const char *name);
01512
01517     void EndArray();
01518
01524     void AddArrayElement(int i);
01525
01531     void AddArrayElement(short i);
01532
01538     void AddArrayElement(long i);
01539
01545     void AddArrayElement(unsigned int i);
01546
01552     void AddArrayElement(unsigned short i);
01553
01559     void AddArrayElement(unsigned long i);
01560
01566     void AddArrayElement(double d);
01567
01573     void AddArrayElement(const char *str);
01574
01580     void AddArrayElement(bool b);
01581
01587     void AddArrayElement(const IPADDR &i);
01588
01593     void AddArrayElementArray();
01594
01599     void AddArrayObjectStart();
01600
01605     void AddNullArrayElement();
01606
01611     void EndObject();
01612
01617     void DoneBuilding();
```

```
01619 };
01620
01629 #endif
01630
```

24.407 lldp.h

```
00001 #ifndef _LLDP_H
00002 #define _LLDP_H 1
00003
00004 #include <netinterface.h>
00005 #include <nettypes.h>
00006 #include <config_obj.h>
00007
00008 #define TLV_CHASSIS_ID 1
00009 #define TLV_PORT_ID 2
00010 #define TLV_TTL 3
00011 #define TLV_PORT_DESC 4
00012 #define TLV_SYS_NAME 5
00013 #define TLV_SYS_DESC 6
00014 #define TLV_SYS_CAP 7
00015 #define TLV_MANAGMENT_ADDR 8
00016 #define TLV_CUSTOM 127
00017
00018 class LLDPEntity : public TimeOutElement, public config_obj
00019 {
00020    private:
00021     int m_SetTxTime;
00022     PoolPtr m_ActivePacket;        // The packet being sent
00023     PoolPtr m_NewPacket;           // The packet being built
00024     InterfaceBlock *m_pIfBlock;    // Poitner to interface block
00025     OS_CRIT m_Entity_Crit;         // Critical section for swapping blocks.
00026     bool m_bFastTx;                // Indicates packet changed should send soon...
00027
00028     virtual void TimeElementEvent();
00029     void RawAdd(uint8_t id, unsigned int len, uint8_t *pData);
00030     void RawAddString(uint8_t id, const char *str);
00031     void RawAddMac(uint8_t id, uint8_t sub_type, MACADR &ma);
00032     void AddMandatoryHeader();    // TLV 1,2,3
00033    protected:
00034     void StartNewPacket();
00035     void AddPortDescription(const char *Description) { RawAddString(TLV_PORT_DESC, Description); };
00036     void AddHostName(const char *HostName) { RawAddString(TLV_SYS_NAME, HostName); };
00037     void AddSysDescription(const char *Description) { RawAddString(TLV_SYS_DESC, Description); };
00038     void AddSysCapabilities(uint16_t capablities, uint16_t enabled);
00039     void AddManagmentAddr(IPADDR4 ipa);
00040     void AddCustomRaw(uint32_t UUID, uint32_t org_sub, uint32_t datalen, puint8_t data);
00041     void AddCustomInt(uint32_t UUID, uint32_t org_sub, int data, uint32_t intlen);
00042     void AddCustomString(uint32_t UUID, uint32_t org_sub, const char *str);
00043     void UseNewPacket();    // automatically adds End LLDPU TLV
00044
00045    public:
00046     config_bool m_bEnable{TRUE, "Enable", "Enable/disable LLDP transmisions"};
00047     config_int m_iTxTime{30, "TxTimer", "Transmission interval in seconds"};
00048     config_int m_iHoldTime{120, "HoldTime", "LLD TTL"};
00049     config_int m_iRetransmitTime{2, "RetransmitTime", "Seconds to wait on Startup"};
00050     ConfigEndMarker;    // No new data members below this line
00051
00052     LLDPEntity(InterfaceBlock &ib);
00053     virtual void BuildPacket();
00054     void RebuildPacket() { BuildPacket(); };
00055 };
00056
00057 #endif
```

24.408 logme.h

```
00001 #ifndef __LOGME_H
00002 #define __LOGME_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <predef.h>
00007 #include <constants.h>
00008 #include <stdio.h>
00009
00010 #define _LOGME_STR(x) #x
00011 #define LOGME iprintf("L: %d - F: %s\n", __LINE__, __FILE__);
00012 #define LOGME_INT(x) iprintf("L: %d - F: %s - %s: %ld\n", __LINE__, __FILE__, _LOGME_STR(x),
      (int32_t)(x));
00013 #define LOGME_HEX(x) iprintf("L: %d - F: %s - %s: %#08lx\n", __LINE__, __FILE__, _LOGME_STR(x),
      (uint32_t)(x));
```

```
00014 #define LOGME_PTR(x) iprintf("L: %d - F: %s - %s: %p\n", __LINE__, __FILE__, _LOGME_STR(x), (void
      *)(x));
00015
00016 #endif /* ----- #ifndef __LOGME_H  ----- */
```

24.409 mailto.h File Reference

Send Emails with SMTP.

```
#include <predef.h>
#include <nettypes.h>
```

Macros

- #define **STATUS_OK** (0)

  OK, no errors.
- #define **CONNECT_TO_SMTP_SERVER_FAILED** (-1)

  Could not connect to SMTP server.
- #define **INITIAL_SERVER_REPLY_FAILED** (-2)

  Initial server reply failed.
- #define **HELO_SERVER_REPLY_FAILED** (-3)

  Server HELO reply failed.
- #define **MAIL_FROM_SERVER_REPLY_FAILED** (-4)

  Mail From server reply failed.
- #define **RCPT_TO_SERVER_REPLY_FAILED** (-5)

  Receipt To server reply failed.
- #define **DATA_SERVER_REPLY_FAILED** (-6)

  Data server reply failed.
- #define **DATA_END_SERVER_REPLY_FAILED** (-7)

  Date end server reply failed.
- #define **AUTH_LOGIN_SERVER_REPLY_FAILED** (-8)

  AUTH login server reply failed.
- #define **USER_ID_SERVER_REPLY_FAILED** (-9)

  User ID server reply failed.
- #define **PASSWORD_SERVER_REPLY_FAILED** (-10)

  Password server reply failed.
- #define **CONNECT931_SMTP_SERVER_FAILED** (-11)

  SMTP connection failed.

Functions

- int SendMail (IPADDR smtp_server, PCSTR userid, PCSTR from_addr, PCSTR to_addr, PCSTR subject, PCSTR textbody)

  Send an email message. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message.
- int SendMailAuth (IPADDR smtp_server, PCSTR userid, PCSTR pass, PCSTR from_addr, PCSTR to_addr, PCSTR subject, PCSTR textbody)

  Send an email message with plain text authentication. The function will open a TCP connection to the specified SMTP server, create a message based on the parameters, and send the message.
- int SendMailEx (IPADDR smtp_server, PCSTR userid, PCSTR from_addr_rev_path, PCSTR from_addr_↩ memo_hdr, PCSTR to_addr, PCSTR subject, PCSTR textbody)

  Send an email message function, extended version.
- int IsMailError ()

  Returns the error status of the last send mail transaction.

- void PrintNBError (int fd=0)

  If an error occurred, prints the error information received from the SMTP server.
- void PrintServerLog (int fd=0)

  Prints the server log of the last send mail transaction.
- int SendMailAuthStartMIME (IPADDR smtp_server, PCSTR userid, PCSTR pass, PCSTR from_addr, PCSTR to_addr, PCSTR subject, int &fd)

  Start a Multi-purpose Internet Mail Extension (MIME)session.
- int SendMailAuthAddMIME (int fd, int ContentType, const char *pContent, const char *FileName)

  Add a MIME part or attachment to an open MIME Session.
- int SendMailAuthEndMIME (int fd, PCSTR userid)

  Send a MIME email message and close the SMTP session.

- enum CONTENT_TYPE_ENUM {
  CONTENT_TYPE_PLAIN_TEXT , CONTENT_TYPE_PLAIN_TEXT_ATTACH , CONTENT_TYPE_BINARY_ATTACH , CONTENT_TYPE_HTML_DECOMP ,
  CONTENT_TYPE_END }

  SMTP MIME Conetnet Types.

24.409.1 Detailed Description

Send Emails with SMTP.

24.410 mailto.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00017 #ifndef _NB_MAILTO
00018 #define _NB_MAILTO
00019
00020 #include <predef.h>
00021 #include <nettypes.h>
00026 #define STATUS_OK (0)
00027 #define CONNECT_TO_SMTP_SERVER_FAILED (-1)
00028 #define INITIAL_SERVER_REPLY_FAILED (-2)
00029 #define HELO_SERVER_REPLY_FAILED (-3)
00030 #define MAIL_FROM_SERVER_REPLY_FAILED (-4)
00031 #define RCPT_TO_SERVER_REPLY_FAILED (-5)
00032 #define DATA_SERVER_REPLY_FAILED (-6)
00033 #define DATA_END_SERVER_REPLY_FAILED (-7)
00034 #define AUTH_LOGIN_SERVER_REPLY_FAILED (-8)
00035 #define USER_ID_SERVER_REPLY_FAILED (-9)
00036 #define PASSWORD_SERVER_REPLY_FAILED (-10)
00037 #define CONNECT931_SMTP_SERVER_FAILED (-11)
00040 /*-------------------------------------------------------------------
00041  * Send an email
00042  * Returns 0 on failure, 1 on success
00043  *------------------------------------------------------------------- */
00060 int SendMail(IPADDR smtp_server, /* IP address of the SMTP server */
00061              PCSTR userid,       /* ASCII String to provide for RFC931 IDentification */
00062              PCSTR from_addr,    /* From E-Mail address */
00063              PCSTR to_addr,      /* To E-Mail Address */
00064              PCSTR subject,      /* E-Mail subject */
00065              PCSTR textbody /* E-Mail body */);
00066
00067 /*-------------------------------------------------------------------
00068  * Send an email with plain text authentication
00069  * Returns 0 on failure, 1 on success
00070  *-------------------------------------------------------------------*/
00094 int SendMailAuth(IPADDR smtp_server, /*IP address of the SMTP server */
00095                  PCSTR userid,       /* ASCII String to provide for RFC931 IDentification */
00096                  PCSTR pass,         /* ASCII String to provide for AUTH IDentification */
00097                  PCSTR from_addr,    /* From E-Mail address */
00098                  PCSTR to_addr,      /* To E-Mail Address */
00099                  PCSTR subject,      /* E-Mail subject */
00100                  PCSTR textbody /* E-Mail body */);
00101
00102 /*-------------------------------------------------------------------
00103  * Similar to SendMail() with the following additions:
00104  * 1. The from_addr_rev_path is used to include the reverse source
```

```
00105  *    route per RFC 821 reverse-path option.
00106  * 2. The email contains the memo header. The mail data includes
00107  *    the memo header items such at DATE, Subject, TO, CC and From.
00108  *-----------------------------------------------------------------*/
00130 int SendMailEx(IPADDR smtp_server,
00131                PCSTR userid,
00132                PCSTR from_addr_rev_path,
00133                PCSTR from_addr_memo_hdr,
00134                PCSTR to_addr,
00135                PCSTR subject,
00136                PCSTR textbody);
00137
00138 /*---------------------------------------------------------------------
00139  * Send an email as if the NetBurner device was a SMTP server, rather
00140  * than the standard way of sending an email through an external
00141  * SMTP server through a user account.
00142  * --------------------------------------------------------------------*/
00143 int SendMailAsServer(PCSTR from_addr, PCSTR to_addr, PCSTR subject, PCSTR textbody);
00144
00145 extern uint16_t SMTP_PORT;
00146 extern uint16_t SMTP_AUTH_PORT;
00147 extern uint16_t RFC931_PORT;
00148 extern uint16_t LOCAL_MAIL_PORT;
00149
00150 /*---------------------------------------------------------------------
00151  * The following functions, variables and definitions are used for
00152  * error reporting of the mail system.
00153  * --------------------------------------------------------------------*/
00154
00155 // Returns 0 or error code
00164 int IsMailError();
00165
00178 void PrintNBError(int fd = 0);
00179
00193 void PrintServerLog(int fd = 0);
00194
00195 // Returns 0 or error code
00196 extern int NB_Mail_Error_Code;
00197
00198 // Last error string reported by NetBurner mail library. This is usually
00199 // displayed on the debug serial port.
00200 extern char NB_Mail_Error_String[];
00201
00202 // Last error string received from mail server
00203 extern char Server_Mail_Log_String[];
00204
00210 extern enum CONTENT_TYPE_ENUM {
00211     CONTENT_TYPE_PLAIN_TEXT,
00212     CONTENT_TYPE_PLAIN_TEXT_ATTACH,
00213     CONTENT_TYPE_BINARY_ATTACH,
00214     CONTENT_TYPE_HTML_DECOMP,
00215     // Add additional types above CONTENT_TYPE_END
00216     CONTENT_TYPE_END
00217 } CONTENT_TYPE;
00256 int SendMailAuthStartMIME(IPADDR smtp_server, PCSTR userid, PCSTR pass, PCSTR from_addr, PCSTR
      to_addr, PCSTR subject, int &fd);
00257
00275 int SendMailAuthAddMIME(int fd, int ContentType, const char *pContent, const char *FileName);
00276
00290 int SendMailAuthEndMIME(int fd, PCSTR userid);
00291
00292 void MIME_SendMultipartHeader(int fd);
00293
00294 #endif
00295
00296 #ifdef _NB_SSL_MAILTO
00297 /*---------------------------------------------------------------------
00298  * Function to look for matching server return codes
00299  * --------------------------------------------------------------------*/
00300 int SMPMatch(int fd, int fd931, PCSTR userid, PCSTR match, uint32_t timeout);
00301 int writeb64string(int fd, const char *cp);
00302 void SaveToMailLog(const char *buffer, int rv);
00303 extern uint16_t Server_String_Count;
00304 #endif
00305
```

24.411 md5.h

```
00001 /*NB_REVISION*/
00002
00003 #ifndef _MD5_H_
00004 #define _MD5_H_
00005
00006 /* MD5.H - header file for MD5C.C
00007  */
```

```
00008
00009 /* Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
00010 rights reserved.
00011
00012 License to copy and use this software is granted provided that it
00013 is identified as the "RSA Data Security, Inc. MD5 Message-Digest
00014 Algorithm" in all material mentioning or referencing this software
00015 or this function.
00016
00017 License is also granted to make and use derivative works provided
00018 that such works are identified as "derived from the RSA Data
00019 Security, Inc. MD5 Message-Digest Algorithm" in all material
00020 mentioning or referencing the derived work.
00021
00022 RSA Data Security, Inc. makes no representations concerning either
00023 the merchantability of this software or the suitability of this
00024 software for any particular purpose. It is provided "as is"
00025 without express or implied warranty of any kind.
00026
00027 These notices must be retained in any copies of any part of this
00028 documentation and/or software.
00029  */
00030
00031 /* MD5 context. */
00032
00033 // NB Definitions
00034 #include <predef.h>
00035
00036 // NB Libs
00037 #include <basictypes.h>
00038 #include <hash.h>
00039
00040 struct MD5_CTX : public HASH_CTX
00041 {
00042     MD5_CTX() = default;
00043     MD5_CTX(uint32_t _state[4],
00044             uint32_t _count[2],
00045             unsigned char _buffer[64]
00046 #ifdef SSL_TLS_SUPPORT
00047             ,
00048             unsigned char inner[64],
00049             unsigned char outer[64]
00050 #endif
00051     );
00052
00053     uint32_t state[4];        /* state (ABCD) */
00054     uint32_t count[2];        /* number of bits, modulo 2^64 (lsb first) */
00055     unsigned char buffer[64]; /* input buffer */
00056 #ifdef SSL_TLS_SUPPORT
00057     unsigned char hmac_inner_pad[64];
00058     unsigned char hmac_outer_pad[64];
00059 #endif
00060     void __Init();
00061     void __Update(const unsigned char *data, unsigned int len);
00062     void __Final(unsigned char *);
00063     int __GetDigestLen() const;
00064     int __GetOIDLen() const;
00065     const unsigned char *__GetOID() const;
00066     void ctor();
00067
00068   private:
00069     static __vtable_HASH_CTX_t _s__vtable;
00070 };
00071
00072 typedef unsigned char md5_digest_t[16];
00073
00074 inline void MD5Init(MD5_CTX *ctx)
00075 {
00076     ctx->ctor();
00077 }
00078 inline void MD5Update(MD5_CTX *ctx, const unsigned char *data, unsigned int len)
00079 {
00080     ctx->__Update(data, len);
00081 }
00082 inline void MD5Final(unsigned char digest[16], MD5_CTX *ctx)
00083 {
00084     ctx->__Final(digest);
00085 }
00086
00087 #endif /* #ifndef _MD5_H_ */
```

24.412 mDNS.h

```
00001
00002 /*NB_REVISION*/
```

```
00003
00004 /*NB_COPYRIGHT*/
00005
00006 #ifndef MDNS_SERVLET
00007 #define MDNS_SERVLET
00008
00009 #include <nbstring.h>
00010 #include <servlets.h>
00011 class InterfaceBlock;
00012
00013 class mDNS_servlet : public servlet
00014 {
00015     int m_fd;
00016
00017     virtual int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00018     virtual void ProcessSelectResult(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00019
00020    public:
00021     mDNS_servlet();
00022 static bool Started();
00023 };
00024
00025 extern mDNS_servlet mDNSServer;
00026
00027 #endif
```

24.413 config_mqtt_obj.h

```
00001 #ifndef  __CONFIG_MQTT_OBJ_H
00002 #define  __CONFIG_MQTT_OBJ_H
00003
00004 #include <predef.h>
00005 #include <config_obj.h>
00006 #include <nbstring.h>
00007
00008 class config_MqttLWT : public config_obj
00009 {
00010     public:
00011     config_string topic{"", "Topic", "Topic to publish Last Will & Testament message to"};
00012     config_string message{"", "Message", "LWT Message to publish on non-normal disconnect"};
00013     config_uint   sessExpIntvl{60, "SessExpIntvl", "Session Expirary delay (in seconds) between
      non-normal disconnect and broker invalidating session and publishing LWT"};
00014     config_bool   retain{true, "Retain", "Whether the broker will retain LWT to publish to future
      subscribers."};
00015     config_string connMessage{"", "Connect", "LWT Message to publish on Connect"};
00016     config_string normalDisconMessage{"", "NormalDisconnect", "Message to publish on normal
      disconnect"};
00017     ConfigEndMarker;
00018
00019     config_MqttLWT(config_obj &owner, const char* name, const char *desc = NULL)
00020         : config_obj(owner, name, desc)
00021     {
00022     }
00023     config_MqttLWT(const char* name, const char *desc = NULL)
00024         : config_obj(name, desc)
00025     {
00026     }
00027
00028     bool valid();
00029 };
00030
00031 class config_MqttClient : public config_obj
00032 {
00033     public:
00034     config_string brokerFqdn{"", "Hostname", ""};
00035     config_uint   brokerPort{MQTT_DEFAULT_PORT, "Port", ""};
00036     config_string cfgClientId{"", "ClientId", ""};
00037     config_string cfgUsername{"", "Username", ""};
00038     config_pass   cfgPassword{"", "Password", ""};
00039     config_bool   alwaysClean{true, "AlwayCleanStart", "Always start session with a clean slate of
      subscriptions."};
00040     config_bool   enableLWT{true, "LWT_Enabled", "Enable client use of LWT message"};
00041     config_uint   maxKeepAlive{MQTT_DEFAULT_KEEP_ALIVE, "KeepAlive", "Maximum desired keep alive
      interval (in seconds) (Note: the server may impose a different interval)"};
00042     ConfigEndMarker;
00043
00044     NBString interpBroker;
00045
00046     config_MqttClient(config_obj &owner, const char* name, const char *desc = NULL)
00047         : config_obj(owner, name, desc)
00048     {
00049     }
00050     config_MqttClient(const char* name, const char *desc = NULL)
00051         : config_obj(name, desc)
00052     {
```

```
00053     }
00054 };
00055
00056
00057 #endif   /* ----- #ifndef __CONFIG_MQTT_OBJ_H  ----- */
```

24.414 mqtt.h File Reference

NetBurner MQTT Library - MQTT protocol types.

```
#include <predef.h>
#include <constants.h>
#include <config_obj.h>
#include <nettimer.h>
#include <mqtt/msg_types.h>
#include <mqtt/mqtt_msg_reqs.h>
#include <webclient/http_funcs.h>
#include <mqtt/mqtt_internal.h>
#include <mqtt/config_mqtt_obj.h>
```

Functions

- void UserMain (void *pd)

  This example will provide a menu through the serial debug port that enables you to set/clear static IP settings, and start/stop the DHCP Client service.

24.414.1 Detailed Description

NetBurner MQTT Library - MQTT protocol types.

24.415 mqtt.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef  __MQTT_API_H
00006 #define  __MQTT_API_H
00007
00008 #include <predef.h>
00009 #include <constants.h>
00010 #include <config_obj.h>
00011 #include <nettimer.h>
00012 #include <mqtt/msg_types.h>
00013 #include <mqtt/mqtt_msg_reqs.h>
00014 #include <webclient/http_funcs.h>
00015
00027 #define MQTT_TASK_PRIO (MAIN_PRIO-1)
00028 #define MQTT_WAIT (TICKS_PER_SECOND*10)
00029
00030 #define MQTT_OBJ_PRIORITY_COUNT (16)
00031 #define MQTT_OBJ_PRIORITY_MAX 0
00032 #define MQTT_OBJ_PRIORITY_MIN 15
00033
00034 #define MQTT_MAX_PKT_SIZE (0x2FFFFFFF)
00035 #define MQTT_MAX_MAX_SUPPORTED_IN_FLIGHT (256)
00036 #define MQTT_MAX_SUPPORTED_IN_FLIGHT (16)
00037 #define MQTT_MAX_HANDLERS_PER_CONNECTION (256)
00038 #define MQTT_MAX_SUPPORTED_RX_TOPIC_ALIASES (128)
00039 #define MQTT_DEFAULT_SESSION_EXPIRY_INTERVAL (3600)
00040 #define MQTT_DEFAULT_KEEP_ALIVE (30)
00041
00042 #define MQTT_MAX_SUB_IDs     (128)
00043
00044 void UserMain(void*pd);
00045
00046 namespace MQTT {
00047 enum eResult_t {
00048     eResult_Success                     = 0,
00049
```

```
00050     eResult_Mqtt_Unspecified             = -Pkt::eReason_Error_Unspecified,
00051     eResult_Mqtt_MalformedPkt            = -Pkt::eReason_Error_MalformedPkt,
00052     eResult_Mqtt_Protocol                = -Pkt::eReason_Error_Protocol,
00053     eResult_Mqtt_Implementation          = -Pkt::eReason_Error_Implementation,
00054     eResult_Mqtt_UnsupportedProtoVer     = -Pkt::eReason_Error_UnsupportedProtoVer,
00055     eResult_Mqtt_ClientIdInvalid         = -Pkt::eReason_Error_ClientIdInvalid,
00056     eResult_Mqtt_UserPassword            = -Pkt::eReason_Error_UserPassword,
00057     eResult_Mqtt_NotAuthorized           = -Pkt::eReason_Error_NotAuthorized,
00058     eResult_Mqtt_ServerUnavail           = -Pkt::eReason_Error_ServerUnavail,
00059     eResult_Mqtt_ServerBusy              = -Pkt::eReason_Error_ServerBusy,
00060     eResult_Mqtt_Banned                  = -Pkt::eReason_Error_Banned,
00061     eResult_Mqtt_ServerShutdown          = -Pkt::eReason_Error_ServerShutdown,
00062     eResult_Mqtt_BadAuthMethod           = -Pkt::eReason_Error_BadAuthMethod,
00063     eResult_Mqtt_KATimeout               = -Pkt::eReason_Error_KATimeout,
00064     eResult_Mqtt_SessTakeover            = -Pkt::eReason_Error_SessTakeover,
00065     eResult_Mqtt_FilterInvalid           = -Pkt::eReason_Error_FilterInvalid,
00066     eResult_Mqtt_NameInvalid             = -Pkt::eReason_Error_NameInvalid,
00067     eResult_Mqtt_PktId_InUse             = -Pkt::eReason_Error_PktId_InUse,
00068     eResult_Mqtt_PktId_NotFound          = -Pkt::eReason_Error_PktId_NotFound,
00069     eResult_Mqtt_RcvMax_Exceeded         = -Pkt::eReason_Error_RcvMax_Exceeded,
00070     eResult_Mqtt_TopicAlias_Invalid      = -Pkt::eReason_Error_TopicAlias_Invalid,
00071     eResult_Mqtt_TooLarge                = -Pkt::eReason_Error_TooLarge,
00072     eResult_Mqtt_MsgRate                 = -Pkt::eReason_Error_MsgRate,
00073     eResult_Mqtt_Quota                   = -Pkt::eReason_Error_Quota,
00074     eResult_Mqtt_Admin                   = -Pkt::eReason_Error_Admin,
00075     eResult_Mqtt_PayFmt_Invalid          = -Pkt::eReason_Error_PayFmt_Invalid,
00076     eResult_Mqtt_NotSupported_Retain     = -Pkt::eReason_Error_NotSupported_Retain,
00077     eResult_Mqtt_NotSupported_QoS        = -Pkt::eReason_Error_NotSupported_QoS,
00078     eResult_Mqtt_UseOther                = -Pkt::eReason_Error_UseOther,
00079     eResult_Mqtt_ServerMoved             = -Pkt::eReason_Error_ServerMoved,
00080     eResult_Mqtt_NotSupported_SharedSubs = -Pkt::eReason_Error_NotSupported_SharedSubs,
00081     eResult_Mqtt_ConnRate                = -Pkt::eReason_Error_ConnRate,
00082     eResult_Mqtt_MaxConnTime             = -Pkt::eReason_Error_MaxConnTime,
00083     eResult_Mqtt_NotSupported_SubIds     = -Pkt::eReason_Error_NotSupported_SubIds,
00084     eResult_Mqtt_NotSupported_WildSubs   = -Pkt::eReason_Error_NotSupported_WildSubs,
00085     eResult_Mqtt_Transport_Error         = -Pkt::eReason_Error_Transport_Error,
00086     eResult_Error_NetErr                 = -Pkt::eReason_Error_Transport_Error,
00087     eResult_Socket_TryAgain              = -256,
00088     eResult_Socket_NeedDNS               = -257,
00089     eResult_Cont_Auth                    = -258,
00090     eResult_Error_InvalidURI             = -259,
00091     eResult_Error_NoneAvail              = -260,
00092     eResult_Error_Malformed              = -261,
00093     eResult_Error_Timeout                = -262,
00094     eResult_Error_BadProp                = -263,
00095     eResult_Error_NoSuchConn             = -264,
00096     eResult_Error_BadArg                 = -265,
00097 };
00098
00099 typedef int (*MsgCallback_t)(Client *mConn, int mqttFd, PacketInfo::Msg *msg, void *ctx);
00100
00101 struct TopicHandler {
00102     NBString topicFilter;
00103     MsgCallback_t handleMsg;
00104     void *ctx;
00105 };
00106
00107
00108 }
00109
00110 #include <mqtt/mqtt_internal.h>
00111 #include <mqtt/config_mqtt_obj.h>
00112
00113 namespace MQTT {
00114
00115 enum {
00116     eObj_Flag_PubOnWrite          = 0x0001,
00117     eObj_Flag_PubQoS_AtLeastOnce  = 0x0002,
00118     eObj_Flag_PubQoS_ExactlyOnce  = 0x0004,
00119     eObj_Flag_PubRetain           = 0x0008,
00120     eObj_Flag_Subscribe           = 0x0010,
00121     eObj_Flag_Sub_HandlerOnly     = 0x0020,
00122     eObj_Flag_Sub_IncludeLocal    = 0x0040,
00123     eObj_Flag_Sub_QoS_AtLeastOnce = 0x0080,
00124     eObj_Flag_Sub_QoS_ExactlyOnce = 0x0100,
00125     eObj_Flag_Sub_NoLocal         = 0x0200,
00126     eObj_Flag_Sub_Retained_IfNew  = 0x0400,
00127     eObj_Flag_Sub_Retained_Never  = 0x0800,
00128
00129     eObj_Flag_WaitForPub          = 0x1000,
00130     eObj_Flag_DynTopicName        = 0x2000,
00131 };
00132
00133 typedef enum {
00134     eObj_Serialize_Disabled     = 0x00,
00135     eObj_Serialize_String       = 0x01,
00136     eObj_Serialize_JSON         = 0x02,
```

```
00137     eObj_Serialize_JSON_ValOnly = 0x04,
00138     eObj_Serialize_MqttString   = 0x08,
00139     eObj_Serialize_BigEndian    = 0x10,
00140     eObj_Serialize_LittleEndian = 0x20,
00141     eObj_Serialize_NativeEndian = 0x40,
00142     eObj_Serialize_CURRENT_FORM = 0x80
00143 } eObj_Serialize_t;
00144
00145 class mqtt_leaf;
00146 namespace internal {
00147 class WrCritObj;
00148 }
00149 const char * GetPacketTypeString(Pkt::eType_t pkt);
00150 const char * GetReturnCodeString(MQTT::eResult_t result);
00151
00152 class Client : public TimeOutManager{
00153     friend void ::UserMain(void *pd);
00154 public:
00155     typedef enum {
00156         eLimit_Size_Tx,
00157         eLimit_Size_Rx,
00158         eLimit_InFlight_Tx,
00159         eLimit_InFlight_Rx,
00160         eLimit_Subscriptions,
00161         eLimit_SubAliases,
00162         eLimit_PublishAliases,
00163         eLimit_QoS,
00164         eLimit_SessionExpirary,
00165     } eSessionLimit;
00166
00167     typedef enum {
00168         eRateLimit_Publish,
00169         eRateLimit_Subscribe,
00170     } eRateLimitType;
00171
00172     typedef enum {
00173         eRateLimitAdj_None,
00174         eRateLimitAdj_Set,
00175         eRateLimitAdj_Add,
00176         eRateLimitAdj_Subtract
00177     } eRateLimitAdjustMethod;
00178
00179     typedef enum {
00180         eConnStat_Error,
00181         eConnStat_No_Broker_Configured,
00182         eConnStat_Resolving,
00183         eConnStat_InProgress,
00184         eConnStat_Established,
00185         eConnStat_Refused,
00186         eConnStat_Disconnected,
00187     } eConnectionStatus;
00188
00189 private:
00190     struct leaf_list_t {
00191         mqtt_leaf *head;
00192         mqtt_leaf *tail;
00193     };
00194
00195     typedef leaf_list_t leaf_queue_t;
00196
00197     static Client *s_clients;
00198     static Client *s_clientsTail;
00199     Client *pNextClient;
00200     leaf_list_t regList;
00201
00202     void dequeue_active(mqtt_leaf **ppNextReq, int activeQueue,
00203             uint32_t &waiting, leaf_queue_t *q);
00204
00205     OS_CRIT queueLock;
00206
00207     ParsedURI broker;
00208     NBString clientId;
00209     NBString username;
00210     NBString password;
00211 protected:
00212     NBString lwtTopic;
00213     NBString lwtMessage;
00214     NBString connMsg;
00215     NBString disconnMsg;
00216 private:
00217     uint32_t pubQueueWaiting;
00218     uint32_t subQueueWaiting;
00219     int m_dns_fd;
00220     leaf_queue_t pubQueue[MQTT_OBJ_PRIORITY_COUNT];
00221     leaf_queue_t subQueue[MQTT_OBJ_PRIORITY_COUNT];
00222
00223     PublishRequest nextPubReq;
```

```
00224     AckCbCtx nextSubAckCtx;
00225
00226     fd_set rd_fds;
00227     fd_set wr_fds;
00228     fd_set er_fds;
00229
00230     int connect();
00231     int SendConnect();
00232     inline bool ReadyToSend()
00233     {
00234         return mSess.txNeedSpace || pubQueueWaiting || subQueueWaiting
00235             || mSess.pubReqQueueWaiting || mSess.subReqQueue.head;
00236     }
00237
00238 protected:
00239     ConnectRequest connReq;
00240     PublishRequest lwtMsg;
00241
00242     internal::Session mSess;
00243     void initSessionInfo(const ParsedURI &brokerURI, bool bCleanStart, uint16_t maxKeepAlive,
      PublishRequest *pLwtMsg, RequestProperty *addProps);
00244     virtual internal::eClientState_t SetState_Rx(internal::eClientState_t newState);
00245     virtual internal::eClientState_t SetState_Tx(internal::eClientState_t newState);
00246
00247     void RegisterClient();
00248     void DeregisterClient();
00249
00250     void InterpolateLeafTopics();
00251
00252     void readyConnMsgsFromConfig(config_MqttClient &clientCfg,
00253             config_MqttLWT *lwtCfg, RequestProperty *addProps);
00254     Client(config_MqttClient &clientCfg,
00255             config_MqttLWT *lwtCfg = NULL, RequestProperty *addProps = NULL);
00256 public:
00257     Client(const char *brokerURI, const char *pClientId, uint16_t maxKeepAlive,
00258             bool bCleanStart = true, PublishRequest *lwtMsg = NULL,
00259             const char *pConnMsg = NULL, const char *pDisconnMsg = NULL,
00260             const char *pUsername = NULL, const char *pPassword = NULL,
00261             RequestProperty *addProps = NULL);
00262     Client(const ParsedURI *brokerURI, const char *pClientId, uint16_t maxKeepAlive,
00263             bool bCleanStart = true, PublishRequest *lwtMsg = NULL,
00264             const char *pConnMsg = NULL, const char *pDisconnMsg = NULL,
00265             const char *pUsername = NULL, const char *pPassword = NULL,
00266             RequestProperty *addProps = NULL);
00267
00268     int Connect(bool bWaitForConnection, const char *pNewBrokerURI,
00269             ConnAckCb_t notifyCb = NULL, void *notifyCtx = NULL);
00270     int Connect(bool bWaitForConnection = true, const ParsedURI *pNewBrokerURI = NULL,
00271             ConnAckCb_t notifyCb = NULL, void *notifyCtx = NULL);
00272     int Disconnect();
00273     void SetConnectNotify(ConnAckCb_t notifyCb, void *notifyCtx);
00274
00275     uint32_t GetLimit(eSessionLimit limit);
00276     bool     SetLimit(eSessionLimit limit, uint32_t value);
00277     int      SetRateLimit(eRateLimitType type, uint8_t bucketMax, uint8_t fillCount,
00278                           uint8_t fillTicks, eRateLimitAdjustMethod adjMethod,
00279                           uint8_t adjCount);
00280     eConnectionStatus GetConnectionStatus();
00281
00282     int QueuePublish(mqtt_leaf &leaf);
00283     int QueueSubscribe(mqtt_leaf &leaf);
00284     int QueueUnsubscribe(mqtt_leaf &leaf);
00285     int QueueAllSubscriptions();
00286
00287     int Publish(const char *topic, uint8_t *pData, uint32_t len, int prio, TickTimeout &t, uint8_t qos
      = 0);
00288     int Publish(const char *topic, const char *pStr, int prio, TickTimeout &t, uint8_t qos = 0);
00289     int Publish(PublishRequest *req, int prio, TickTimeout &t);
00290     int QueuePublish(PublishRequest *req, int *prio);
00291
00292     int Subscribe(TopicHandler &handler, int *handlerIdx, TickTimeout &t, uint8_t subOpts =
      Pkt::eSubOpt_NoLocal);
00293     int Subscribe(SubscribeRequest *req, int *handlerIdx, TickTimeout &t);
00294     int QueueSubscribe(SubUnsubQueueMsg *qmsg);
00295
00296     int Unsubscribe(int handlerIdx, TickTimeout &t, const char *topicFilter = NULL);
00297     int Unsubscribe(const char *topicFilter, TickTimeout &t);
00298     int QueueUnsubscribe(SubUnsubQueueMsg *qmsg);
00299
00300     int RegisterTopicHandler(TopicHandler *newHandler, int *handlerIdx);
00301     int RemoveTopicHandler(TopicHandler *handler);
00302     int RemoveTopicHandler(int handlerIdx);
00303
00304     void RegisterMqttObj(mqtt_leaf &leaf, uint32_t interval=0);
00305     void RemoveObj(mqtt_leaf &leaf);
00306
00307     int SendNextPubReq();
```

```
00308     int SendNextSubReq();
00309
00310     int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00311     TickTimeout ProcessSelectResult(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00312
00313 protected:
00314     static volatile OS_TCB *TaskTcb;
00315     static OS_CRIT s_ListCrit;
00316     static OS_SEM  s_TaskWakeSem;
00317     static void Task(void *);
00318     friend class internal::WrCritObj;
00319     friend class internal::Session;
00320
00321 public:
00322     static void StartTask(uint8_t taskPrio = MQTT_TASK_PRIO);
00323 };
00324
00325 class ConfiguredClient : public config_MqttClient, public Client {
00326 //    config_MqttClient clientCfg;
00327     config_MqttLWT    clientCfgLWT{*this,"LWT","Last Will and Testament message configuration"};
00328     ConfigEndMarker;
00329    void init(RequestProperty *addProps);
00330
00331     virtual internal::eClientState_t SetState_Tx(internal::eClientState_t newState) override;
00332 public:
00333     ConfiguredClient(config_obj &owner, const char* name, const char *desc = NULL,
00334             RequestProperty *addProps = NULL);
00335
00336     virtual void CommitTestedValue(uint32_t permission_mask) override;
00337 };
00338
00339 class mqtt_leaf : public TimeOutElement{
00340 public:
00341     enum {
00342         eObj_State_PubReady    = 0x00,
00343         eObj_State_PubPendTimer= 0x01,
00344         eObj_State_PubQueued   = 0x02,
00345         eObj_State_PubNeedAck  = 0x03,
00346
00347         eObj_State_SubNone         = 0x00,
00348         eObj_State_SubSendSub      = 0x10,
00349         eObj_State_SubNeedAck      = 0x20,
00350         eObj_State_SubReady        = 0x30,
00351         eObj_State_SubSendUnsub    = 0x40,
00352         eObj_State_SubUnsubNeedAck = 0x50,
00353
00354         eObj_State_PubState_MASK = 0x0F,
00355         eObj_State_SubState_MASK = 0xF0,
00356     };
00357
00358 private:
00359     OS_SEM *pSemDone;
00360     Client *mClient;
00361     mqtt_leaf *pNextLeaf;
00362
00363     mqtt_leaf *pNextPubReq;
00364     AckCbCtx pubAckCtx;
00365     mqtt_leaf *pNextSubReq;
00366     AckCbCtx subAckCtx;
00367
00368     uint16_t subHandlerId;
00369     uint8_t state;
00370     uint8_t priority;
00371
00372 protected:
00373     NBInterpolatedString topicName;
00374     eObj_Serialize_t publishForm;
00375     uint16_t flags;
00376
00377     mqtt_leaf(const char *topic, Client &mClient, uint16_t flags = 0);
00378
00379     virtual int GetPubTopicName(NBString &s);
00380     virtual int GetSubTopicFilter(NBString &s);
00381
00382     int QueuePublish();
00383     int QueueSubscribe();
00384     int QueueUnsubscribe();
00385
00386     int Assign();
00387     uint8_t PubState() { return state & eObj_State_PubState_MASK; }
00388     uint8_t SubState() { return state & eObj_State_SubState_MASK; }
00389     void    SetPubState(uint8_t newState)
00390     {
00391         state = ((state & ~eObj_State_PubState_MASK)
00392                 | (newState & eObj_State_PubState_MASK));
00393     }
00394     void    SetSubState(uint8_t newState)
```

```
00395     {
00396         state = ((state & ~eObj_State_SubState_MASK)
00397                 | (newState & eObj_State_SubState_MASK));
00398     }
00399
00400     int HandleSubAck(int mqttFd, PacketInfo::Msg *msg);
00401     int HandleUnsubAck(int mqttFd, PacketInfo::Msg *msg);
00402     int HandlePubAck(int mqttFd, PacketInfo::Msg *msg);
00403 public:
00404
00405     virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg);
00406     virtual int GetRenderLength(eObj_Serialize_t form = eObj_Serialize_CURRENT_FORM);
00407             int Publish(int waitForCompletion = -1);
00408             void SetPublishInterval(uint32_t intervalPeriodTicks);
00409
00410     virtual void SetSerializationForm(eObj_Serialize_t form);
00411     inline eObj_Serialize_t GetSerializationForm()
00412         { return publishForm; }
00413
00414     void SetFlags(uint16_t newFlags);
00415     void ClearFlags(uint16_t delFlags);
00416     inline uint16_t GetFlags() { return flags; }
00417
00418     inline int getPriority() { return priority; }
00419
00420     static int MqttRxPublish(Client *mClient, int mqttFd, PacketInfo::Msg *msg, void *ctx);
00421
00422     virtual int RenderToBuffer(eObj_Serialize_t form, uint8_t *buf, int buflen);
00423     virtual int RenderToFd(eObj_Serialize_t form, int fd);
00424     virtual void TimeElementEvent() override;
00425 private:
00426     struct leaf_queue_t {
00427         mqtt_leaf *head;
00428         mqtt_leaf *tail;
00429     };
00430     static int puback_cb(Client *mSess, void *ctx, PacketInfo::PublishAck *ack);
00431     static int suback_cb(Client *mSess, void *ctx, PacketInfo::PublishAck *ack);
00432
00433     static void dequeue_active(mqtt_leaf **ppNextReq, int activeQueue,
00434                                uint32_t &waiting, leaf_queue_t *q);
00435     friend class Client;
00436 };
00437
00438 class mqtt_int : public mqtt_leaf {
00439 private:
00440     int32_t val;
00441
00442 public:
00443     mqtt_int(int val, const char *topic, Client &mClient, uint16_t flags = 0);
00444     mqtt_int(const char *topic, Client &mClient, uint16_t flags = 0);
00445
00446     mqtt_int &operator=(int i);
00447     inline operator int() const { return val; }
00448
00449     virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg) override;
00450     virtual int RenderToBuffer(eObj_Serialize_t form, uint8_t *buf, int buflen) override;
00451     virtual int RenderToFd(eObj_Serialize_t form, int fd) override;
00452 };
00453
00454
00455
00456 class mqtt_uint : public mqtt_leaf {
00457 private:
00458     uint32_t val;
00459
00460 public:
00461     mqtt_uint(int val, const char *topic, Client &mClient);
00462     mqtt_uint(const char *topic, Client &mClient);
00463
00464     mqtt_uint &operator=(uint32_t i);
00465     inline operator unsigned() const { return val; }
00466
00467     virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg) override;
00468     virtual int RenderToBuffer(eObj_Serialize_t form, uint8_t *buf, int buflen) override;
00469     virtual int RenderToFd(eObj_Serialize_t form, int fd) override;
00470 };
00471
00472 class mqtt_float : public mqtt_leaf {
00473 private:
00474     double val;
00475     uint8_t serPrecision;
00476     char serStrFormat;
00477
00478 public:
00479     mqtt_float(int val, const char *topic, Client &mClient, uint16_t flags = 0);
00480     mqtt_float(const char *topic, Client &mClient, uint16_t flags = 0);
00481
```

```
00482     mqtt_float &operator=(double f);
00483     inline operator double() const { return val; }
00484
00485     void SetStringFormat(char formatChar, uint8_t precision)
00486     {
00487         switch (formatChar)
00488         {
00489             case 'f':
00490             case 'F':
00491             case 'e':
00492             case 'E':
00493             case 'g':
00494             case 'G':
00495             case 'a':
00496             case 'A':
00497                 serStrFormat = formatChar;
00498                 break;
00499             default:
00500                 break;
00501         }
00502         serPrecision = precision;
00503     }
00504
00505     virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg) override;
00506     virtual int RenderToBuffer(eObj_Serialize_t form, uint8_t *buf, int buflen) override;
00507     virtual int RenderToFd(eObj_Serialize_t form, int fd) override;
00508 };
00509
00510 class mqtt_string : public mqtt_leaf {
00511     NBString val;
00512
00513 public:
00514     mqtt_string(const char *val, const char *topic, Client &mClient, uint16_t flags = 0);
00515     mqtt_string(const char *topic, Client *mClient, uint16_t flags = 0);
00516
00517     mqtt_string &operator=(const NBString &s);
00518     mqtt_string &operator=(const char *s);
00519
00520     inline operator NBString() const { return val; }
00521
00522     virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg) override;
00523     virtual int RenderToBuffer(eObj_Serialize_t form, uint8_t *buf, int buflen) override;
00524     virtual int RenderToFd(eObj_Serialize_t form, int fd) override;
00525 };
00526
00527
00528 class ConfigExposer : public mqtt_leaf {
00529 public:
00530     enum eRenderFlag_t {
00531         eRender_Hidden,
00532         eRender_ModifiedOnly,
00533         eRender_Pretty
00534     };
00535 private:
00536     config_leaf *pLeaf;
00537     config_leaf *pendingPubLeaf;
00538
00539     uint32_t renderFlags;
00540
00541 public:
00542     ConfigExposer(config_leaf &leaf, const char *topicRoot, Client &mClient, uint16_t initFlags = 0,
      uint32_t renderFlags = 0);
00543 //    MqttConfigExposer(const char *topicRoot, Session *mClient);
00544
00545     virtual int GetPubTopicName(NBString &s) override;
00546     virtual int GetSubTopicFilter(NBString &s) override;
00547
00548     virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg) override;
00549     virtual int GetRenderLength(eObj_Serialize_t form = eObj_Serialize_CURRENT_FORM) override;
00550     virtual int RenderToBuffer(eObj_Serialize_t form, uint8_t *buf, int buflen) override;
00551     virtual int RenderToFd(eObj_Serialize_t form, int fd) override;
00552 };
00553
00554 typedef int (*SubHandlerCb)(int mqttFd, const char *pathExt, uint32_t msgLen);
00555
00556 class SubHandler : public mqtt_leaf {
00557      SubHandlerCb handler;
00558 public:
00559      SubHandler(SubHandlerCb handler, const char * topicFilter, Client &mClient, uint16_t flags = 0);
00560      virtual int SubscribeEvent(int mqttFd, PacketInfo::Msg *msg) override;
00561
00562 };
00563
00564 class PropMarker {
00565     uint32_t markers[8];
00566     int num;
00567 public:
```

```
00568     PropMarker()
00569         : markers{0,0,0,0,0,0,0,0}, num(0)
00570     {
00571     }
00572
00573     PropMarker(uint32_t val_0, uint32_t val_1, uint32_t val_2, uint32_t val_3,
00574             uint32_t val_4, uint32_t val_5, uint32_t val_6, uint32_t val_7)
00575         : markers{val_0, val_1, val_2, val_3, val_4, val_5, val_6, val_7}, num(0)
00576     {
00577         for (int i = 0; i < 8; i++)
00578             for (uint32_t v = 1; v; v <<= 1)
00579                 if (markers[i]&v)
00580                     num++;
00581     }
00582
00583     inline void reset() { for(int i = 0; i < 8; i++) markers[i] = 0; }
00584
00585     inline void set(int prop)
00586     {
00587         prop--;
00588         num += ((markers[prop>>5] & (1UL << (prop&0x1F))) == 0);
00589         markers[prop>>5] |= 1UL << (prop&0x1F);
00590     }
00591     inline void clr(int prop)
00592     {
00593         prop--;
00594         num -= ((markers[prop>>5] & (1UL << (prop&0x1F))) != 0);
00595         markers[prop>>5] &= ~(1UL << (prop&0x1F));
00596     }
00597     inline bool isset(int prop) { prop--; return ((markers[prop>>5] & (1UL << (prop&0x1F))) != 0); }
00598     inline int  count() { return num; }
00599     void show();
00600 };
00601
00602
00603
00604 }
00605 #endif   /* ----- #ifndef __MQTT_API_H  ----- */
00606
```

24.416 mqtt_internal.h

```
00001 #ifndef  __NBMQTT_INTERNAL_H
00002 #define  __NBMQTT_INTERNAL_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <predef.h>
00007 #include <nbrtos.h>
00008 #include <webclient/http_funcs.h>
00009 #include <mqtt/mqtt.h>
00010
00011 #define MQTT_MAX_SESSIONS (8)
00012 #define MQTT_OBJ_PRIORITY_COUNT (16)
00013
00014 #define MQTT_MATCH_HANDLER_MASK_SIZE ((MQTT_MAX_HANDLERS_PER_CONNECTION/32)
      +((MQTT_MAX_HANDLERS_PER_CONNECTION&0x1F)!=0))
00015
00016 #define __MQTT_STR(x) #x
00017 #define __MQTT_XSTR(x) __MQTT_STR(x)
00018
00019 #if (MQTT_MAX_SUPPORTED_IN_FLIGHT  > MQTT_MAX_MAX_SUPPORTED_IN_FLIGHT)
00020 #error __MQTT_STR(MQTT_MAX_SUPPORTED_IN_FLIGHT) ## " must be less than " ##
      __MQTT_XSTR(MQTT_MAX_MAX_SUPPORTED_IN_FLIGHT)
00021 #endif
00022
00023 #define MQTT_STATE_FREE 0x0
00024 #define MQTT_STATE_ALLOCED 0x1
00025 #define MQTT_STATE_CONN_SENT 0x2
00026 #define MQTT_STATE_ESTABLISHED 0x3
00027
00028 namespace MQTT {
00029 class Client;
00030
00031 namespace internal {
00032 enum {
00033     eTransport_TCP,
00034     eTransport_TLS,
00035     eTransport_WS,
00036     eTransport_WSS
00037 };
00038
00039 typedef enum {
00040     eState_Free,
00041     eState_Alloced,
```

```
00042     eState_StartConnect,
00043     eState_NetConnected,
00044     eState_NoBrokerURI,
00045     eState_Aborted,
00046     eState_Connecting,
00047     eState_ConnAck,
00048     eState_Authenticating,
00049     eState_Disconnecting,
00050     eState_Closing,
00051     eState_Established,
00052     eState_TxObjPub,
00053     eState_TxSubUnsub,
00054     eState_TxPub,
00055     eState_TxObjSubUnsub,
00056     eState_MsgFixed,
00057     eState_MsgDispatch,
00058     eState_MsgAck,
00059     eState_PingReq,
00060     eState_PingResp,
00061     eState_ProtocolError,
00062 } eClientState_t;
00063
00064 typedef enum {
00065     ePktState_Done,
00066     ePktState_Writing,
00067     ePktState_NeedAck,
00068     ePktState_WritingAck,
00069     ePktState_NeedReceived,
00070     ePktState_WritingReceived,
00071     ePktState_NeedRelease,
00072     ePktState_WritingRelease,
00073     ePktState_NeedComplete,
00074     ePktState_WritingComplete,
00075 } ePacketState_t;
00076
00077 struct MqttAckTracker {
00078     uint16_t window;
00079     uint16_t inFlight[MQTT_MAX_SUPPORTED_IN_FLIGHT];
00080     uint8_t  nextIdx;
00081     uint8_t  lastIdx;
00082
00083 };
00084
00085
00086 struct HandlerMatchMask {
00087     uint32_t mask[MQTT_MATCH_HANDLER_MASK_SIZE];
00088     int num;
00089     void reset();
00090     void Assign(const HandlerMatchMask &rhs);
00091     void SetFrom(const HandlerMatchMask &rhs);
00092     HandlerMatchMask & operator=(const HandlerMatchMask &rhs);
00093     HandlerMatchMask & operator|=(const HandlerMatchMask &rhs);
00094 };
00095
00096
00097 struct RxTopicAlias {
00098     NBString name;
00099     HandlerMatchMask matches;
00100 };
00101
00102 struct ffBufInterposer {
00103     PoolPtr ppCurr;
00104     uint8_t *pCurr;
00105     int wrFd;
00106     int thisFd;
00107     bool closed;
00108     uint32_t offset;
00109
00110     PoolPtr ppStart;
00111     uint8_t *origStart;
00112     PoolPtr ppEnd;
00113     uint8_t *end;
00114 };
00115
00116 struct TxAckCtx {
00117     uint16_t id;
00118     Pkt::eType_t expecting;
00119     uint8_t subHandlerIdx;
00120     AckCbCtx ctx;
00121 };
00122
00123 struct FdInterposer {
00124     int      baseFd;
00125     uint32_t availSpace;
00126     uint32_t remLen;
00127     bool     closed;
00128 };
```

```
00129
00130 struct RateBucket {
00131     uint8_t bucket;
00132     uint8_t max;
00133     uint8_t count;
00134     uint8_t ticks;
00135     tick_t lastFill;
00136 };
00137
00138 struct PublishQueue {
00139     PublishRequest *head;
00140     PublishRequest *tail;
00141 };
00142
00143 struct SubUnsubQueue {
00144     SubUnsubQueueMsg *head;
00145     SubUnsubQueueMsg *tail;
00146 };
00147
00148 struct Session {
00149     Client *pOwner;
00150     OS_CRIT wrCrit;
00151     OS_CRIT qCrit;
00152     OS_CRIT rdCrit;
00153     ConnectRequest *connReq;
00154     NBString reasonStr;
00155     NBString respInfo;
00156     NBString serverRef;
00157     NBString authMethod;
00158     NBString rxPubTopic;
00159     fifo_buffer_storage authData;
00160
00161     FdInterposer interposer;
00162     int transportFd;
00163     int interFd;
00164     int notifFd;
00165     bool txWasBlocked;
00166     int transportType;
00167     fifo_buffer_storage *fdRxBuf;
00168
00169     uint32_t txMaxSize;
00170     uint32_t rxMaxSize;
00171     uint32_t sessionStart;
00172     uint32_t sessionTime;
00173     tick_t   needKeepAliveBy;
00174     uint16_t keepAliveInterval;
00175     uint16_t topicAliases_max;
00176     uint16_t topicAliases_inUse;
00177     Pkt::eReasonCode_t reason_disconn;
00178
00179     bool retainAvail:1;
00180     bool wildSubAvail:1;
00181     bool subIdsAvail:1;
00182     bool sharedSubsAvail:1;
00183     bool updateFdSets:1;
00184
00185     uint16_t sessExpIntvl;
00186     uint8_t maxQoS;
00187
00188     eClientState_t txState;
00189     eClientState_t rxState;
00190
00191     int txNeedSpace;
00192     int rxNeedData;
00193
00194 #ifdef TCP_NOCOPY_TX
00195     // the amount of space allocated within each PoolBuffer in the tcp TxBuffer
00196     //  for data (in the event of TCP_TX_NOCOPY it's less than ETHER_BUFFER_SIZE)
00197     uint32_t txInternalBuffersLen;
00198 #endif
00199     bool txBuffersAlreadyElevated;
00200     bool txBuffersNeedReduced;
00201     int  txBufferSpaceAvail;
00202
00203     uint16_t rxNeedsToAckId;
00204     Pkt::eType_t rxSendAckType;
00205     Pkt::eReasonCode_t rxAckReason;
00206
00207     uint16_t txWindow;
00208     uint16_t rxWindow;
00209     beuint16_t seqNum;
00210     uint16_t nextHandlerIdx;
00211
00212     RateBucket rateLimit_TxPublish;
00213     RateBucket rateLimit_Subscribe;
00214
00215     int logFd;
```

```
00216     int logLevel;
00217
00218 //    MqttPublisher *currPublisher;
00219 //    MqttSubHandler *currSubHandle;
00220
00221     ConnAckCb_t   connAckCb;
00222     void         *connAckCtx;
00223     DisconnCb_t   disconnCb;
00224
00225     uint32_t pubReqQueueWaiting;
00226     uint32_t pubReqQueueActive;
00227     PublishQueue  pubReqQueue[MQTT_OBJ_PRIORITY_COUNT];
00228     SubUnsubQueue      subReqQueue;
00229
00230     PacketInfo::Msg rxHdr;
00231     HandlerMatchMask rxPubMatches;
00232     RxTopicAlias rxAliases[MQTT_MAX_SUPPORTED_RX_TOPIC_ALIASES];
00233     TxAckCtx txNotAcked[MQTT_MAX_SUPPORTED_IN_FLIGHT];
00234     uint16_t rxNotReleased[MQTT_MAX_SUPPORTED_IN_FLIGHT];
00235     TopicHandler handlers[MQTT_MAX_HANDLERS_PER_CONNECTION];
00236
00237     Session(Client *owner);
00238     ~Session() {};
00239     int netConnect();
00240     inline Client *getOwner() { return pOwner; }
00241
00242     void fillRateBuckets();
00243     void ResetState();
00244     int  MakeTxSpace(uint32_t need, TickTimeout &t);
00245     void SkipInterposerData(int nbytes);
00246     void ReadyInterposerData(int nbytes);
00247
00248     int WaitForTxQuota(TickTimeout &t);
00249     int MakeTcpTxSpace(uint32_t need, TickTimeout &t, bool force = false);
00250     int WaitForSubscribeQuota(TickTimeout &t);
00251
00252     void SetInterposerAvailSpace(int availSpace);
00253
00254     MQTT::eResult_t RegisterTopicHandler(TopicHandler *newHandler, uint16_t *retIdx);
00255     int             GetFreeHandlerIdx();
00256     MQTT::eResult_t RemoveTopicHandler(uint16_t handlerIdx);
00257     MQTT::eResult_t RemoveTopicHandlerByFilter(const NBString &filter, uint16_t handlerIdx, bool
      strict = true);
00258     MQTT::eResult_t SetRxMatchesFromAlias(uint16_t alias);
00259     void            RegisterRxAlias(NBString *topicName, uint16_t aliasIdx);
00260     void            SetMatchingHandlers(NBString *topicName, HandlerMatchMask *matches);
00261
00262     int GetNewPktId();
00263
00264     TxAckCtx *  addTxAckCtx(uint16_t id);
00265     TxAckCtx *  getTxAckCtx(uint16_t id);
00266     bool        addRxPktIdCtx(uint16_t pktId);
00267     int         getRxPktIdCtx(uint16_t pktId);
00268     int         removeRxPktId(uint16_t pktId);
00269
00270     int QueueSubUnsub(SubUnsubQueueMsg *qmsg, TickTimeout &t);
00271     int DequeueSubUnsub(SubUnsubQueueMsg *qmsg, TickTimeout &t);
00272     int PublishFromQueue(TickTimeout &t);
00273     int QueuePublish(PublishRequest *req, int *prio, TickTimeout &t);
00274     int DequeuePublish(PublishRequest *req, int prio, TickTimeout &t);
00275
00276     int RxFixedHeader();
00277     int RxMsg();
00278
00279
00280     int RxMsg_ConnAck();
00281     int RxMsg_Publish();
00282     int RxMsg_PubRel();
00283     int RxMsg_PubAck(Pkt::eType_t type);
00284     int RxMsg_Disconnect();
00285
00286
00287     int SendMsg_Connect(ConnectRequest *req, TickTimeout &t);
00288     int SendMsg_Publish(PublishRequest *req, TickTimeout &t, bool force = false);
00289     int SendMsg_Subscribe_UserProperties(SubscribeRequest *req);
00290     int SendMsg_Subscribe_Filters(SubscribeRequest *req);
00291     int SendMsg_Queued_SubUnsub(TickTimeout &t);
00292     int SendMsg_SubUnsub(SubUnsubQueueMsg *qmsg, TickTimeout &t);
00293     int SendMsg_Ack(TickTimeout &t);
00294     int SendMsg_Disconnect(TickTimeout &t);
00295
00296     int CleanupConnection();
00297
00298     int RxMsg_ConnAck_Properties();
00299     int RxMsg_ConnAck_SessionExpirary(fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00300     int RxMsg_ConnAck_Property_ClientId(
00301             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
```

```
00302     int RxMsg_ConnAck_Property_KeepAlive(
00303             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00304     int RxMsg_ConnAck_Property_AuthMethod(
00305             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00306     int RxMsg_ConnAck_Property_AuthData(
00307             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00308     int RxMsg_ConnAck_Property_ResponseInfo(
00309             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00310     int RxMsg_ConnAck_Property_ServerReference(
00311             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00312     int RxMsg_ConnAck_Property_ReasonString(
00313             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00314     int RxMsg_ConnAck_Property_ReceiveMax(
00315             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00316     int RxMsg_ConnAck_Property_TopicAliasMax(
00317             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00318     int RxMsg_ConnAck_Property_MaximumQoS(
00319             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00320     int RxMsg_ConnAck_Property_RetainAvail(
00321             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00322     int RxMsg_ConnAck_Property_UserProperty(
00323             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00324     int RxMsg_ConnAck_Property_MaxPktSize(
00325             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00326     int RxMsg_ConnAck_Property_WildSubAvail(
00327             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00328     int RxMsg_ConnAck_Property_SubIdAvail(
00329             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00330     int RxMsg_ConnAck_Property_SharedSubsAvail(
00331             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00332
00333     int RxMsg_Publish_Properties();
00334     int RxMsg_Publish_SubId(fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00335     int RxMsg_Publish_TopicAlias(fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00336
00337     int RxMsg_Disconn_Properties();
00338     int RxMsg_Disconn_Property_SessionExpirary(
00339             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00340     int RxMsg_Disconn_Property_ReasonString(
00341             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00342     int RxMsg_Disconn_Property_ServerReference(
00343             fifo_buffer_storage::PeekIterator &pkr, uint32_t &remProps);
00344
00345     int writeProperty(const RequestProperty *prop, uint32_t *propsLen);
00346     int SendMsg_PropertiesBuf(const uint8_t *propBuf, uint32_t propsLen);
00347     int SendMsg_PropertyList(RequestProperty *propList, uint32_t propsLen);
00348     int SendMsg_Payload_PoolPtrList(PoolPtr pp, uint32_t payloadLen);
00349 };
00350
00351 const char * GetString_State(MQTT::internal::eClientState_t state);
00352
00353 }
00354 }
00355
00356 #endif   /* ----- #ifndef __NBMQTT_INTERNAL_H  ----- */
```

24.417 mqtt_msg_reqs.h

```
00001 #ifndef  __MQTT_MSG_REQS_H
00002 #define  __MQTT_MSG_REQS_H
00003 #include <predef.h>
00004 #include <basictypes.h>
00005 #include <nbstring.h>
00006 #include <webclient/http_funcs.h>
00007 #include <mqtt/msg_types.h>
00008
00009 namespace MQTT {
00010 namespace PacketInfo {
00011 struct ConnectAck {
00012     uint8_t flags;
00013     uint32_t remLen;
00014     bool sessPresent;
00015     uint32_t propsLen;
00016     Pkt::eReasonCode_t reason;
00017 };
00018
00019 struct Publish {
00020     uint8_t flags;
00021     uint32_t remLen;
00022     NBString *topic;
00023     uint16_t pktId;
00024     uint32_t propsLen;
00025     uint32_t payLen;
00026 };
00027
```

```
00028 struct PublishAck{
00029     uint8_t flags;
00030     uint32_t remLen;
00031     uint16_t pktId;
00032     uint32_t propsLen;
00033     Pkt::eReasonCode_t reason;
00034 };
00035
00036 struct SubscribeAck {
00037     uint8_t flags;
00038     uint32_t remLen;
00039     uint16_t pktId;
00040     uint32_t propsLen;
00041     Pkt::eReasonCode_t reason;
00042     NBString *reasonStr;
00043 };
00044
00045 struct UnsubscribeAck {
00046     uint8_t flags;
00047     uint32_t remLen;
00048     uint16_t pktId;
00049     uint32_t propsLen;
00050     uint32_t paylLen;
00051     Pkt::eReasonCode_t reason;
00052     uint32_t subId;
00053     NBString *reasonStr;
00054 };
00055
00056 struct Disconnect {
00057     uint8_t flags;
00058     uint32_t remLen;
00059     uint32_t propsLen;
00060     Pkt::eReasonCode_t reason;
00061     NBString *reasonStr;
00062     NBString *serverRef;
00063 };
00064
00065 struct Msg {
00066     union {
00067         struct {
00068             uint8_t flags;
00069             uint32_t remLen;
00070         };
00071         ConnectAck       connack;
00072         Publish          publish;
00073         PublishAck       puback;
00074         SubscribeAck     suback;
00075         UnsubscribeAck   unsuback;
00076         Disconnect       disconn;
00077     };
00078     inline Pkt::eType_t GetType()
00079         { return (Pkt::eType_t)(flags>>4); }
00080     inline uint8_t GetFlags()
00081         { return (flags & 0xF); }
00082     Msg() {};
00083     ~Msg() {};
00084 };
00085
00086
00087 }
00088
00089 class Client;
00090
00091 typedef int (*ConnAckCb_t) (Client *mConn, int mqttFd, void *ctx, PacketInfo::ConnectAck *connack);
00092 typedef bool(*TopicMatchCb_t)(Client *mConn, NBString *topicName, void *matchCtx);
00093 typedef int (*WrPropCb_t) (Client *mConn, int mqttFd, void *ctx, uint32_t propLen);
00094 typedef int (*WrPayloadCb_t) (Client *mClient, int mqttFd, void *ctx, uint32_t payloadLen);
00095 typedef int (*AckCb_t) (Client *mClient, void *ctx, PacketInfo::PublishAck *ack);
00096 typedef int (*DisconnCb_t) (Client *mClient, PacketInfo::Disconnect *disconn);
00097
00098 struct TopicHandler;
00099
00100 struct AckCbCtx {
00101     AckCb_t cb;
00102     void *ctx;
00103 };
00104
00105 typedef enum {
00106     ePropListType_PData,
00107     ePropListType_PList,
00108     ePropListType_Callback,
00109     ePropListType_None
00110 } ePropertyListType;
00111
00112 typedef enum {
00113     ePayloadType_PData,
00114     ePayloadType_PoolPtr,
```

```
00115     ePayloadType_Callback,
00116     ePayloadType_None
00117 } ePayloadType;
00118
00119 typedef enum {
00120     ePropValType_ValByte,
00121     ePropValType_ValShort,
00122     ePropValType_ValLong,
00123     ePropValType_ValPStr,
00124     ePropValType_ValPStrPair,
00125     ePropValType_ValPData,
00126     ePropValType_ValLocStr,
00127     ePropValType_ValLocStrPair,
00128     ePropValType_ValLocData,
00129     ePropValType_Callback
00130 } ePropertyValueType;
00131
00132 struct RequestProperty {
00133     RequestProperty *pNext;
00134     ePropertyValueType valType;
00135     uint8_t prop;
00136     union {
00137         struct {
00138             union {
00139                 uint8_t  valByte;
00140                 uint16_t valShort;
00141                 uint32_t valLong;
00142                 uint16_t data_len;
00143                 uint16_t sLen[2];
00144             };
00145             union {
00146                 char * pStr[2];
00147                 uint8_t * pData;
00148                 struct {
00149                     WrPayloadCb_t cb;
00150                     void *ctx;
00151                 };
00152             };
00153         };
00154         struct {
00155             union {
00156                 uint8_t  _val8;
00157                 uint16_t _val16;
00158                 uint32_t _val32;
00159                 uint16_t _data_len;
00160             };
00161             uint8_t data[];
00162         };
00163     };
00164     uint32_t GetEncodedLength();
00165     uint32_t ListGetEncodedLength();
00166 };
00167
00168 struct PublishRequest;
00169
00170 struct ConnectRequest {
00171     ParsedURI brokerURI;
00172     uint8_t flags;
00173     uint16_t keepalive;
00174     const char *clientId;
00175     const char *username;
00176     const char *password;
00177
00178     uint32_t propsType:4;
00179     uint32_t propsLen:28;
00180     union {
00181         const uint8_t *propBuf;
00182         RequestProperty *propList;
00183         struct {
00184             WrPropCb_t propWrCb;
00185             void * propWrCtx;
00186         };
00187     };
00188     PublishRequest *pLwtReq;
00189     ConnAckCb_t   connAckCb;
00190     void         *connAckCtx;
00191 };
00192
00193
00194 struct PublishRequest {
00195     PublishRequest *pNextReq;
00196     PublishRequest *pPrevReq;
00197     uint8_t flags;
00198     const char *topicName;
00199     uint32_t propsType : 4; // ePropertyValueType
00200     uint32_t propsLen : 28;
00201
```

```
00202     union {
00203         const uint8_t *propBuf;
00204         RequestProperty *propList;
00205         struct {
00206             WrPropCb_t propWrCb;
00207             void *propWrCtx;
00208         };
00209     };
00210     uint32_t payloadType : 4;
00211     uint32_t payloadLen : 28;
00212     union {
00213         const uint8_t *pBuf;
00214 //        PoolPtr pp;
00215         pool_buffer * pp;
00216         struct {
00217             WrPayloadCb_t payWrCb;
00218             void *payWrCtx;
00219         };
00220     };
00221     AckCbCtx *cbCtx;
00222     void dump();
00223 };
00224
00225 typedef enum {
00226     eSubReq_Pay_cstr,
00227     eSubReq_Pay_cstrList,
00228     eSubReq_Pay_nbstr,
00229     eSubReq_Pay_nbstrList,
00230     eSubReq_Pay_cb,
00231 } eSubscribeRequestPayloadType;
00232
00233 typedef enum {
00234     eSubReq_UProp_None,
00235     eSubReq_UProp_up,
00236     eSubReq_UProp_ptrupList,
00237     eSubReq_UProp_upList,
00238     eSubReq_UProp_nbstr,
00239     eSubReq_UProp_ptrnbstrList,
00240     eSubReq_UProp_nbstrList,
00241     eSubReq_UProp_cb,
00242 } eSubscribeRequestUserPropertyType;
00243
00244 struct UserProperty {
00245     const char *key;
00246     const char *val;
00247 };
00248
00249 struct UserProperty_NbStr {
00250     const NBString *key;
00251     const NBString *val;
00252 };
00253
00254
00255 struct SubscribeFilter {
00256     const char *filter;
00257     uint8_t     opts;
00258 };
00259
00260 struct SubscribeFilter_NbStr {
00261     const NBString *filter;
00262     uint8_t     opts;
00263 };
00264
00265 struct SubscribeRequest {
00266     SubscribeRequest *pNextSubReq;
00267     int     subId;
00268     uint8_t subType : 4;
00269     uint8_t propsType: 4;
00270     TopicHandler *pubHandler;
00271     union {
00272         SubscribeFilter_NbStr filterNB;
00273         SubscribeFilter filter;
00274         SubscribeFilter *filterList;
00275         SubscribeFilter_NbStr *filterListNB;
00276         struct {
00277             uint32_t totalStrLen;
00278             WrPayloadCb_t filtersCb;
00279             void *filtersCtx;
00280         };
00281     };
00282     union{
00283         struct {
00284             uint32_t listLen;
00285             UserProperty *pUPropList;
00286         };
00287         UserProperty UProp; // optional, needs both key and val
00288         UserProperty_NbStr nbUProp;
```

```
00289         struct {
00290             uint32_t _listLen0;
00291             UserProperty_NbStr *pUPropNBList;
00292         };
00293         struct {
00294             uint32_t _listLen1;
00295             UserProperty UPropList[];
00296         };
00297         struct {
00298             uint32_t _listLen2;
00299             UserProperty_NbStr UPropNBList[];
00300         };
00301         struct {
00302             uint32_t totalUserPropsLen;
00303             WrPayloadCb_t UPropsCb;
00304             void *UPropsCbCtx;
00305         };
00306     };
00307 };
00308
00309 struct SubUnsubQueueMsg {
00310     SubUnsubQueueMsg *pNextReq;
00311     SubscribeRequest *req;
00312     AckCbCtx *cb;
00313     uint16_t handlerIdx;
00314     bool unsub;
00315 };
00316
00317 }
00318
00319
00320 #endif   /* ----- #ifndef __MQTT_MSG_REQS_H  ----- */
```

24.418 msg_types.h

```
00001 #ifndef  __NBMQTT_MSG_TYPES_H
00002 #define  __NBMQTT_MSG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <predef.h>
00007 #include <basictypes.h>
00008 #include <buffers.h>
00009
00010 #define MQTT_DEFAULT_PORT (1883)
00011 #define MQTT_DEFAULT_SECURE_PORT (8883)
00012
00013 namespace MQTT {
00014 namespace Pkt {
00015
00016 typedef enum {
00017     eType_None        = 0x0,
00018     eType_Connect     = 0x1,
00019     eType_ConnAck     = 0x2,
00020     eType_Publish     = 0x3,
00021     eType_PubAck      = 0x4,
00022     eType_PubRec      = 0x5,
00023     eType_PubRelease  = 0x6,
00024     eType_PubComp     = 0x7,
00025     eType_Subscribe   = 0x8,
00026     eType_SubAck      = 0x9,
00027     eType_UnSubscribe = 0xA,
00028     eType_UnSubAck    = 0xB,
00029     eType_PingReq     = 0xC,
00030     eType_PingResp    = 0xD,
00031     eType_DisConn     = 0xE,
00032     eType_Auth        = 0xF
00033 } eType_t;
00034
00035 typedef enum {
00036     eFlag_Retain          = 0x01,
00037     eFlag_QoS_AtLeastOnce = 0x02,
00038     eFlag_QoS_ExactlyOnce = 0x04,
00039     eFlag_Duplicate       = 0x08,
00040
00041     eFlag_PubRel          = 0x02,
00042     eFlag_Subscribe       = 0x02,
00043     eFlag_UnSubscribe     = 0x02,
00044 } eFlag_t;
00045
00046 typedef enum {
00047     ePropType_None           = 0x00,
00048     ePropType_PayFmt         = 0x01,
00049     ePropType_MsgExpIntv     = 0x02,
00050     ePropType_ContentType    = 0x03,
```

```
00051     ePropType_RespTopic      = 0x08,
00052     ePropType_CorrData       = 0x09,
00053     ePropType_SubId          = 0x0B,
00054     ePropType_SessExpIntv    = 0x11,
00055     ePropType_AsgnClientId   = 0x12,
00056     ePropType_SKeepAlive     = 0x13,
00057     ePropType_AuthMethod     = 0x15,
00058     ePropType_AuthData       = 0x16,
00059     ePropType_ReqProbInfo    = 0x17,
00060     ePropType_WillDlyIntv    = 0x18,
00061     ePropType_ReqRespInfo    = 0x19,
00062     ePropType_RespInfo       = 0x1A,
00063     ePropType_ServerRef      = 0x1C,
00064     ePropType_ReasonStr      = 0x1F,
00065     ePropType_ReceiveMax     = 0x21,
00066     ePropType_TopicAliasMax  = 0x22,
00067     ePropType_TopicAlias     = 0x23,
00068     ePropType_MaximumQoS     = 0x24,
00069     ePropType_RetainAvail    = 0x25,
00070     ePropType_UserProperty   = 0x26,
00071     ePropType_MaxPktSiz      = 0x27,
00072     ePropType_WildSubAvail   = 0x28,
00073     ePropType_SubIdAvail     = 0x29,
00074     ePropType_SharedSubAvail = 0x2A,
00075
00076     ePropType_ERROR          = 0xFF,
00077     ePropType_Unknown        = 0xFF,
00078     ePropType_MAXIMUM        = 0xFF
00079 } eProperty_t;
00080
00081 typedef enum {
00082     eFieldType_None,
00083
00084     eFieldType_Byte,
00085     eFieldType_Short,
00086     eFieldType_Long,
00087     eFieldType_VarInt,
00088     eFieldType_U8Str,
00089     eFieldType_U8Str_Pair,
00090     eFieldType_Binary,
00091
00092     ePropField_None           = eFieldType_None,
00093     ePropField_PayFmt         = eFieldType_Byte,
00094     ePropField_MsgExpIntv     = eFieldType_Long,
00095     ePropField_ContentType    = eFieldType_U8Str,
00096     ePropField_RespTopic      = eFieldType_U8Str,
00097     ePropField_CorrData       = eFieldType_Binary,
00098     ePropField_SubId          = eFieldType_VarInt,
00099     ePropField_SessExpIntv    = eFieldType_Long,
00100     ePropField_AsgnClientId   = eFieldType_U8Str,
00101     ePropField_SKeepAlive     = eFieldType_Short,
00102     ePropField_AuthMethod     = eFieldType_U8Str,
00103     ePropField_AuthData       = eFieldType_Binary,
00104     ePropField_ReqProbInfo    = eFieldType_Byte,
00105     ePropField_WillDlyIntv    = eFieldType_Long,
00106     ePropField_ReqRespInfo    = eFieldType_Byte,
00107     ePropField_RespInfo       = eFieldType_U8Str,
00108     ePropField_ServerRef      = eFieldType_U8Str,
00109     ePropField_ReasonStr      = eFieldType_U8Str,
00110     ePropField_ReceiveMax     = eFieldType_Short,
00111     ePropField_TopicAliasMax  = eFieldType_Short,
00112     ePropField_TopicAlias     = eFieldType_Short,
00113     ePropField_MaximumQoS     = eFieldType_Byte,
00114     ePropField_RetainAvail    = eFieldType_Byte,
00115     ePropField_UserProperty   = eFieldType_U8Str_Pair,
00116     ePropField_MaxPktSiz      = eFieldType_Long,
00117     ePropField_WildSubAvail   = eFieldType_Byte,
00118     ePropField_SubIdAvail     = eFieldType_Byte,
00119     ePropField_SharedSubAvail = eFieldType_Byte,
00120     ePropField_Unknown        = eFieldType_None
00121 } eField_t;
00122
00123 typedef enum {
00124     eReason_Success                      = 0x00,
00125     eReason_NormalDisconn                = 0x00,
00126     eReason_Grant_QoS_AtMostOnce         = 0x00,
00127     eReason_Grant_QoS_AtLeastOnce        = 0x01,
00128     eReason_Grant_QoS_ExactlyOnce        = 0x02,
00129     eReason_Disconn_w_Will               = 0x04,
00130     eReason_NoMatch_Subscribers          = 0x10,
00131     eReason_NoSubscriptions_Existed      = 0x11,
00132     eReason_Cont_Auth                    = 0x18,
00133     eReason_ReAuth                       = 0x19,
00134     eReason_Error_Unspecified            = 0x80,
00135     eReason_Error_MalformedPkt           = 0x81,
00136     eReason_Error_Protocol               = 0x82,
00137     eReason_Error_Implementation         = 0x83,
```

```
00138     eReason_Error_UnsupportedProtoVer     = 0x84,
00139     eReason_Error_ClientIdInvalid         = 0x85,
00140     eReason_Error_UserPassword            = 0x86,
00141     eReason_Error_NotAuthorized           = 0x87,
00142     eReason_Error_ServerUnavail           = 0x88,
00143     eReason_Error_ServerBusy              = 0x89,
00144     eReason_Error_Banned                  = 0x8A,
00145     eReason_Error_ServerShutdown          = 0x8B,
00146     eReason_Error_BadAuthMethod           = 0x8C,
00147     eReason_Error_KATimeout               = 0x8D,
00148     eReason_Error_SessTakeover            = 0x8E,
00149     eReason_Error_FilterInvalid           = 0x8F,
00150     eReason_Error_NameInvalid             = 0x90,
00151     eReason_Error_PktId_InUse             = 0x91,
00152     eReason_Error_PktId_NotFound          = 0x92,
00153     eReason_Error_RcvMax_Exceeded         = 0x93,
00154     eReason_Error_TopicAlias_Invalid      = 0x94,
00155     eReason_Error_TooLarge                = 0x95,
00156     eReason_Error_MsgRate                 = 0x96,
00157     eReason_Error_Quota                   = 0x97,
00158     eReason_Error_Admin                   = 0x98,
00159     eReason_Error_PayFmt_Invalid          = 0x99,
00160     eReason_Error_NotSupported_Retain     = 0x9A,
00161     eReason_Error_NotSupported_QoS        = 0x9B,
00162     eReason_Error_UseOther                = 0x9C,
00163     eReason_Error_ServerMoved             = 0x9D,
00164     eReason_Error_NotSupported_SharedSubs = 0x9E,
00165     eReason_Error_ConnRate                = 0x9F,
00166     eReason_Error_MaxConnTime             = 0xA0,
00167     eReason_Error_NotSupported_SubIds     = 0xA1,
00168     eReason_Error_NotSupported_WildSubs   = 0xA2,
00169
00170     eReason_Error_Transport_Error        = 0xFE,
00171     eReason_Unknown = 0xFF,
00172     eReason_MAXIMUM = 0xFF
00173
00174 } eReasonCode_t;
00175
00176 typedef enum {
00177     eConnFlag_RESERVED             = 0x01,
00178     eConnFlag_CleanStart           = 0x02,
00179     eConnFlag_Will                 = 0x04,
00180     eConnFlag_Will_QoS_AtLeastOnce = 0x08,
00181     eConnFlag_Will_QoS_ExactlyOnce = 0x10,
00182     eConnFlag_Will_Retain          = 0x20,
00183     eConnFlag_Password             = 0x40,
00184     eConnFlag_Username             = 0x80,
00185 } eConnFlag_t;
00186
00187 typedef enum {
00188     eSubOpt_QoS_AtLeastOnce    = 0x01,
00189     eSubOpt_QoS_ExactlyOnce    = 0x02,
00190     eSubOpt_NoLocal            = 0x04,
00191     eSubOpt_AsPublished        = 0x08,
00192     eSubOpt_SendRetained_IfNew = 0x10,
00193     eSubOpt_SendRetained_Never = 0x20,
00194     eSubOpt_RESERVED_0         = 0x40,
00195     eSubOpt_RESERVED_1         = 0x80,
00196 } eSubscribeOption_t;
00197
00198 typedef enum {
00199     eConnAckFlag_SessPresent = 0x01,
00200 } eConnAckFlag_t;
00201
00202 struct mqtt_str_t {
00203     beuint16_t len;
00204     char pData[];
00205 };
00206 }
00207 }
00208 #endif   /* ----- #ifndef __NBMQTT_MSG_TYPES_H  ----- */
```

24.419 multicast.h File Reference

NetBurner Multicast API.

```
#include <predef.h>
#include <nbrtos.h>
#include <nettypes.h>
```

Functions

- void RegisterMulticastFifo4 (IPADDR4 group, uint16_t dest_port, OS_FIFO *pfifo, int interface=0)

  Register to join a Multicast group.
- void UnregisterMulticastFifo4 (IPADDR4 group, uint16_t destination_port, int interface=0)

  Unregister from a Multicast group.
- void RegisterMulticastFifo6 (IPADDR group, uint16_t dest_port, OS_FIFO *pfifo, int interface=0)

  Register to join a Multicast group.
- void UnregisterMulticastFifo6 (IPADDR group, uint16_t destination_port, int interface=0)

  Unregister from a Multicast group.

24.419.1 Detailed Description

NetBurner Multicast API.

24.420 multicast.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef _NB_MULTICAST_H
00016 #define _NB_MULTICAST_H
00017 // NB Definitions
00018 #include <predef.h>
00019
00020 // NB Libs
00021 #include <nbrtos.h>
00022 #include <nettypes.h>
00023
00024 //  This Module provides code for joining multicast groups, using the IGMP protocol per RFC1112 and
      RFC 2236
00025
00044 void RegisterMulticastFifo4(IPADDR4 group, uint16_t dest_port, OS_FIFO *pfifo, int interface = 0);
00045
00058 void UnregisterMulticastFifo4(IPADDR4 group, uint16_t destination_port, int interface = 0);
00059
00060 #ifdef IPV6
00079 void RegisterMulticastFifo6(IPADDR group, uint16_t dest_port, OS_FIFO *pfifo, int interface = 0);
00080 inline void RegisterMulticastFifo(IPADDR group, uint16_t dest_port, OS_FIFO *pfifo, int interface = 0)
00081 {
00082     RegisterMulticastFifo6(group, dest_port, pfifo, interface);
00083 }
00084
00097 void UnregisterMulticastFifo6(IPADDR group, uint16_t destination_port, int interface = 0);
00098 inline void UnregisterMulticastFifo(IPADDR group, uint16_t destination_port, int interface = 0)
00099 {
00100     UnregisterMulticastFifo6(group, destination_port, interface);
00101 }
00102
00103 OS_FIFO *ListeningForGroup(const IPADDR &group);
00104 #else
00105 inline void RegisterMulticastFifo(IPADDR group, uint16_t dest_port, OS_FIFO *pfifo, int interface = 0)
00106 {
00107     RegisterMulticastFifo4(group, dest_port, pfifo, interface);
00108 }
00109
00110 inline void UnregisterMulticastFifo(IPADDR group, uint16_t destination_port, int interface = 0)
00111 {
00112     UnregisterMulticastFifo4(group, destination_port, interface);
00113 }
00114 #endif
00115
00116 #endif   // _NB_MULTICAST_H
00117
```

24.421 multihome.h File Reference

Create Multihome and VLAN Interfaces.

`#include <netinterface.h>`

Functions

- int AddVlanInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, const char *ParentName)

  Add a VLAN interface with a Parent Name.
- int AddVlanInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, InterfaceBlock &parent)

  Add a VLAN interface with a Partent InterfaceBlock reference.
- int AddVlanInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, int root_if=0)

  Add a VLAN interface with an interface number.
- int AddInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, const char *ParentName)

  Add an interface with a Parent Name.
- int AddInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, InterfaceBlock &parent)

  Add an interface with a Partent InterfaceBlock reference.
- int AddInterface (IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, int root_if=0)

  Add an interface with an interface number.

24.421.1 Detailed Description

Create Multihome and VLAN Interfaces.

24.422 multihome.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00016 #ifndef _NB_MULTI_IP_H
00017 #define _NB_MULTI_IP_H
00018
00019 #include <netinterface.h>
00020
00021 class MultiHomeInterface : public InterfaceBlock
00022 {
00023     MultiHomeInterface *pNext;
00024     static MultiHomeInterface *pHead;   // Static list of interface
00025     static bool bAfterConfig;
00026     static void FixupMultiHomeConfigTree();
00027
00028     bool bConnected;             // True if the interface is connneted
00029     InterfaceBlock *pIfParent;   // Pointer to the interface parent
00030     NBString Connector;
00031
00032     virtual void send_func(PoolPtr poolPtr);
00033     virtual void EnableMulticast(MACADR macAddress, BOOL addAddress);
00034     virtual bool LinkActive();
00035     virtual int LinkSpeed();
00036     virtual bool LinkDuplex();
00037     void CoreSetup(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, int VlanTag, bool bShowInConfig, bool
      bLetConfigModify);
00038     void FixConfig();
00039
00040   public:
00041     MultiHomeInterface(const char *name,
00042                        IPADDR4 addr,
00043                        IPADDR4 mask,
00044                        IPADDR4 gateway,
00045                        InterfaceBlock &parent,
00046                        int VlanTag = 0,
00047                        bool bShowInConfig = true,
00048                        bool bLetConfigModify = true);
00049     MultiHomeInterface(const char *name,
00050                        IPADDR4 addr,
00051                        IPADDR4 mask,
00052                        IPADDR4 gateway,
00053                        const char *Parentname,
00054                        int VlanTag = 0,
00055                        bool bShowInConfig = true,
00056                        bool bLetConfigModify = true);
00057     MultiHomeInterface(const char *name,
00058                        IPADDR4 addr,
```

```
00059                       IPADDR4 mask,
00060                       IPADDR4 gateway,
00061                       int ParentInterfaceNumber,
00062                       int VlanTag = 0,
00063                       bool bShowInConfig = true,
00064                       bool bLetConfigModify = true);
00065     operator int() const { return my_ifnum; };
00066     IPADDR4 IP() { return (IPADDR4)ip4.cur_addr.i4; };
00067
00068     friend void init();
00069 };
00070
00082 int AddVlanInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, const char
      *ParentName);
00083
00095 int AddVlanInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, InterfaceBlock
      &parent);
00096
00108 int AddVlanInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, uint16_t vlan_tag, int root_if = 0);
00109
00120 inline int AddInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, const char *ParentName)
00121 {
00122     return AddVlanInterface(addr, mask, gateway, 0, ParentName);
00123 }
00124
00135 inline int AddInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, InterfaceBlock &parent)
00136 {
00137     return AddVlanInterface(addr, mask, gateway, 0, parent);
00138 }
00139
00150 inline int AddInterface(IPADDR4 addr, IPADDR4 mask, IPADDR4 gateway, int root_if = 0)
00151 {
00152     return AddVlanInterface(addr, mask, gateway, 0, root_if);
00153 }
00154
00155 #endif
00156
```

24.423 nbprintfinternal.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NB_INTERNAL_PRINTF
00006 #define NB_INTERNAL_PRINTF
00007
00008 #include <basictypes.h>
00009 #include <stdarg.h>
00010
00011 class TCP_SOCKET;
00012 struct tlbuffer
00013 {
00014     TCP_SOCKET *ps;
00015     char tbuffer[128];
00016     uint8_t cnt;
00017 };
00018
00019 void putatchar(void *data, char c);
00020
00021 struct pfstate
00022 {
00023     int nsent;
00024     uint16_t flags;
00025     int width;
00026     int dwidth;
00027     int len;
00028 };
00029
00030 typedef int(PutCharsFunction)(void *data, const char *chars, int len);
00031 typedef int(ParsePrintfFloatFunc)(char f, PutCharsFunction *pf, void *data, double d, pfstate &pfs);
00032 extern ParsePrintfFloatFunc *pPrintfFloatFunc;
00033
00034 int TheFloatPrintf(char f, PutCharsFunction *pf, void *data, double d, pfstate &pfs);
00035
00036 int NB_internal_iprintf(PutCharsFunction *pf, void *data, const char *format, va_list arg);
00037
00038 #define NB_PRINTF_EXTEND (1)
00039 #define PRINTF_FLAG_DONE (0x0001)
00040 #define PRINTF_FLAG_LEFT (0x0002)
00041 #define PRINTF_FLAG_PLUSSIGN (0x0004)
00042 #define PRINTF_FLAG_BLANKSIGN (0x0008)
00043 #define PRINTF_FLAG_LEADZERO (0x0010)
00044 #define PRINTF_FLAG_SPECIAL (0x0020)
00045 #define PRINTF_FLAG_LOWERHEX (0x0040)
```

```
00046 #define PRINTF_FLAG_SAWDOT (0x0080)
00047 #define PRINTF_FLAG_SAWWID (0x0100)
00048 #define PRINTF_FLAG_FIRST_L (0x0200)
00049 #define PRINTF_FLAG_LONG_LONG (0x0400)
00050 #define PRINTF_FLAG_FIRST_H (0x0800)
00051 #define PRINTF_FLAG_HALF_HALF (0x1000)
00052 #define PRINTF_FLAG_WAS_NEG (0x2000)
00053 #define PRINTF_FLAG_ZERO_PREC (0x4000)
00054
00055 int decimallen(uint32_t dw);
00056 bool prespace(int width, uint32_t flags, int len, PutCharsFunction *pf, void *data, int &nsent);
00057 bool postspace(int width, uint32_t flags, int len, PutCharsFunction *pf, void *data, int &nsent);
00058 bool leadzero(int width, uint32_t flags, int len, PutCharsFunction *pf, void *data, int &nsent);
00059
00060 #endif
```

24.424 nbrtos.h File Reference

NetBurner Real-Time Operating System (NBRTOS) API.

```
#include <predef.h>
#include <constants.h>
#include <basictypes.h>
#include <nbrtoscpu.h>
```

Classes

- class TickTimeout

  TickTimeout objects are used to facilitate sequential function calls with timeout parameters that need to indexed from an initial start time, and to prevent TimeTick rollover errors.
- struct OS_SEM

  Semaphores are used to control access to shared resources or or to communicate between tasks in a multithreaded system or with interrupt service routines. Semaphores can be 0 or 1, or they can be counting semaphores that increment and decreement based on calls to Pend() and Post() functions.
- struct OS_MBOX

  Mailboxes single value storage locations used to communicate between tasks.
- struct OS_Q

  A message queue is an object that enables tasks and interrupt service routines to pend and post pointer sized messages. The pointer values typically point to some type of object or structure that contains the actual message or data. A queue functions as a fixed size First In First Out (FIFO) storage for 32-bit void pointers that can be used for communication between tasks.
- struct os_fifo_el

  OS_FIFO element definition.
- struct OS_FIFO
- struct OS_CRIT

  An OS_CRIT object is used to establish critical sections of code that can only be run by one task at a time. Tasks that try to claim a critical section which is currently claimed by another task will stop and wait for that task to release the critical section before continuing execution.
- struct OS_FLAGS

  OSFlags enables a function or task to pend on multiple flags or events.
- class OSLockObj

  A simple wrapper class that helps use OS locks effectively.
- class OSCriticalSectionObj

  A simple wrapper class that helps utilize OS_CRIT objects more effectively.
- class OSLockAndCritObj

  A simple wrapper class that helps utilize OS_CRIT objects to lock tasks and enter critical sections more effectively.
- class OSSpinCrit

  A simple wrapper class that uses an OS_CRIT object to try and claim a critical section, and will continue the attempt until it is able to do so.

- class USERCritObj

  User critial section object class.
- class NBRtosInitObj

  A simple class to derive from if you are creating tasks that are constructed at global scope and need to do RTOS initalization.

Macros

- #define **OS_STAT_RDY** 0x00

  Ready to run.
- #define **OS_STAT_MBOX** 0x01

  Pending on mailbox.
- #define **OS_STAT_SEM** 0x02

  Pending on semaphore.
- #define **OS_STAT_Q** 0x04

  Pending on queue.
- #define **OS_STAT_FIFO** 0x08

  Pending on FIFO.
- #define **OS_STAT_CRIT** 0x10

  Pending on Critical Section.
- #define **OS_STAT_DELAY** 0x20

  Reserved.
- #define **OS_STAT_RES4** 0x40

  Reserved.
- #define **OS_STAT_RES5** 0x80

  Reserved.
- #define **OS_NO_ERR** 0

  No error.
- #define **OS_TIMEOUT** 10

  Timeout.
- #define **OS_MBOX_FULL** 20

  Mailbox full.
- #define **OS_Q_FULL** 30

  Queue full.
- #define **OS_Q_EXISTS** 31

  Queue already exists.
- #define **OS_PRIO_EXIST** 40

  Task priority number already exists.
- #define **OS_PRIO_INVALID** 41

  Invalid task priority number.
- #define **OS_SEM_ERR** 50

  Semaphore error.
- #define **OS_SEM_OVF** 51

  Semaphore count overflow.
- #define **OS_CRIT_ERR** 60

  Critical section error.
- #define **OS_NO_MORE_TCB** 70

  No Task Control Blocks (TCB) available to create task.
- #define **WAIT_FOREVER** 0

  Parameter macro used for timeout parameters that have a 0 value and wait forever.
- #define OSSimpleTaskCreatewName(x, p, n)

  Simpler form of creating a new task. Will automatically allocate the default task stack size.
- #define OSSimpleTaskCreateLambda(p, n, f) LambdaTask2(p,n,[ ](void * pv)f,__COUNTER__)

  This macro functions the same as OSTaskCreatewName().

Typedefs

- typedef struct os_fifo_el OS_FIFO_EL

  OS_FIFO element definition.

Functions

- void OSFlagSet (OS_FLAGS *flags, uint32_t bits_to_set)

  This function sets the corresponding bits asserted in `bits_to_set` of an OS_FLAGS object pointed to by `*flags`.
- void OSFlagClear (OS_FLAGS *flags, uint32_t bits_to_clr)

  *This function clears the bits asserted in `bits_to_clr` of an OS_FLAGS object pointed to by *flags..*
- uint8_t OSFlagPendAny (OS_FLAGS *flags, uint32_t bit_mask, uint16_t timeout)

  This function waits a number of time ticks specified by `timeout` until any of the flags indicated by `bit_mask` are set.
- uint8_t OSFlagPendAnyNoWait (OS_FLAGS *flags, uint32_t bit_mask)

  This function immediately checks to see if any of the flag bits indicated by `bit_mask` are set; it does not wait.
- uint8_t OSFlagPendAll (OS_FLAGS *flags, uint32_t bit_mask, uint16_t timeout)

  *This function waits a number of time ticks specified by `timeout` until **all** the flags indicated by `bit_mask` are set.*
- uint8_t OSFlagPendAllNoWait (OS_FLAGS *flags, uint32_t bit_mask)

  *This function immediately checks to see if **all** the flag bits indicated by `bit_mask` are set; it does not wait.*
- uint32_t OSFlagState (OS_FLAGS *flags)

  This function returns the current values of the flags stored in the OS_FLAGS object structure.
- uint8_t OSTaskCreatewName (void(*task)(void *dptr), void *data, void *pstktop, void *pstkbot, uint8_t prio, const char *name)

  Create a new task.
- void OSTimeWaitUntil (uint32_t systemTickValue)

  Delay the task until the specified value of the system timer tick. The number of system ticks per second is defined by the constant: `TICKS_PER_SECOND` in `<nburn_install>/nbrtos/include/constants.h`. The default value is 20 ticks per second.
- void OSTimeDly (uint32_t to_count)

  Delay the task until the specified value of the system timer ticks. The number of system ticks per second is defined by the constant: `TICKS_PER_SECOND` in `<nburn_install>/nbrtos/include/constants.h`. The default value is 20 ticks per second.
- void OSTaskDelete (void)

  This function deletes the current calling task, but we do not recommend the use of this function because it can cause memory leaks.
- uint8_t OSChangePrio (uint32_t newp)

  Set the priority of the calling task.
- void OSSetName (const char *cp)

  Set the name of the calling task.
- void OSLock (void)

  Calling the OSLock function will prevent the OS from changing tasks.
- void OSUnlock (void)

  This function unlocks the OS.
- uint8_t OSSemInit (OS_SEM *psem, long value)

  Initializes a semaphore.
- uint8_t OSSemPost (OS_SEM *psem)

  *Increases the value of the semaphore by one. **Note:** If any higher priority tasks were waiting on the semaphore - it releases them.*
- uint8_t OSSemPend (OS_SEM *psem, uint16_t timeout)

  *Wait timeout ticks for the value of the semaphore to be non zero. **Note:** A timeout value of 0 (zero) waits forever.*
- uint8_t OSSemPendNoWait (OS_SEM *psem)

  OSSemPendNoWait() is identical to OSSemPend(), but it does not wait.

- uint8_t OSMboxInit (OS_MBOX *pmbox, void *msg)

  This function is used to initialize an `OS_MBOX` structure.
- uint8_t OSMboxPost (OS_MBOX *pmbox, void *msg)

  This function posts a message to a Mail box.
- void * OSMboxPend (OS_MBOX *pmbox, uint16_t timeout, uint8_t *err)

  Wait timeout ticks for some other task to post to the Mailbox.
- void * OSMboxPendNoWait (OS_MBOX *pmbox, uint8_t *err)

  *OSMboxPendNoWait() is identical to OSMboxPend(), but it does **not** wait.*
- uint8_t OSQInit (OS_Q *pq, void **start, uint8_t size)

  A queue functions as a fixed size FIFO for communication between tasks. This function initializes an `OS_Q` structure.
- uint8_t OSQPost (OS_Q *pq, void *msg)

  This function posts a message to a Queue.
- uint8_t OSQPostFirst (OS_Q *pq, void *msg)

  This function posts a message like `OSQPost`, but posts the message at the head of the queue.
- uint8_t OSQPostUnique (OS_Q *pq, void *msg)

  This function posts a message like `OSQPost`, but only if the message isn't already in the queue The function performs a brute force check to see if the message is already in the queue.
- uint8_t OSQPostUniqueFirst (OS_Q *pq, void *msg)

  This function posts a message like `OSQPostFirst`, but only if the message isn't already in the queue The function performs a brute force check to see if the message is already in the queue.
- void * OSQPend (OS_Q *pq, uint16_t timeout, uint8_t *err)

  Wait timeout ticks for another task to post to the queue.
- void * OSQPendNoWait (OS_Q *pq, uint8_t *err)

  OSQPendNoWait() is identical to the OSQPend() function but it does not wait.
- uint8_t OSFifoInit (OS_FIFO *pFifo)

  Initialize a FIFO, which is used to pass structures from one task to another.
- uint8_t OSFifoPost (OS_FIFO *pFifo, OS_FIFO_EL *pToPost)

  This function posts to a FIFO.
- uint8_t OSFifoPostFirst (OS_FIFO *pFifo, OS_FIFO_EL *pToPost)

  This function is identical to OSFifoPost(), but the element posted is put at the beginning of the FIFO list.
- OS_FIFO_EL * OSFifoPend (OS_FIFO *pFifo, uint16_t timeout)

  This function pends on a FIFO.
- OS_FIFO_EL * OSFifoPendNoWait (OS_FIFO *pFifo)

  *This function is identical to the OSFifoPen() function, but it does **not** wait.*
- uint8_t OSCritInit (OS_CRIT *pCrit)

  This function initializes the critical section.
- uint8_t OSCritEnter (OS_CRIT *pCrit, uint16_t timeout)

  This function tries to enter or claim the critical section.
- uint8_t OSCritEnterNoWait (OS_CRIT *pCrit)

  This function tries to enter or claim the critical section.
- uint8_t OSCritLeave (OS_CRIT *pCrit)

  This function releases the critical section.
- uint8_t **OSTaskID** (void)

  Returns the current task's priority.
- const char * **OSTaskName** ()

  Returns the current task's name.
- void OSChangeTaskDly (uint16_t task_prio, uint32_t to_count)

  This function allows the User to modify the timeout delay for a task that is waiting.
- void **OSDumpStack** (void)

  Dump the task stack to the stdout.
- void OSDumpTCBStacks (void)

This function dumps information about the UCOS stacks and tasks to stdout. This function is useful for debugging.

- void OSDumpTasks (void)

  Dump the state and call stack for every task to stdout. This function is useful for debugging.

- void OSStartTaskDumper (uint8_t prio, uint32_t reportInterval)

  This function creates a task that calls OSDumpTasks() at the specified system time tick interval. The task is intended for use when debugging run status of multiple tasks.

- void ShowTaskList (void)

  This functions dumps the current RTOS task states to stdio.

24.424.1 Detailed Description

NetBurner Real-Time Operating System (NBRTOS) API.

24.425 nbrtos.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00016 #ifndef _NBRTOS_H
00017 #define _NBRTOS_H
00018
00019 // NB Definitions
00020 #include <predef.h>
00021
00022 // NB Constants
00023 #include <constants.h>
00024
00025 // NB Libs
00026 #include <basictypes.h>
00027 #include <nbrtoscpu.h>
00028 #include <predef.h>
00029
00030 /*********************************************************
00031  *                          NBRTOS.H
00032  *                   SYSTEM DECLARATIONS
00033  *********************************************************
00034  */
00035 #define OS_LO_PRIO 63 /*IDLE task priority      */
00036
00043 #define OS_STAT_RDY 0x00
00044 #define OS_STAT_MBOX 0x01
00045 #define OS_STAT_SEM 0x02
00046 #define OS_STAT_Q 0x04
00047 #define OS_STAT_FIFO 0x08
00048 #define OS_STAT_CRIT 0x10
00049 #define OS_STAT_DELAY 0x20
00050 #define OS_STAT_RES4 0x40
00051 #define OS_STAT_RES5 0x80
00059 #define OS_NO_ERR 0
00060 #define OS_TIMEOUT 10
00061 #define OS_MBOX_FULL 20
00062 #define OS_Q_FULL 30
00063 #define OS_Q_EXISTS 31
00064 #define OS_PRIO_EXIST 40
00065 #define OS_PRIO_INVALID 41
00066 #define OS_SEM_ERR 50
00067 #define OS_SEM_OVF 51
00068 #define OS_CRIT_ERR 60
00069 #define OS_NO_MORE_TCB 70
00072 #define WAIT_FOREVER 0
00073
00074 typedef volatile uint32_t tick_t;
00075
00077 // GLOBAL VARIABLES
00078 //
00079 extern vuint32_t Secs;             // Number of seconds since system start
00080 extern volatile tick_t TimeTick;   // Number of time ticks since system start
00081
00082 // Is test_time later than now (TimeTick)
00083 inline bool IsTickLater(uint32_t test_time)
00084 {
00085     return ((int)(TimeTick - test_time) < 0);
00086 }
00087
00088 // Is test_time NoworEarlier than TimeTick
```

```
00089 inline bool IsTickNowOrEarlier(uint32_t test_time)
00090 {
00091     return ((int)(TimeTick - test_time) >= 0);
00092 }
00093
00094 // Compare two timetick values
00095 inline bool Is2ndTickEarlier(uint32_t t1, uint32_t t2)
00096 {
00097     return (((int)(t1 - t2)) > 0);
00098 }
00099
00100 // Compare two timetick values
00101 inline bool Is2ndTickNowOrEarlier(uint32_t t1, uint32_t t2)
00102 {
00103     return (((int)(t1 - t2)) >= 0);
00104 }
00105
00106 class TickTimeout;
00107 struct RawTickTimeout_t
00108 {
00109     tick_t expiration;
00110
00111     inline bool expired() const { return expiration ? (((int)(expiration - TimeTick)) <= 0) : false; }
00112     inline bool expired() volatile { return expiration ? (((int)(expiration - TimeTick)) <= 0) :
      false; }
00113     inline operator bool() const { return !expired(); }
00114     const RawTickTimeout_t &operator=(const TickTimeout &rhs);
00115     volatile RawTickTimeout_t &operator=(const TickTimeout &rhs) volatile;
00116
00117     inline bool operator<(const RawTickTimeout_t &later)
00118     {
00119         if (!expiration)
00120             return false;
00121         if (!later.expiration)
00122             return true;
00123         return (((int)(expiration - later.expiration)) <= 0);
00124     }
00125
00126     inline bool operator<(tick_t later)
00127     {
00128         if (!expiration)
00129             return false;
00130         return (((int)(expiration - later)) <= 0);
00131     }
00132
00133     inline tick_t operator-(const tick_t &tick) { return expiration - tick; }
00134     inline tick_t operator-(const tick_t &tick) const { return expiration - tick; }
00135     inline tick_t operator-(const tick_t &tick) volatile { return expiration - tick; }
00136 };
00137
00138 inline bool operator==(const RawTickTimeout_t &lhs, const int &rhs)
00139 {
00140     return lhs.expiration == (tick_t)rhs;
00141 }
00142
00143 inline bool operator==(const volatile RawTickTimeout_t &lhs, const int &rhs)
00144 {
00145     return lhs.expiration == (tick_t)rhs;
00146 }
00147
00153 class TickTimeout
00154 {
00155     RawTickTimeout_t raw;
00156
00157     void set(uint32_t timeout)
00158     {
00159         if (!timeout) { raw.expiration = 0; }
00160         else
00161         {
00162             raw.expiration = TimeTick + (timeout & 0x7FFFFFFF);
00163             // A 1 tick delay extension is introduced on TimeTick overflow
00164             // in order to allow for infinite delays to be indicated with an
00165             // expiration of zero
00166             if (timeout && !raw.expiration) { raw.expiration = 1; }
00167         }
00168     }
00169
00170   public:
00171     class uint32_nonboolean_t
00172     {
00173         uint32_t value;
00174
00175       public:
00176         uint32_nonboolean_t(uint32_t val) : value(val) {}
00177         inline explicit operator bool() { return (bool)value; }
00178         inline operator uint32_t() { return value; }
00179     };
```

```
00180
00181     explicit TickTimeout() { set(0); }
00182
00188     TickTimeout(uint32_t timeout) { set(timeout); }
00189     TickTimeout(uint16_t timeout) { set(timeout); }
00190     TickTimeout(int      timeout) { set(timeout); }
00191
00192
00198     inline uint32_t val() const
00199     {
00200         if (!raw.expiration) { return raw.expiration; }
00201         int ret = raw.expiration - TimeTick;
00202         // Prevent passing an infinite or otherwise bogus timeout in a tick race
00203         return (ret > 0) ? ret : 1;
00204     }
00205
00212     inline bool expired() const { return raw.expired(); }
00213
00225     inline operator bool() const { return !expired(); }
00226
00227     inline operator uint32_t() const { return val(); }
00228
00229     inline operator uint16_t() const
00230     {
00231         uint32_t ret = val();
00232         return ret > 0xFFFF ? 0xFFFE : ret;
00233     }
00234
00235     inline TickTimeout &operator=(const TickTimeout &rhs)
00236     {
00237         raw.expiration = rhs.raw.expiration;
00238         return *this;
00239     }
00240     inline TickTimeout &operator=(uint32_t val)
00241     {
00242         set(val);
00243         return *this;
00244     }
00245     inline bool operator<(TickTimeout later)
00246     {
00247         return raw < later.raw;
00248     }
00249     inline bool operator<(tick_t later)
00250     {
00251         return raw < later;
00252     }
00253
00259     inline void SetUntil(uint32_t when) {raw.expiration = when; }
00260
00261     friend void OSTimeWaitUntil(uint32_t systemTickValue);
00262
00263     friend class RawTickTimeout_t;
00264 };
00265
00266 inline const RawTickTimeout_t &RawTickTimeout_t::operator=(const TickTimeout &rhs)
00267 {
00268     expiration = rhs.raw.expiration;
00269     return *this;
00270 }
00271
00272 inline volatile RawTickTimeout_t &RawTickTimeout_t::operator=(const TickTimeout &rhs) volatile
00273 {
00274     expiration = rhs.raw.expiration;
00275     return *this;
00276 }
00277
00278 // The following class holds a list of tasks
00279
00280 class task_bit_list
00281 {
00282    public:
00283     volatile uint32_t OSTbl[TASK_TABLE_SIZE];
00284
00285     // task_bit_list();
00286     void Init() volatile;
00287     void Copy(volatile task_bit_list &rhs)
00288     {
00289         for (int i = 0; i < TASK_TABLE_SIZE; i++)
00290         {
00291             OSTbl[i] = rhs.OSTbl[i];
00292         }
00293     }
00294
00295     // The following functions are all guaranteed to be atomic
00296     void set(int set_num) volatile;
00297     void clr(int clr_num) volatile;
00298
```

```
00299     // The following functions return 0 if no bits are set.
00300     uint32_t gethigh() volatile;
00301     uint32_t get_high_and_clear() volatile;
00302
00303     inline bool isSet(int num) volatile const { return (OSTbl[num / 32] & (0x80000000 » (num % 32)));
      }
00304 };
00305
00306 class OS_TCB;
00307
00308 // The common element of all the task objects
00309 class OS_TASK_DLY_OBJ
00310 {
00311     task_bit_list tasks_waiting;
00312
00313   public:
00314     OS_TASK_DLY_OBJ();
00315     void Init();
00316
00317     // Returns true if woken up, returns false if timed out
00318     bool Wait_when(uint8_t StatReason, TickTimeout &timeout);
00319
00320     inline bool Wait(uint8_t StatReason, uint32_t to_count)
00321     {
00322         TickTimeout to_when(to_count);
00323         return Wait_when(StatReason, to_when);
00324     }
00325
00326     // This releases the highest priority task waiting on this object.
00327     void MakeHighTaskWaitingReady(uint8_t StatReason);
00328     friend class OS_TCB;
00329 } __attribute__((packed));
00330
00331 // class OS_TCB;//forward
00332
00333 class OS_TCB : public cpu_tcb
00334 {
00335   public:
00336     uint8_t OSTCBStat;      // Holds the current status of the TCB
00337     uint8_t OSTCBResult;    // Basically holds the timeout or not flag when woken.
00338     uint8_t OSTCBPrio;      // Index to prio table.... not sure if we need to keep this.
00339     uint8_t pad;            // Make 32 bit aligned.
00340
00341     RawTickTimeout_t OSTCBDly_when;   // When to release timeout the task, 0= never.
00342
00343     const char *pOSTCBName;
00344     OS_TCB *OSTCBNext;
00345     OS_TCB *OSTCBPrev;
00346
00347 #ifdef NBRTOS_PRIO_PROMOTION
00348     // These pointers are for handling Priority Promotion when OS_CRITs create
00349     // an inversion situation
00350     volatile OS_TCB *pPromotedTo;
00351     volatile OS_TCB *pDisplacedBy;
00352     OS_TASK_DLY_OBJ *pWaiting;
00353     uint32_t displacedByOrigPrio;
00354
00355     void PromoteToCurPrio() volatile;
00356     void Demote() volatile;
00357 #endif
00358
00359     static volatile OS_TCB *GetCur();
00360
00361 #ifdef NBRTOS_TIME
00362     unsigned long switchTimeTick;
00363     unsigned long switchTimeFraction;
00364     unsigned long runningTime;
00365     unsigned long runningTimeFraction;
00366 #endif
00367     //  OS_TCB(); //Zero everything
00368     void Init();
00369
00370     // Set up TCB assuming that the STACK has already been setup and properly initalized.
00371     bool Init(uint8_t prio, void *pActualTop, void *pstk, long *pbot, const char *name);
00372 };
00373
00374 /* Forward Declaration */
00375 struct OS_CRIT;
00376
00382 struct OS_SEM : public OS_TASK_DLY_OBJ
00383 {
00384     volatile uint32_t OSSemCnt;
00385     volatile uint32_t OSSemUsed;
00386
00387   private:
00388     bool Claim();
00389
```

```
00390    public:
00398     inline OS_SEM(int32_t cnt = 0) : OS_TASK_DLY_OBJ() { Init(cnt); }
00399
00409     uint8_t Init(int32_t cnt = 0);
00410
00420     uint8_t Post();
00421
00431     inline uint8_t Pend(uint32_t timeoutTicks = WAIT_FOREVER){TickTimeout tt(timeoutTicks); return
      Pend(tt);};
00432
00442     inline uint8_t PendUntil(uint32_t timeout_time){TickTimeout tt(0); tt.SetUntil(timeout_time);
      return Pend(tt);};
00443
00453     uint8_t Pend(TickTimeout &t);
00454
00464     uint8_t PendNoWait();
00465
00475     inline uint32_t Avail() { uint32_t v; USER_ENTER_CRITICAL();  v=OSSemCnt-OSSemUsed;
      USER_EXIT_CRITICAL(); return v;};
00476
00477 } __attribute__((packed));
00478
00493 struct OS_MBOX : public OS_TASK_DLY_OBJ
00494 {
00495     void *OSMboxMsg;
00496     bool OSMboxDataAvail;
00497
00498     bool Claim(void *&Result);
00499
00500    public:
00506     inline OS_MBOX() : OS_TASK_DLY_OBJ() { Init(); }
00507
00515     inline OS_MBOX(void *msg) : OS_TASK_DLY_OBJ() { Init(msg); }
00516
00526     uint8_t Init(void *msg = NULL);
00527
00538     uint8_t Post(void *msg);
00539
00552     inline void *Pend(uint32_t timeoutTicks, uint8_t &result){TickTimeout tt(timeoutTicks); return
      Pend(tt,result);};
00553
00566     void *Pend(TickTimeout &t, uint8_t &result);
00567
00578     inline void *Pend(uint32_t timeoutTicks = WAIT_FOREVER)
00579     {
00580         uint8_t unused;
00581         return Pend(timeoutTicks, unused);
00582     };
00583
00596     inline void *PendUntil(uint32_t timeoutTime, uint8_t &result){TickTimeout tt(0);
      tt.SetUntil(timeoutTime); return Pend(tt,result);};
00597
00609     void *PendNoWait(uint8_t &result);
00610
00619     inline void *PendNoWait()
00620     {
00621         uint8_t unused;
00622         return PendNoWait(unused);
00623     };
00624 };
00625
00626
00627 template<typename T>
00628 class TEMPL_MBOX
00629 {
00630    protected:
00631     OS_MBOX m_mbox;
00632
00633    public:
00634     TEMPL_MBOX(const T *msg) : m_mbox((void *)msg) {}
00635     inline uint8_t Init(const T *msg) { return m_mbox.Init((void *)msg); }
00636     inline uint8_t Post(const T *msg) { return m_mbox.Post((void *)msg); }
00637
00638     inline T *Pend(uint32_t timeoutTicks, uint8_t &result) { return static_cast<T
      *>(m_mbox.Pend(timeoutTicks, result)); };
00639
00640     inline T *PendNoWait(uint8_t &result) { return static_cast<T *>(m_mbox.PendNoWait(result)); };
00641
00642     inline T *Pend(uint32_t timeoutTicks = WAIT_FOREVER) { return static_cast<T
      *>(m_mbox.Pend(timeoutTicks)); };
00643
00644     inline T *PendNoWait() { return static_cast<T *>(m_mbox.PendNoWait()); };
00645 };
00646
00647
00668 struct OS_Q : public OS_TASK_DLY_OBJ
00669 {
```

```
00670     void **OSQStart;
00671     void **OSQEnd;
00672     void **OSQIn;
00673     void **OSQOut;
00674     uint8_t OSQSize;
00675     uint8_t OSQEntries;
00676
00677     bool Claim(void *&Result);
00678
00679   public:
00695     inline OS_Q(void **pQueueStorage, uint8_t size) : OS_TASK_DLY_OBJ() { Init(pQueueStorage, size); }
00696
00705     uint8_t Init(void **pQueueStorage, uint8_t size);
00706
00717     uint8_t Post(void *pItem);
00718
00729     uint8_t PostFirst(void *pItem);
00730
00743     uint8_t PostUnique(void *pItem);
00744
00757     uint8_t PostUniqueFirst(void *msg);
00758
00771     inline void *Pend(uint32_t timeoutTicks, uint8_t &result){TickTimeout tt(timeoutTicks); return
      Pend(tt,result);};
00772
00785     void *Pend(TickTimeout &t, uint8_t &result);
00786
00797     inline void *Pend(uint32_t timeoutTicks = WAIT_FOREVER)
00798     {
00799         uint8_t unused;
00800         return Pend(timeoutTicks, unused);
00801     };
00802
00815     inline void *PendUntil(uint32_t timeoutTime, uint8_t &result){TickTimeout tt(0);
      tt.SetUntil(timeoutTime); return Pend(tt,result);};
00816
00828     void *PendNoWait(uint8_t &result);
00829
00838     inline void *PendNoWait()
00839     {
00840         uint8_t unused;
00841         return PendNoWait(unused);
00842     };
00843 };
00844
00845 template<typename T>
00846 struct TEMPL_Q
00847 {
00848   protected:
00849     OS_Q m_q;
00850
00851   public:
00852     TEMPL_Q() {}
00853     TEMPL_Q(T **pQueueStorage, uint8_t size)
00854         : m_q((void**)pQueueStorage,size) { }
00855     uint8_t Init(T **pQueueStorage, uint8_t size) { return m_q.Init((void **)pQueueStorage, size); }
00856     uint8_t Post(T *item) { return m_q.Post((void *)item); }
00857     uint8_t PostFirst(T *item) { return m_q.PostFirst((void *)item); };
00858     uint8_t PostUnique(T *item) { return m_q.PostUnique((void *)item); };
00859     uint8_t PostUniqueFirst(T *item) { return m_q.PostUniqueFirst((void *)item); };
00860
00861     inline T *Pend(uint32_t timeoutTicks, uint8_t &result) { return static_cast<T
      *>(m_q.Pend(timeoutTicks, result)); };
00862
00863     inline T *Pend(uint32_t timeoutTicks = WAIT_FOREVER) { return static_cast<T
      *>(m_q.Pend(timeoutTicks)); };
00864
00865     inline T *PendNoWait() { return static_cast<T *>(m_q.PendNoWait()); };
00866
00867     inline T *PendNoWait(uint8_t &result) { return static_cast<T *>(m_q.PendNoWait(result)); };
00868 };
00869
00876 typedef struct os_fifo_el
00877 {
00883     union
00884     {
00885         struct os_fifo_el *pNextFifo_El;
00886         puint8_t pAsBytePtr;
00887     };
00888 } OS_FIFO_EL;
00889
00899 struct OS_FIFO : public OS_TASK_DLY_OBJ
00900 {
00901     OS_FIFO_EL *pHead;
00902     OS_FIFO_EL *pTail;
00903
00904     bool Claim(volatile OS_FIFO_EL *&pToRet);
```

```
00905
00906    public:
00912     inline OS_FIFO() : OS_TASK_DLY_OBJ() { Init(); }
00913
00921     uint8_t Init();
00922
00934     uint8_t Post(OS_FIFO_EL *pToPost);
00935
00947     uint8_t PostFirst(OS_FIFO_EL *pToPost);
00948
00961     inline OS_FIFO_EL *Pend(uint32_t timeoutTicks, uint8_t &result){TickTimeout tt(timeoutTicks);
      return Pend(tt,result);};
00962
00975     OS_FIFO_EL *Pend(TickTimeout &t, uint8_t &result);
00976
00987     inline OS_FIFO_EL *Pend(uint32_t timeoutTicks = WAIT_FOREVER)
00988     {
00989         uint8_t unused;
00990         return Pend(timeoutTicks, unused);
00991     };
00992
01005     inline OS_FIFO_EL *PendUntil(uint32_t timeoutTime, uint8_t &result){TickTimeout tt(0);
      tt.SetUntil(timeoutTime); return Pend(tt,result);};
01006
01018     OS_FIFO_EL *PendNoWait(uint8_t &result);
01019
01020
01029     inline OS_FIFO_EL *PendNoWait()
01030     {
01031         uint8_t unused;
01032         return PendNoWait(unused);
01033     };
01034 };
01035
01036
01037
01038 template<typename T>
01039 struct TEMPL_FIFO
01040 {
01041    protected:
01042     OS_FIFO m_fifo;
01043
01044    public:
01045     TEMPL_FIFO() { m_fifo.Init(); }
01046     uint8_t Init() { return m_fifo.Init(); }
01047     uint8_t Post(T *item) { return m_fifo.Post((OS_FIFO_EL *)item); }
01048     uint8_t PostFirst(T *item) { return m_fifo.PostFirst((OS_FIFO_EL *)item); };
01049
01050     inline T *Pend(uint32_t timeoutTicks, uint8_t &result) { return static_cast<T
      *>(m_fifo.Pend(timeoutTicks, result)); };
01051
01052     inline T *Pend(uint32_t timeoutTicks = WAIT_FOREVER) { return static_cast<T
      *>(m_fifo.Pend(timeoutTicks)); };
01053
01054     inline T *PendNoWait() { return static_cast<T *>(m_fifo.PendNoWait()); };
01055
01056     inline T *PendNoWait(uint8_t &result) { return static_cast<T *>(m_fifo.PendNoWait(result)); };
01057
01058     inline OS_FIFO *GetRawFIFO() { return &m_fifo; }
01059 };
01060
01061 /*
01062  * Critical Section DATA STRUCTURES
01063  *
01064  * Added by PTB 5/09/99
01065  * Modified by DEC 3/21/16
01066  */
01067
01068 class BufferCriticalLock;
01069 class fifo_buffer_storage;
01070
01076 struct OS_CRIT : public OS_TASK_DLY_OBJ
01077 {
01078     OS_TCB *OSCritOwnerTCB;
01079     uint32_t OSCritDepthCount;
01080
01081     bool Claim();
01082
01083    public:
01089     OS_CRIT() { Init(); }
01090
01098     uint8_t Init();
01099
01115     uint8_t LockAndEnter(uint32_t timeoutTicks = WAIT_FOREVER);
01116
01131     uint8_t Enter(TickTimeout &t);
01132
```

```
01147     inline uint8_t Enter(uint32_t timeoutTicks = WAIT_FOREVER) {TickTimeout tt(timeoutTicks); return
      Enter(tt);};
01148
01160     uint8_t EnterNoWait();
01161
01173     uint8_t Leave();
01174
01185     uint8_t LeaveAndUnlock();
01186
01187     friend class BufferCriticalLock;
01188     friend class fifo_buffer_storage;
01189
01195     bool UsedFromISR() { return OSCritOwnerTCB == (OS_TCB *)0xFFFFFFFF; }
01196
01204     void SetUseFromISR(bool enableFromISR)
01205     {
01206         if (enableFromISR) { OSCritOwnerTCB = (OS_TCB *)0xFFFFFFFF; }
01207         else if (UsedFromISR())
01208         {
01209             OSCritOwnerTCB = NULL;
01210         }
01211     }
01212
01216     bool OwnedByCurTask();
01217
01223     uint32_t CurDepth() { return OSCritDepthCount; }
01224
01228     friend void ForceReboot(bool fromIRQ);
01229 } __attribute__((packed));
01230
01231
01232
01233 struct OS_FLAGS_WAIT;
01234
01245 struct OS_FLAGS
01246 {
01247    protected:
01248     vuint32_t  m_current_flags;
01249     void *m_pWaitinglist;
01250     void TestFlags();
01251     void AddOFW(OS_FLAGS_WAIT *ofw);
01252     void RemoveOFW(OS_FLAGS_WAIT *ofw);
01253
01254    public:
01260     OS_FLAGS();
01261
01267     void Init();
01268
01276     void Set(uint32_t bits_to_set);
01277
01285     void Clear(uint32_t bits_to_clr);
01286
01287
01295     void Write(uint32_t bits_to_force);
01296
01304     uint32_t State();
01305
01320     uint8_t PendAny(uint32_t bit_mask, uint16_t timeout = WAIT_FOREVER){TickTimeout tt(timeout);
      return PendAny(bit_mask,tt);}
01321
01336     uint8_t PendAny(uint32_t bit_mask, TickTimeout & timeout);
01337
01338
01350     uint8_t PendAnyUntil(uint32_t bit_mask, uint32_t end_time){TickTimeout tt(0);
      tt.SetUntil(end_time); return PendAny(bit_mask,tt);}
01351
01352
01362     uint8_t PendAnyNoWait(uint32_t bit_mask);
01363
01377     uint8_t PendAll(uint32_t bit_mask, uint16_t timeout = WAIT_FOREVER){TickTimeout tt(timeout);
      return PendAll(bit_mask,tt);}
01378
01392     uint8_t PendAll(uint32_t bit_mask,TickTimeout & timeout);
01393
01405     uint8_t PendAllUntil(uint32_t bit_mask, uint32_t end_time){TickTimeout tt(0);
      tt.SetUntil(end_time); return PendAll(bit_mask,tt);}
01406
01407
01418     uint8_t PendAllNoWait(uint32_t bit_mask);
01419
01420 };
01421
01422
01423 /* Create and initialize an OS flags object
01424 This function must be called before you use an OS_FLAGS object.
01425 */
01426 [[deprecated]] inline void OSFlagCreate(OS_FLAGS *pf)
```

```
01427 {
01428     pf->Init();
01429 };
01430
01441 [[deprecated]] inline void OSFlagSet(OS_FLAGS *flags, uint32_t bits_to_set)
01442 {
01443     flags->Set(bits_to_set);
01444 };
01445
01456 [[deprecated]] inline void OSFlagClear(OS_FLAGS *flags, uint32_t bits_to_clr)
01457 {
01458     flags->Clear(bits_to_clr);
01459 };
01460
01475 [[deprecated]] inline uint8_t OSFlagPendAny(OS_FLAGS *flags, uint32_t bit_mask, uint16_t timeout)
01476 {
01477     return flags->PendAny(bit_mask, timeout);
01478 };
01479
01493 [[deprecated]] inline uint8_t OSFlagPendAnyNoWait(OS_FLAGS *flags, uint32_t bit_mask)
01494 {
01495     return flags->PendAnyNoWait(bit_mask);
01496 };
01497
01512 [[deprecated]] inline uint8_t OSFlagPendAll(OS_FLAGS *flags, uint32_t bit_mask, uint16_t timeout)
01513 {
01514     return flags->PendAll(bit_mask, timeout);
01515 };
01516
01530 [[deprecated]] inline uint8_t OSFlagPendAllNoWait(OS_FLAGS *flags, uint32_t bit_mask)
01531 {
01532     return flags->PendAllNoWait(bit_mask);
01533 };
01534
01546 [[deprecated]] inline uint32_t OSFlagState(OS_FLAGS *flags)
01547 {
01548     return flags->State();
01549 };
01550
01551 /*
01552  ************************************************************
01553  *                 NBRTOS GLOBAL VARIABLES
01554  ************************************************************
01555  */
01556 // Pointers to each of the OS_TCB structure sorted by priority
01557 extern volatile OS_TCB *OSTCBPrioTbl[OS_MAX_PRIOS];
01558
01559 // taks bits for what tasks are ready to run
01560 extern volatile task_bit_list OSTaskReadyList;
01561 #ifdef NBRTOS_PRIO_PROMOTION
01562 extern volatile task_bit_list OSInUsePrioList FAST_SYS_VAR;
01563 extern volatile task_bit_list OSActivePrioList FAST_SYS_VAR;
01564 #endif
01565
01566 extern OS_TCB OSTCBTbl[OS_MAX_TASKS];   // All the possible TCB's
01567
01568 extern OS_TCB *pOSActiveTCBList;   // Linked list of activ TCB's
01569
01570 // One of the following two variables wsill go away....
01571 extern volatile uint32_t nPrioOfCurTask;     // Current task priority number
01572 extern volatile uint32_t nPrioOfHighReady;   // highest task ready to run
01573 extern volatile OS_TCB *OSTCBCur;
01574 extern volatile OS_TCB *OSTCBHighRdy;
01575
01576 extern OS_TCB *pOSTCBFreeList;   // List of unused TCB's
01577
01578 extern volatile uint32_t OSIntNesting;
01579 extern volatile uint32_t OSLockNesting;
01580 extern volatile uint32_t OSISRLevel32;
01581
01582 extern volatile bool OSRunning;
01583
01584 // So non-recompilable files can reference the right struct size
01585 extern unsigned long OSTcbStructSize;
01586
01587 #ifdef NBRTOS_TASK_LOG
01588 extern void (*pTaskLogger)(uint8_t nextPrio);
01589 #endif
01590 /*
01591 Removed
01592    extern volatile BOOLEAN OSShowTasksOnLeds;
01593 */
01594
01595 /*
01596 ************************************************************
01597 *                  NBRTOS FUNCTION PROTOTYPES
01598 ************************************************************
```

```
01599 */
01600 void OSInit(uint8_t maxtasks);
01601 void OSStart(void);
01602 void OSCreateIdleTask();
01603
01632 uint8_t OSTaskCreatewName(void (*task)(void *dptr),    // Function to call
01633                           void *data,                  // Data to pass as a parameter
01634                           void *pstktop,               // Stack top
01635                           void *pstkbot,               // Stack bottom
01636                           uint8_t prio,                // current priority
01637                           const char *name             // task name
01638 );
01639
01650 #define OSSimpleTaskCreatewName(x, p, n)
      \
01651     {
      \
01652         static uint32_t func_##x_Stk[USER_TASK_STK_SIZE] __attribute__((aligned(4)));
      \
01653         OSTaskCreatewName(x, NULL, (void *)&func_##x_Stk[USER_TASK_STK_SIZE], (void *)func_##x_Stk, p,
      n); \
01654     }
01655
01656 #define OSSimpleTaskCreatewNameSRAM(x, p, n)
      \
01657     {
      \
01658         static uint32_t func_##x_Stk[USER_TASK_STK_SIZE] __attribute__((aligned(4))) FAST_USER_STK;
      \
01659         OSTaskCreatewName(x, NULL, (void *)&func_##x_Stk[USER_TASK_STK_SIZE], (void *)func_##x_Stk, p,
      n); \
01660     }
01661
01662 /* Helper macro*/
01663 #define LambdaTask2(p,n,f,vn) static uint32_t func_##vn_Stk[USER_TASK_STK_SIZE]
      __attribute__((aligned(4)));OSTaskCreatewName(f, NULL, (void *)&func_##vn_Stk[USER_TASK_STK_SIZE],
      (void *)func_##vn_Stk, p, n)
01664
01680 #define OSSimpleTaskCreateLambda(p,n,f) LambdaTask2(p,n,[]( void * pv)f,__COUNTER__)
01681
01682
01702 void OSTimeWaitUntil(uint32_t systemTickValue);
01703
01716 inline void OSTimeDly(uint32_t to_count)
01717 {
01718     uint32_t to_when = to_count + TimeTick;
01719     if (!to_when) to_when++;
01720     OSTimeWaitUntil(to_when);
01721 };
01722
01723 // void OSIntEnter( void );
01724
01725 extern "C"
01726 {
01727     void OSIntExit(void);
01728     void OSCtxSw(void);
01729     void OSTickISR(void);
01730     void OSStartHighRdy(void);
01731     void OSSetupVBR(void);
01732     void OSSched(void);
01733     void OSTimeTick(void);
01734
01745     void OSTaskDelete(void);
01746 }
01747
01748 OS_TCB *OSTCBGetFree(void);
01749
01768 uint8_t OSChangePrio(uint32_t newp);
01769
01776 void OSSetName(const char *cp);
01777
01791 void OSLock(void);
01792
01800 void OSUnlock(void);
01801
01816 [[deprecated]] inline uint8_t OSSemInit(OS_SEM *psem, long value)
01817 {
01818     return (psem != nullptr) ? psem->Init(value) : OS_CRIT_ERR;
01819 }
01820
01834 [[deprecated]] inline uint8_t OSSemPost(OS_SEM *psem)
01835 {
01836     return psem->Post();
01837 }
01838
01853 [[deprecated]] inline uint8_t OSSemPend(OS_SEM *psem, uint16_t timeout)
01854 {
```

```
01855     return psem->Pend(timeout);
01856 }
01857
01870 [[deprecated]] inline uint8_t OSSemPendNoWait(OS_SEM *psem)
01871 {
01872     return psem->PendNoWait();
01873 }
01874
01888 [[deprecated]] inline uint8_t OSMboxInit(OS_MBOX *pmbox, void *msg)
01889 {
01890     return (pmbox != nullptr) ? pmbox->Init(msg) : OS_CRIT_ERR;
01891 }
01892
01906 [[deprecated]] inline uint8_t OSMboxPost(OS_MBOX *pmbox, void *msg)
01907 {
01908     return pmbox->Post(msg);
01909 }
01910
01922 [[deprecated]] inline void *OSMboxPend(OS_MBOX *pmbox, uint16_t timeout, uint8_t *err)
01923 {
01924     return pmbox->Pend(timeout, *err);
01925 }
01926
01937 [[deprecated]] inline void *OSMboxPendNoWait(OS_MBOX *pmbox, uint8_t *err)
01938 {
01939     return pmbox->PendNoWait(*err);
01940 }
01941
01956 [[deprecated]] inline uint8_t OSQInit(OS_Q *pq, void **start, uint8_t size)
01957 {
01958     return (pq != nullptr) ? pq->Init(start, size) : OS_CRIT_ERR;
01959 }
01960
01974 [[deprecated]] inline uint8_t OSQPost(OS_Q *pq, void *msg)
01975 {
01976     return pq->Post(msg);
01977 }
01978
01992 [[deprecated]] inline uint8_t OSQPostFirst(OS_Q *pq, void *msg)
01993 {
01994     return pq->PostFirst(msg);
01995 }
01996
02012 [[deprecated]] inline uint8_t OSQPostUnique(OS_Q *pq, void *msg)
02013 {
02014     return pq->PostUnique(msg);
02015 }
02016
02032 [[deprecated]] inline uint8_t OSQPostUniqueFirst(OS_Q *pq, void *msg)
02033 {
02034     return pq->PostUniqueFirst(msg);
02035 }
02036
02048 [[deprecated]] inline void *OSQPend(OS_Q *pq, uint16_t timeout, uint8_t *err)
02049 {
02050     return pq->Pend(timeout, *err);
02051 }
02052
02063 [[deprecated]] inline void *OSQPendNoWait(OS_Q *pq, uint8_t *err)
02064 {
02065     return pq->PendNoWait(*err);
02066 }
02067
02080 [[deprecated]] inline uint8_t OSFifoInit(OS_FIFO *pFifo)
02081 {
02082     return (pFifo != nullptr) ? pFifo->Init() : OS_CRIT_ERR;
02083 }
02084
02097 [[deprecated]] inline uint8_t OSFifoPost(OS_FIFO *pFifo, OS_FIFO_EL *pToPost)
02098 {
02099     return pFifo->Post(pToPost);
02100 }
02113 [[deprecated]] inline uint8_t OSFifoPostFirst(OS_FIFO *pFifo, OS_FIFO_EL *pToPost)
02114 {
02115     return pFifo->PostFirst(pToPost);
02116 };
02117
02128 [[deprecated]] inline OS_FIFO_EL *OSFifoPend(OS_FIFO *pFifo, uint16_t timeout)
02129 {
02130     return pFifo->Pend(timeout);
02131 };
02132
02142 [[deprecated]] inline OS_FIFO_EL *OSFifoPendNoWait(OS_FIFO *pFifo)
02143 {
02144     return pFifo->PendNoWait();
02145     ;
02146 }
```

```
02147
02159 [[deprecated]] inline uint8_t OSCritInit(OS_CRIT *pCrit)
02160 {
02161     return pCrit->Init();
02162 }
02163 [[deprecated]] inline uint8_t OSCritLockAndEnter(OS_CRIT *pCrit, uint16_t timeout)
02164 {
02165     return pCrit->LockAndEnter(timeout);
02166 }
02167
02181 [[deprecated]] inline uint8_t OSCritEnter(OS_CRIT *pCrit, uint16_t timeout)
02182 {
02183     return pCrit->Enter(timeout);
02184 }
02185
02198 [[deprecated]] inline uint8_t OSCritEnterNoWait(OS_CRIT *pCrit)
02199 {
02200     return pCrit->EnterNoWait();
02201 }
02202
02215 [[deprecated]] inline uint8_t OSCritLeave(OS_CRIT *pCrit)
02216 {
02217     return pCrit->Leave();
02218 }
02219 [[deprecated]] inline uint8_t OSCritLeaveAndUnlock(OS_CRIT *pCrit)
02220 {
02221     return pCrit->LeaveAndUnlock();
02222 }
02223
02227 uint8_t OSTaskID(void);
02228
02232 const char *OSTaskName();
02233
02234 void OSChangeTaskWhen(uint16_t task_prio, uint32_t to_when);
02235
02246 inline void OSChangeTaskDly(uint16_t task_prio, uint32_t to_count)
02247 {
02248     uint32_t to_when = to_count + TimeTick;
02249     if (!to_when) to_when++;
02250     OSChangeTaskWhen(task_prio, to_when);
02251 }
02252
02256 void OSDumpStack(void);
02257
02258 #if (defined NBRTOS_STACKOVERFLOW) || (defined NBRTOS_STACKUNDERFLOW)
02259 void EnableOSStackProtector();
02260 extern "C" void OSStackProtectCtxSw();       // ASM task switcher
02261 extern "C" void OSStackProtectIntCtxSw();    // ASM task switcher
02262 extern "C" void OSStackProtector();          // extern because must be called from ASM
02263 #endif
02264
02265 #ifdef NBRTOS_STACKCHECK
02274 void OSDumpTCBStacks(void);
02275
02284 void OSDumpTasks(void);
02285
02294 void OSStartTaskDumper(uint8_t prio, uint32_t reportInterval);
02295 #endif
02296
02297 #ifdef NBRTOS_TASKLIST
02303 void ShowTaskList(void);
02304 #endif
02305
02306 #ifdef NBRTOS_TIME
02307 uint32_t GetCurrentTaskTime(uint32_t *const TotalTicks);
02308 void ShowTaskTimes(void);
02309 void ClearTaskTimes(void);
02310 #endif
02311
02323 class OSLockObj
02324 {
02325    public:
02329     OSLockObj() { OSLock(); };
02330
02334     ~OSLockObj() { OSUnlock(); };
02335 } __attribute__((packed));
02336
02346 class OSCriticalSectionObj
02347 {
02348     OS_CRIT *pcrit;
02349
02350    public:
02357     OSCriticalSectionObj(OS_CRIT &ocrit)
02358     {
02359         pcrit = &ocrit;
02360         ocrit.Enter(0);
02361     };
```

```
02362
02371     OSCriticalSectionObj(OS_CRIT &ocrit, bool NoWait, TickTimeout &timeout)
02372     {
02373         pcrit = &ocrit;
02374         if (NoWait)
02375             ocrit.EnterNoWait();
02376         else
02377             ocrit.Enter(timeout);
02378     };
02379
02384     ~OSCriticalSectionObj() { pcrit->Leave(); }
02385 } __attribute__((packed));
02386
02397 class OSLockAndCritObj
02398 {
02399     OS_CRIT *pcrit;
02400
02401   public:
02408     OSLockAndCritObj(OS_CRIT &ocrit) : pcrit(&ocrit) { pcrit->LockAndEnter(0); }
02409
02414     ~OSLockAndCritObj() { pcrit->LeaveAndUnlock(); }
02415 };
02416
02428 class OSSpinCrit
02429 {
02430     OS_CRIT *pcrit;
02431
02432   public:
02439     OSSpinCrit(OS_CRIT &ocrit) : pcrit(&ocrit)
02440     {
02441         while (pcrit->EnterNoWait() == OS_TIMEOUT) {}
02442     }
02443
02448     ~OSSpinCrit() { pcrit->Leave(); }
02449 };
02450
02451
02455 class USERCritObj
02456 {
02457   public:
02458     USERCritObj() { USER_ENTER_CRITICAL(); }
02459     ~USERCritObj() { USER_EXIT_CRITICAL(); }
02460 };
02461
02462
02463
02470 inline bool OS_CRIT::OwnedByCurTask()
02471 {
02472     return OSCritOwnerTCB == OSTCBCur;
02473 }
02474
02486 class NBRtosInitObj
02487 {
02488     static NBRtosInitObj * pHead;
02489     static bool bAlreadInited;
02490     NBRtosInitObj *pNext;
02491
02492     protected:
02493       inline bool WasInitDone() {return bAlreadInited; }
02494       NBRtosInitObj();
02495      ~NBRtosInitObj();
02496
02497     public:
02501         virtual void Notify()=0;
02502         static void InitWholeSet();   //Used internally
02503 };
02504
02505
02506 #endif
02507
02508
02509
02510
```

24.426 nbssh.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NBSSH_H_
00006 #define _NBSSH_H_
00007
00008 #include <ssh/NetBurner/NbWolfSsh.h>
00009
```

```
00010 #endif   //_NBSSH_H_
```

24.427 nbstring.h File Reference

NetBurner String Class.

```
#include <stdint.h>
#include <stdio.h>
#include <system.h>
#include <fd_adapter.h>
```

Classes

- class NBString

  Lightweight alternative to C++ CString class.

24.427.1 Detailed Description

NetBurner String Class.
A lightweight alternative to the C++ CString class. Note that if you prefer CString, you can add it using the Standard Template Library (STL), but the STL consumes a significant amount of resources.

24.428 nbstring.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00020 #ifndef DEF_NB_STRING
00021 #define DEF_NB_STRING (1)
00022
00023 #include <stdint.h>
00024 #include <stdio.h>
00025 #include <system.h>
00026 #include <fd_adapter.h>
00027 class JsonRef;
00028 class ParsedJsonDataSet;
00029
00030
00031 const int native_string_storage = 15;   // Strings up to this amount are stored locally instead of
      allocating memory
00032
00033
00034 //Flag for no position or last position in string.
00035 //named npos to be compaitblke with other string clsses...
00036 const size_t npos = -1;
00037 const size_t NO_POS =-1; //More inline with netburner code foiomats constats are all caps.
00038
00039 const uint8_t FLAG_LOCAL = 0x80;
00040 const uint8_t FLAG_CONST = 0x40;
00041 const uint8_t FLAG_ALLOC = 0x20;
00042 const uint8_t FLAG_CLEAR_LOCAL_MASK = 0x7F;
00043 const int EXTRA_ALLOC_SPACE = 64;
00044
00045
00046
00047
00048 struct nbstring_allocated_info
00049 {
00050     uint8_t flag;   // 0 = local storage, > native_string_storage = allocated memory
00051     uint8_t unused1;
00052     uint16_t unused2;
00053     uint32_t used;      // How many bytes are used in the string
00054     uint32_t alloced;   // Size of data pointed at by pData (used and unused). Will be 0 for data
      declared as constant
00055     uint8_t *pData;     // Pointer to string
00056 };
00057
00058 struct nbstring_local
00059 {
00060     uint8_t siz_flag;                     // Number of bytes stored in local storage
```

```
00061     uint8_t stor[native_string_storage];   // actual string data stored locally
00062 };
00063
00064 class NBStorageManager
00065 {
00066    private:
00067     union
00068     {
00069         nbstring_local nbss;
00070         nbstring_allocated_info inf;
00071     };
00072
00073    public:
00074     inline bool IsConstStore() const {return  ((inf.flag & FLAG_CONST)!=0);};
00075     inline bool IsLocalStore() const { return ((inf.flag & FLAG_LOCAL)!=0);};
00076
00077     void reservespace(size_t n);
00078     void setusedsize(size_t n);
00079
00080     size_t GetUsed() const;
00081     size_t GetAvailable() const;
00082     char *GetBuf();
00083     const char *GetConstBuf() const;
00084
00085     void SetFromConstant(const char *c, int len);
00086
00087     void clear();
00088
00089     ~NBStorageManager();
00090     NBStorageManager();
00091
00092     static void swap(NBStorageManager &n1, NBStorageManager &n2);
00093 #ifdef STRING_DIAGS
00094     void DumpDiag();
00095 #endif
00096 };
00097
00098 class NBString;
00099
00100
00101 class NBStringBuilder : public NBStorageManager, public fd_adapter
00102 {
00103    public:
00104     int read(char *buf, int nbytes) override;
00105     int write(const char *buf, int nbytes) override;
00106     int close() override;
00107
00108     void moveTo(NBString &s);
00109 };
00110
00116 class NBString
00117 {
00118    protected:
00119     NBStorageManager nbs;
00120
00121     void DoStorage(const char *cp, int len);
00122
00123     static bool testInterpolationModifier(const char *begin, const char *end);
00124     void modifyInterpValue(char *bufStart, char *valStart, const char *valEnd, const char *modStart,
      const char *modEnd);
00125
00126
00127    public:
00128     // constructors
00132     NBString();
00133
00139     NBString(const NBString &str);
00140
00148     NBString(const NBString &str, size_t pos, size_t len = npos);
00149
00155     NBString(const char *s);
00156
00163     NBString(const char *s, size_t n);
00164
00168     ~NBString();
00169
00170     // Copy ...
00171     NBString &operator=(const NBString &str);
00172     NBString &operator=(const char *s);
00173     NBString &operator=(char c);
00174
00175     //Actual assigment implimented in jsonlexer not NBString
00176     NBString &operator=(const JsonRef &jr);
00177
00178
00179
00180     // Look up
```

```
00181     char &operator[](size_t pos);
00182     const char &operator[](size_t pos) const;
00183
00192     const char *c_str() const;
00193
00202     NBString substr(size_t pos = 0, size_t len = npos) const;
00203
00213     int compare(const NBString &str) const;
00214
00224     int compare(const char *s) const;
00225
00226     /*
00227      * The following methods implement finding a string within another string.
00228      * str - The string to find in this string.
00229      * pos - The position in this string to start searching.
00230      * n - How many characters of str to search for.
00231      * c - What character to find.
00232      *
00233      * Returns a -1 if unable to find or if the search fails.
00234      */
00244     size_t find(const NBString str, size_t pos = 0) const;
00245
00255     size_t find(const char *str, size_t pos = 0) const;
00256
00267     size_t find(const char *s, size_t pos, size_t n) const;
00268
00278     size_t find(char c, size_t pos = 0) const;
00279
00291     size_t replace(const char *findStr, char rep, size_t startPos = 0, size_t end = 0);
00292
00304     size_t replace(char c, char rep, size_t startPos = 0, size_t end = 0);
00305
00306
00307
00308
00309
00315     size_t size() const;
00316
00322     size_t length() const;
00323
00330     void clear();
00331
00340     bool empty() const;
00341
00352     NBString &Append(const char *str, size_t len);
00353
00364     NBString &FdAppend(int fd, size_t len);
00365
00375     NBString &Reserve(size_t len);
00376
00377     NBString &operator+=(const NBString &str);
00378     NBString &operator+=(const char *str);
00379     NBString &operator+=(char c);
00380
00381
00382     NBString & ToB64Url();
00383
00384
00385
00386 #ifdef STRING_DIAGS
00387     void DumpDiag();
00388 #endif
00389
00397     friend void swap(NBString &x, NBString &y);
00398
00399     friend void NBStringBuilder::moveTo(NBString &);
00400     // concatinate
00401     friend NBString operator+(const NBString &lhs, const NBString &rhs);
00402     friend NBString operator+(const NBString &lhs, const char *rhs);
00403     friend NBString operator+(const char *lhs, const NBString &rhs);
00404     friend NBString operator+(const NBString &lhs, char rhs);
00405     friend NBString operator+(char lhs, const NBString &rhs);
00406
00407     // Compare...
00408     friend bool operator==(const NBString &lhs, const NBString &rhs);
00409     friend bool operator==(const char *lhs, const NBString &rhs);
00410     friend bool operator==(const NBString &lhs, const char *rhs);
00411     friend bool operator!=(const NBString &lhs, const NBString &rhs);
00412     friend bool operator!=(const char *lhs, const NBString &rhs);
00413     friend bool operator!=(const NBString &lhs, const char *rhs);
00414     friend bool operator<(const NBString &lhs, const NBString &rhs);
00415     friend bool operator<(const char *lhs, const NBString &rhs);
00416     friend bool operator<(const NBString &lhs, const char *rhs);
00417     friend bool operator<=(const NBString &lhs, const NBString &rhs);
00418     friend bool operator<=(const char *lhs, const NBString &rhs);
00419     friend bool operator<=(const NBString &lhs, const char *rhs);
00420     friend bool operator>(const NBString &lhs, const NBString &rhs);
```

```
00421     friend bool operator>(const char *lhs, const NBString &rhs);
00422     friend bool operator>(const NBString &lhs, const char *rhs);
00423     friend bool operator>=(const NBString &lhs, const NBString &rhs);
00424     friend bool operator>=(const char *lhs, const NBString &rhs);
00425     friend bool operator>=(const NBString &lhs, const char *rhs);
00426
00427     // Member functions to add that are not in the standard string classes
00428
00435     int vsiprintf(const char *format, va_list &vl);
00436
00442     int sprintf(const char *format, ...);
00443
00449     int sprintf(NBString const format, ...);
00450
00456     int siprintf(const char *format, ...);
00457
00463     int siprintf(NBString const format, ...);
00464
00470     int stoi() const;
00471
00477     long stol() const;
00478
00484     unsigned int stoui() const;
00485     unsigned int stouihex() const;
00486
00492     unsigned long stoul() const;
00493     unsigned long stoulhex() const;
00494
00495
00501     double stod() const;
00502
00508     IPADDR to_ipaddr() const;
00509
00510
00511     /*
00512     Calculate the base64url encode and returnt he string.
00513     */
00514     friend NBString  b64UrlToString(const puint8_t p, int len);
00515
00516
00517
00518
00519     /* Possible future additions, Compare substrings.
00520     int compare (size_t pos, size_t len, const NBString& str) const;
00521     int compare (size_t pos, size_t len, const NBString& str,size_t subpos, size_t sublen) const;
00522     int compare (size_t pos, size_t len, const char* s) const;
00523     int compare (size_t pos, size_t len, const char* s, size_t n) const;
00524     */
00525
00535     size_t copy(char *s, size_t len, size_t pos = 0) const;
00536
00546     size_t strcopy(char *s, size_t len, size_t pos = 0) const;
00547
00548     /* *
00549      * @brief Perform a string interpolation and place the finished interpolation
00550      *  in the Destination string
00551      *
00552      * @param dest  Destination string
00553      *
00554      * @returns Whether the interpolation was successful
00555      */
00556     bool Interpolate(NBString &dest);
00557
00558 };
00559
00560 NBString  b64UrlToString(const puint8_t p, int len);
00561
00562
00563
00564 class NBInterpolatedString : public NBString
00565 {
00566     bool bNeedInterpolate;
00567 public:
00568     // constructors
00572     NBInterpolatedString();
00573
00579     NBInterpolatedString(const NBString &str);
00580
00588     NBInterpolatedString(const NBString &str, size_t pos, size_t len = npos);
00589
00595     NBInterpolatedString(const char *s);
00596
00603     NBInterpolatedString(const char *s, size_t n);
00604
00608     ~NBInterpolatedString();
00609
00610     // Copy ...
```

```
00611     NBInterpolatedString &operator=(const NBString &str);
00612     NBInterpolatedString &operator=(const char *s);
00613
00624     NBString &Append(const char *str, size_t len);
00625
00636     NBString &FdAppend(int fd, size_t len);
00637
00638     NBString &operator+=(const NBString &str);
00639     NBString &operator+=(const char *str);
00640     NBString &operator+=(char c);
00641
00642     NBInterpolatedString &Interpolate();
00643 };
00644
00645
00646
00647
00648
00649
00650
00651 // concatinate
00652 NBString operator+(const NBString &lhs, const NBString &rhs);
00653 NBString operator+(const NBString &lhs, const char *rhs);
00654 NBString operator+(const char *lhs, const NBString &rhs);
00655 NBString operator+(const NBString &lhs, char rhs);
00656 NBString operator+(char lhs, const NBString &rhs);
00657
00658 // Compare...
00659 inline bool operator==(const NBString &lhs, const NBString &rhs)
00660 {
00661     return lhs.compare(rhs) == 0;
00662 };
00663 inline bool operator==(const char *lhs, const NBString &rhs)
00664 {
00665     return rhs.compare(lhs) == 0;
00666 };
00667 inline bool operator==(const NBString &lhs, const char *rhs)
00668 {
00669     return lhs.compare(rhs) == 0;
00670 };
00671 inline bool operator!=(const NBString &lhs, const NBString &rhs)
00672 {
00673     return lhs.compare(rhs) != 0;
00674 };
00675 inline bool operator!=(const char *lhs, const NBString &rhs)
00676 {
00677     return rhs.compare(lhs) != 0;
00678 };
00679 inline bool operator!=(const NBString &lhs, const char *rhs)
00680 {
00681     return lhs.compare(rhs) != 0;
00682 };
00683 inline bool operator<(const NBString &lhs, const NBString &rhs)
00684 {
00685     return lhs.compare(rhs) > 0;
00686 };
00687 inline bool operator<(const char *lhs, const NBString &rhs)
00688 {
00689     return rhs.compare(lhs) < 0;
00690 };
00691 inline bool operator<(const NBString &lhs, const char *rhs)
00692 {
00693     return lhs.compare(rhs) > 0;
00694 };
00695 inline bool operator<=(const NBString &lhs, const NBString &rhs)
00696 {
00697     return lhs.compare(rhs) >= 0;
00698 };
00699 inline bool operator<=(const char *lhs, const NBString &rhs)
00700 {
00701     return rhs.compare(lhs) <= 0;
00702 };
00703 inline bool operator<=(const NBString &lhs, const char *rhs)
00704 {
00705     return lhs.compare(rhs) >= 0;
00706 };
00707 inline bool operator>(const NBString &lhs, const NBString &rhs)
00708 {
00709     return lhs.compare(rhs) < 0;
00710 };
00711 inline bool operator>(const char *lhs, const NBString &rhs)
00712 {
00713     return rhs.compare(lhs) > 0;
00714 };
00715 inline bool operator>(const NBString &lhs, const char *rhs)
00716 {
00717     return lhs.compare(rhs) < 0;
```

```
00718 };
00719 inline bool operator>=(const NBString &lhs, const NBString &rhs)
00720 {
00721     return lhs.compare(rhs) <= 0;
00722 };
00723 inline bool operator>=(const char *lhs, const NBString &rhs)
00724 {
00725     return rhs.compare(lhs) >= 0;
00726 };
00727 inline bool operator>=(const NBString &lhs, const char *rhs)
00728 {
00729     return lhs.compare(rhs) <= 0;
00730 };
00731
00732
00733 #endif
00734
```

24.429 nbtime.h File Reference

NetBurner Time Header File.

```
#include <servlets.h>
#include <config_obj.h>
#include <nbstring.h>
#include <nettimer.h>
#include <time.h>
```

Functions

- time_t set_time (time_t time_to_set, uint32_t fraction=0)

  Set the system time to the value passed in the time_t parameter.
- time_t timegm (struct tm *bts)

  Returns the value of type time_t that represents the GMT time described by the tm structure pointed by bts.
- void tzsetchar (const char *tzenv)

  Set the system local time.

24.429.1 Detailed Description

NetBurner Time Header File.

24.430 nbtime.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <servlets.h>
00006 #include <config_obj.h>
00007 #include <nbstring.h>
00008 #include <nettimer.h>
00048 #include <time.h>
00049
00050 #ifndef _NB_TIME_H
00051 #define _NB_TIME_H
00052
00053 #ifdef __cplusplus
00054 extern "C"
00055 {
00056 #endif /* __cplusplus */
00057
00058     time_t time(time_t *pt);
00059 #ifdef __cplusplus
00060 };
00061 #endif /* __cplusplus */
00062
00072 time_t set_time(time_t time_to_set, uint32_t fraction=0);
00073
00085 time_t timegm(struct tm *bts);
```

```
00086
00087 #ifdef NB_NET_TYPES_H
00088
00099 BOOL SetNTPTime(IPADDR ntpserver);
00100
00112 BOOL SetTimeNTPFromPool(bool debug = FALSE);
00113
00125 time_t GetNTPTime(IPADDR NTP_server_ip, uint32_t * pFraction=0);
00126
00127
00128 #define TWELVE_HOURS_SECS (12*3600)
00129 #define ONE_DAY_SECS (14*3600)
00130
00153 class NtpClientServlet: public servlet, public config_obj, public TimeOutElement
00154 {
00155
00156 config_string m_ServerAddress;
00157 config_int    m_interval;
00158 ConfigEndMarker;   // No new data members below this line
00159 OS_FLAGS NTPFlag;
00160 NBString m_server;  //Name of server
00161 time_t last_update;  //time of last valid update in system Secs
00162 time_t last_attempt; //time of last update in system Secs
00163 uint32_t m_TickSent; //Transmit time
00164 servlet_list * m_pWhoToReJoin;
00165
00166 int m_fd;         //fd to hold DNS Requests and UDP send/rx
00167 int cur_state; //State variable
00168
00169
00170 //Servlet virtual functions
00171  virtual int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00172  virtual void ProcessSelectResult(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00173 //Timeout element virtual function
00174  virtual void TimeElementEvent();
00175
00176 public:
00184  NtpClientServlet(const char* NTP_Server="pool.ntp.org",uint32_t
      update_interval_sec=TWELVE_HOURS_SECS);
00185
00186
00196  NtpClientServlet(servlet_list * List_To_Join, const char* NTP_Server="pool.ntp.org",uint32_t
      update_interval_sec=TWELVE_HOURS_SECS);
00197
00207  bool TimeIsValid();
00208
00218 bool TimeIsCurrent();
00219
00230  bool RefreshNow(uint32_t ticks_to_wait);
00231
00241  bool WaitForValid(uint32_t ticks_to_wait);
00242
00250  inline  time_t GetSecsSinceUpdate() {return Secs-last_update; }
00251 };
00252
00253
00254
00255
00256
00257 #endif /* NB_NET_TYPES_H */
00258
00259 /*
00260  * Legacy function calls.
00261  */
00262 // int IOBoardRTCSetSystemFromRTCTime(void);
00263 // int IOBoardRTCSetRTCfromSystemTime(void);
00264
00265 /*
00266  * Set the local time zone by using the TZ environment variable.
00267  * Use one of the following for the U.S.:
00268  *
00269  *    tzsetchar( "EST5EDT4,M3.2.0/02:00:00,M11.1.0/02:00:00" );     Eastern
00270  *    tzsetchar( "CST6CDT5,M3.2.0/02:00:00,M11.1.0/02:00:00" );     Central
00271  *    tzsetchar( "MST7MDT6,M3.2.0/02:00:00,M11.1.0/02:00:00" );     Mountain
00272  *    tzsetchar( "MST7" );                                          Arizona
00273  *    tzsetchar( "PST8PDT7,M3.2.0/02:00:00,M11.1.0/02:00:00" );     Pacific
00274  *    tzsetchar( "AKST9AKDT8,M3.2.0/02:00:00,M11.1.0/02:00:00" );   Alaska
00275  *    tzsetchar( "HST10" );                                         Hawaii
00276  *
00277  * Effective 2007, the U.S. starts daylight savings time on the second Sunday of
00278  * March at 2:00 AM and ends on the first Sunday of November at 2:00 AM.  The
00279  * starting and ending date/times for daylight savings time can be omitted from
00280  * the string since these rules are used by default.
00281  */
00282
00306 void tzsetchar(const char *tzenv);
00307
```

```
00308 #endif /* _NB_TIME_H */
00309
```

24.431 nbupdate.h File Reference

Signed Application Update API.

```
#include <predef.h>
#include <constants.h>
#include <basictypes.h>
#include <PlatformHeader.h>
```

Macros

- #define **NBUP_ERR_NO_ERR** 0

  No errors.
- #define **NBUP_ERR_BAD_SOCK** -1

  File descriptor socket is invlaid.
- #define **NBUP_ERR_TIMEOUT** -2

  Timeout.
- #define **NBUP_ERR_TOO_LARGE** -3

  Application update record is too large.
- #define **NBUP_ERR_WRONG_PLAT** -4

  Device platform name does not match.
- #define **NBUP_ERR_BAD_PROG** -5

  Programming failed, invalid checksum.
- #define **NBUP_ERR_BAD_AUTH** -6

  Application update record authentication failed.

Functions

- void RegisterAppSigningPublicKey (const char *pKey)

  Enable APP Signing by registering a RSA PEM or DER format public key.
- int ProgramApplication (uint32_t where, uint8_t *pAppImage)

  Program an appliction image into Flash memory.
- int UpdateFromStream (int fd, AppUpdateRecord *&pu, uint32_t timeout)

  Program/update an appliction image from a data stream.

24.431.1 Detailed Description

Signed Application Update API.

24.432 nbupdate.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef __NBUPDATE_H
00016 #define __NBUPDATE_H
00017
00018 #include <predef.h>
00019 #include <constants.h>
00020 #include <basictypes.h>
00021 #include <PlatformHeader.h>
00022
00027 #define NBUP_ERR_NO_ERR 0
00028 #define NBUP_ERR_BAD_SOCK -1
```

```
00029 #define NBUP_ERR_TIMEOUT -2
00030 #define NBUP_ERR_TOO_LARGE -3
00031 #define NBUP_ERR_WRONG_PLAT -4
00032 #define NBUP_ERR_BAD_PROG -5
00033 #define NBUP_ERR_BAD_AUTH -6
00036 // States...
00037 // Looking for an S....
00038 // S getting S record type
00039 // Parsing Address up to 8 chars)
00040 // Parsing Data
00041 // Checking csum....
00042
00043 // S0MOD5441X
00044 // S315C0040000 40 00 04 00 40 00 05 40 00 05 F6 3C 00 03 AA 5C  1D
00045 // S315C0040010 1B 41 90 65 A7 0A 18 11 00 41 FF FF 01 4C 6C 80  73
00046 // S7054000054075
00047
00048 extern const char PlatformName[];
00049
00050 const uint16_t FLAG_DOING_S0 = 1;
00051 const uint16_t FLAG_DOING_S3 = 2;
00052 const uint16_t FLAG_DOING_S7 = 4;
00053 const uint16_t FLAG_SUS_VALID = 8;
00054 const uint16_t FLAG_SAW_FINALS7 = 16;
00055 const uint16_t FLAG_SAW_BINARY = 32;
00056 const uint16_t FLAG_DOING_BINARY = 64;
00057 const uint16_t FLAG_NEED_AUTH_CHARS = 128;
00058 const uint16_t FLAG_DONE_S0 = 0x100;
00059
00060 class AppUpdateRecord
00061 {
00062     bool m_bWrittenToFlash;
00063     uint8_t m_state;
00064     uint8_t m_csum;
00065     uint8_t m_cnt;
00066     uint16_t m_flags;
00067     uint32_t tempv;
00068
00069     PlatformFlashHeaderStruct bsus;
00070     PlatformFlashHeaderStruct sus;
00071     uint8_t *pRecord;
00072     uint32_t DataRead;
00073     uint32_t BaseAddress;
00074     const char *m_Error;
00075     int m_ErrorNum;
00076
00077    public:
00078     AppUpdateRecord();
00079     ~AppUpdateRecord();
00080     // Return true when done
00081     bool ParseChars(char *chars, int nChars);
00082     void ParseOneChar(char c);
00083     void PutValue(uint8_t v);
00084     bool resp;
00085     bool bUpdateRefused;
00086
00087     enum
00088     {
00089         ERR_NO_ERR,
00090         ERR_NO_REC,
00091         ERR_BAD_RX,
00092         ERR_WRITE_FAIL,
00093         ERR_ALREADY_WRITTEN,
00094         ERR_BAD_ALLOC,
00095         ERR_BAD_PLAT,
00096         ERR_BAD_AUTH,
00097         ERR_BAD_CSUM
00098     };
00099
00100     int DoFlashProgram();
00101     bool CsumValid();
00102     bool ImageReceived()
00103     {
00104         if ((m_Error) || (m_ErrorNum) || (!pRecord) || (DataRead < sus.CompleteRecordSize() -
      sus.SizeWithoutPad()) ||
00105             ((m_flags & FLAG_SAW_FINALS7) == 0) || (m_flags & FLAG_NEED_AUTH_CHARS))
00106             return false;
00107         return true;
00108     };
00109
00110     const char *GetError() { return m_Error; };
00111
00112     int GetErrorNum() { return m_ErrorNum; }
00113
00114     void SetBadAuth()
00115     {
00116         m_ErrorNum = ERR_BAD_AUTH;
```

```
00117         m_Error = "Bad Authentication";
00118     }
00119
00120     bool bWaitingOnAuth() { return (((m_flags & FLAG_NEED_AUTH_CHARS) != 0) || (m_ErrorNum ==
      ERR_BAD_AUTH)); }
00121
00122     uint32_t GetbaseAddress() { return BaseAddress; };
00123
00124     uint32_t GetLength() { return sus.CompleteRecordSize() - sus.SizeWithoutPad(); };
00125
00126     uint32_t GetFullLength() { return bsus.CompleteRecordSize(); };
00127
00128     const uint8_t *GetData() { return pRecord; };
00129
00130     bool IsBinary() { return ((m_flags & FLAG_DOING_BINARY) != 0); }
00131
00132     int32_t GetRemainingLen() { return (int32_t)bsus.CompleteRecordSize() - DataRead; }
00133 };
00134
00135 enum AppAuthResult
00136 {
00137     eOKToProgram,
00138     eBadSignature,
00139     eMoreData
00140 };
00141
00142 // Helper functiosn for signed apps.
00143 typedef AppAuthResult (*AppAuthenticateFunc)(AppUpdateRecord *ar, uint8_t *chars, int n);
00144 extern AppAuthenticateFunc pAppAuthFunc;
00145
00154 void RegisterAppSigningPublicKey(const char *pKey);
00155
00166 int ProgramApplication(uint32_t where, uint8_t *pAppImage);
00167
00179 int UpdateFromStream(int fd, AppUpdateRecord *&pu, uint32_t timeout);
00180
00181 #endif /* ----- #ifndef __NBUPDATE_H  ----- */
00182
```

24.433 netbios.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NETBIOS_H_
00006 #define _NETBIOS_H_
00007
00008 #include <basictypes.h>
00009 #include <buffers.h>
00010 #include <nettypes.h>
00011
00012 /*
00013  ******************************************************************************
00014  *
00015  * Reference
00016  *    RFC 883 - Domain Names - Implementation and Specification, November 1983
00017  *
00018  *    RFC 1001 - Protocol standard for a NetBIOS service on a TCP/UDP
00019  *       transport: Concepts and methods, March 1987
00020  *    RFC 1002 - Protocol standard for a NetBIOS service on a TCP/UDP
00021  *       transport: Detailed specifications, March 1987
00022  *    IANA - WELL KNOWN PORT NUMBERS, July 10, 2008
00023  *
00024  *    Microsoft Help and Support - NetBIOS Suffixes (16th Character of NetBIOS
00025  *       Name) Article ID: 163409, February 26, 2007 Revision: 4.2
00026  *
00027  *    Note: NetBIOS names are described as "compressed" which is a second level
00028  *       encoding described in RFC 883. All name comparisons are case
00029  *       insensitive.
00030  *
00031  ******************************************************************************
00032  */
00033
00034 /*
00035  ******************************************************************************
00036  *
00037  * Definitions
00038  *
00039  ******************************************************************************
00040  */
00041
00042 /*
00043  ******************************************************************************
00044  * Microsoft variant (comment to remove)
```

```
00045  ***************************************************************************
00046  */
00047 #define NETBIOS_NAME_MICROSOFT_VARIANT (1)
00048
00049 /* NetBIOS Name size in upper case alphanumerics */
00050 #ifdef NETBIOS_NAME_MICROSOFT_VARIANT
00051 #define NETBIOS_NAME_SIZE_IN_CHARS (15)
00052 #define NETBIOS_NAME_MICROSOFT_SUFFIX_IN_CHARS (1)
00053 #else /* #ifdef NETBIOS_NAME_MICROSOFT_VARIANT */
00054 #define NETBIOS_NAME_SIZE_IN_CHARS (16)
00055 #endif /* #ifdef NETBIOS_NAME_MICROSOFT_VARIANT */
00056
00057 /* NetBIOS name suffixes (Microsoft variant ) */
00058 #define NETBIOS_NAME_SUFFIX_WORKSTATION (0x00)
00059
00060 /* Operation specifier */
00061 #define NETBIOS_OPCODE_QUERY (0)
00062 #define NETBIOS_OPCODE_REGISTRATION (5)
00063 #define NETBIOS_OPCODE_RELEASE (6)
00064 #define NETBIOS_OPCODE_WACK (7)
00065 #define NETBIOS_OPCODE_REFRESH (8)
00066
00067 /* Owner node type */
00068 #define NETBIOS_OWNER_NODE_TYPE_B_NODE (0x0)
00069 #define NETBIOS_OWNER_NODE_TYPE_P_NODE (0x1)
00070 #define NETBIOS_OWNER_NODE_TYPE_M_NODE (0x2)
00071 #define NETBIOS_OWNER_NODE_TYPE_H_NODE (0x3)
00072
00073 /*
00074    Name first byte bit patterns
00075
00076       NETBIOS_NAME_LABEL_NAME
00077       NETBIOS_NAME_LABEL_POINTER
00078
00079  */
00080 #define NETBIOS_NAME_LABEL_NAME (0x20)
00081 #define NETBIOS_NAME_LABEL_POINTER (0xC0)
00082
00083 /* Compressed name length number of bytes */
00084 #define NETBIOS_COMPRESSED_NAME_LENGTH (NETBIOS_NAME_SIZE_IN_CHARS * 2)
00085
00086 /*
00087    Entry Type
00088
00089       NETBIOS_REQUEST_TYPE_NB             - General name service resource record
00090       NETBIOS_REQUEST_TYPE_NBSTAT         - Node status resource record
00091
00092  */
00093 #define NETBIOS_REQUEST_TYPE_NB (0x0020)
00094 #define NETBIOS_REQUEST_TYPE_NBSTAT (0x0021)
00095
00096 /*
00097    Entry Class
00098
00099       NETBIOS_REQUEST_CLASS_IN            - Internet class
00100
00101  */
00102 #define NETBIOS_REQUEST_CLASS_IN (0x0001)
00103
00104 /* Infinite TTL */
00105 #define NETBIOS_INFINITE_TTL (0)
00106
00107 /* Node Status Response Statistics unit ID */
00108 #define NETBIOS_UNIT_ID_IN_BYTES (6)
00109
00110 /*
00111  ***************************************************************************
00112  *
00113  * Enumerations
00114  *
00115  ***************************************************************************
00116  */
00117 /*
00118  ***************************************************************************
00119  *
00120  * Structures
00121  *
00122  ***************************************************************************
00123  */
00124 /*
00125       Packet type code (OPCODE)
00126          r                        - Response flag, 0 - request, 1 - response
00127          opcode                   - Operation specifier
00128
00129       Operation flags (NM_FLAGS)
00130          aa                       - Authorative answer flag,
00131          tc                       - Truncation flag
```

```
00132          rd                      - Recursion desired flag
00133          ra                      - Recursion available flag
00134          b                       - Broadcast flag, 0 - Unicast, 1 - Broadcast
00135
00136       Result codes (RCODE)
00137          rcode                   - Result code
00138
00139
00140  */
00141 typedef struct _OpCodeNmFlagsRCode
00142 {
00143     uint16_t r : 1;
00144     uint16_t opcode : 4;
00145     uint16_t aa : 1;
00146     uint16_t tc : 1;
00147     uint16_t rd : 1;
00148     uint16_t ra : 1;
00149     uint16_t mbz : 2;
00150     uint16_t b : 1;
00151     uint16_t rcode : 4;
00152
00153 } __attribute__((packed)) OpCodeNmFlagsRCode;
00154
00155 /*
00156       Name Service Packet Header
00157
00158          name_trn_id             - Transaction id, unique for requestor
00159          opcode_nmflags_rcode    - Packet type code, operation flags, results
00160          qdcount                 - Question entry count
00161          ancount                 - Resource record count
00162          nscount                 - Authority record count
00163          arcount                 - Additional record count
00164
00165 */
00166 typedef struct _NameServicePacketHeader
00167 {
00168     uint16_t name_trn_id;
00169     OpCodeNmFlagsRCode opcode_nmflags_rcode;
00170     uint16_t qdcount;
00171     uint16_t ancount;
00172     uint16_t nscount;
00173     uint16_t arcount;
00174
00175 } __attribute__((packed)) NameServicePacketHeader;
00176
00177 /*
00178       Name Service Name
00179
00180          labelLength             - Label length
00181          name                    - Compressed name
00182          zero_termination        - Zero termination
00183
00184 */
00185 typedef struct _NameServiceName
00186 {
00187     uint8_t label_length_count;
00188     uint8_t name[NETBIOS_COMPRESSED_NAME_LENGTH];
00189 #ifdef NETBIOS_NAME_MICROSOFT_VARIANT
00190     uint8_t suffix[(NETBIOS_NAME_MICROSOFT_SUFFIX_IN_CHARS * 2)];
00191 #endif /* #ifdef NETBIOS_NAME_MICROSOFT_VARIANT */
00192     uint8_t zero_termination;
00193
00194 } __attribute__((packed)) NameServiceName;
00195
00196 /*
00197       Name Service Question Entry
00198
00199          name                    - Name
00200          question_type           - Request type (rr_type for answer)
00201          question_class          - Class of request (rr_class for answer)
00202
00203 */
00204 typedef struct _NameServiceQuestionEntry
00205 {
00206     NameServiceName name;
00207     uint16_t question_type;
00208     uint16_t question_class;
00209
00210 } __attribute__((packed)) NameServiceQuestionEntry;
00211
00212 /*
00213       Netbios flags
00214          g                       - Group name flag
00215                                    0 - Unique name, 1 - Group name
00216          ont                     - Owner node type
00217
00218  */
```

```
00219 typedef struct _NbFlags
00220 {
00221     uint16_t g : 1;
00222     uint16_t ont : 2;
00223     uint16_t mbz : 13;
00224
00225 } __attribute__((packed)) NbFlags;
00226
00227 /*
00228       Name Service Resource Record Header (Name Pointer)
00229
00230          rr_type                 - Request type
00231          rr_class                - Class of request
00232
00233  */
00234 typedef struct _NameServiceResourceRecordNamePointerHeader
00235 {
00236     uint8_t rr_nameLabel;
00237     uint8_t rr_nameIndex;
00238     uint16_t rr_type;
00239     uint16_t rr_class;
00240
00241 } __attribute__((packed)) NameServiceResourceRecordNamePointerHeader;
00242
00243 /*
00244       Name Service Resource Record Trailer
00245
00246          rr_type                 - Request type
00247          rr_class                - Class of request
00248          ttl                     - Time to live for resource name
00249          rdlength                - Bytes in RDATA field
00250          nb_flags                - Flags (RDATA)
00251          nb_address              - Address
00252
00253  */
00254 typedef struct _NameServiceResourceRecordTrailer
00255 {
00256     uint32_t ttl;
00257     uint16_t rdlength;
00258     NbFlags nb_flags;
00259     uint32_t nb_address;
00260
00261 } __attribute__((packed)) NameServiceResourceRecordTrailer;
00262
00263 /*
00264       Node Entry Name Flags
00265          g                       - Group name flag
00266                                    0 - Unique name, 1 - Group name
00267          ont                     - Owner node type
00268          drg                     - Deregister flag
00269                                    0 - None, 1 - Name is delete process
00270          cnf                     - Conflict flag
00271                                    0 - None, 1 - Name in conflict
00272          act                     - Active name flag (must be one)
00273          prm                     - Permanent name flag
00274                                    1 - permanent node name, 0 - all others
00275
00276  */
00277 typedef struct _NameFlags
00278 {
00279     uint16_t g : 1;
00280     uint16_t ont : 2;
00281     uint16_t drg : 1;
00282     uint16_t cnf : 1;
00283     uint16_t act : 1;
00284     uint16_t prm : 1;
00285     uint16_t mbz : 9;
00286
00287 } __attribute__((packed)) Name_Flags;
00288
00289 /*
00290       Node Name Entry
00291
00292          name                    - ASCII name
00293          question_type           - Request type (rr_type for answer)
00294          question_class          - Class of request (rr_class for answer)
00295
00296 */
00297 typedef struct _NodeNameEntry
00298 {
00299     char name[NETBIOS_NAME_SIZE_IN_CHARS];
00300 #ifdef NETBIOS_NAME_MICROSOFT_VARIANT
00301     char suffix;
00302 #endif /* #ifdef NETBIOS_NAME_MICROSOFT_VARIANT */
00303     Name_Flags name_flags;
00304
00305 } __attribute__((packed)) NodeNameEntry;
```

```
00306
00307 /*
00308       Name Service Node Status Response Trailer
00309
00310          ttl                     - Time to live for resource name
00311          rdlength                - Bytes in RDATA field
00312          num_names               - Number of names
00313
00314  */
00315 typedef struct _NameServiceNodeStatusResponseTrailer
00316 {
00317     uint32_t ttl;
00318     uint16_t rdlength;
00319     uint8_t num_names;
00320
00321 } __attribute__((packed)) NameServiceNodeStatusResponseTrailer;
00322
00323 /*
00324       Name Service Node Status Response Statistics
00325
00326          unit_id                      - Unique unit id
00327          jumpers                      - jumpers
00328          test_result                  - Test results
00329          version_number               - Version
00330          period_of_statistics         - Period
00331          number_of_crc                - CRCs
00332          number_alignment_errors      - Alignment errors
00333          number_of_collisions         - Collisions
00334          number_send_aborts           - Send aborts
00335          number_good_sends            - Good sends
00336          number_good_receives         - Good receives
00337          number_retransmits           - Re-transmits
00338          number_of_conditions         - Resource conditions
00339          number_free_command_blocks   - Number of free command blocks
00340          total_number_command_blocks  - Number of command blocks
00341          nax_total_command_blocks     - Maxmimum number of command blocks
00342          number_pending_sessions      - Pending sessions
00343          max_number_pending_sessions  - Maximum pending sessions
00344          max_total_pending_sessions   - Maximum total pending sessions
00345          session_data_packet_size     - Session packet size
00346
00347  */
00348 typedef struct _NameServiceNodeStatusResponseStatistics
00349 {
00350     uint8_t unit_id[NETBIOS_UNIT_ID_IN_BYTES];
00351     uint8_t jumpers;
00352     uint8_t test_result;
00353     uint16_t version_number;
00354     uint16_t number_of_crc;
00355     uint16_t period_of_statistics;
00356     uint16_t number_alignment_errors;
00357     uint16_t number_of_collisions;
00358     uint16_t number_send_aborts;
00359     uint32_t number_good_sends;
00360     uint32_t number_good_receives;
00361     uint16_t number_retransmits;
00362     uint16_t number_of_conditions;
00363     uint16_t number_free_command_blocks;
00364     uint16_t total_number_command_blocks;
00365     uint16_t nax_total_command_blocks;
00366     uint16_t number_pending_sessions;
00367     uint16_t max_number_pending_sessions;
00368     uint16_t max_total_pending_sessions;
00369     uint16_t session_data_packet_size;
00370
00371 } __attribute__((packed)) NameServiceNodeStatusResponseStatistics;
00372
00373 /*
00374  ****************************************************************************
00375  *
00376  * NetBIOS callback declaration
00377  *
00378  ****************************************************************************
00379  */
00380 typedef void(UdpNetbiosNameServiceFunc)(PoolPtr poolPtr);
00381 extern UdpNetbiosNameServiceFunc *UdpNetbiosNameServicePtr;
00382
00383 /*
00384  ****************************************************************************
00385  *
00386  * netbios "C" Library Interface
00387  *
00388  ****************************************************************************
00389  */
00390
00391 /*
00392  ****************************************************************************
```

```
00393  *
00394  * Runtime Library Routines
00395  *
00396  ***************************************************************************
00397  */
00398
00399 /*
00400  ***************************************************************************
00401
00402    Convert string into NETBIOS name
00403
00404    Parameters:
00405       netBIOSnamePtr          - Correctly formed name
00406       namePtr                 - Name to convert
00407       netBIOSnameSize         - NetBIOS name (netBIOSnamePtr) size in bytes
00408
00409    Return:
00410       None
00411
00412    Notes:
00413       Name can contain any alphanumeric character except spaces
00414       and \ : / * ? ; | . and it will be truncated if greater than 15
00415       characters. The resulting name is all uppercase all excepted characters
00416       will be set to '0'.
00417
00418  ***************************************************************************
00419  */
00420 void NetbiosConvertName(char *netBIOSnamePtr, const char *namePtr, int netBIOSnameSize);
00421
00422 /*
00423  ***************************************************************************
00424
00425    Get NETBIOS name in ASCII formatted IAW RFC 883
00426
00427    Parameters:
00428       netBIOSnamePtr          - Correctly formed name
00429       netBIOSnameSize         - NetBIOS name size capacity in bytes
00430                                 should be at least
00431                                 NETBIOS_NAME_SIZE_IN_CHARS + 1
00432
00433    Return:
00434       Number of bytes in name
00435
00436    Notes:
00437       Name can contain any alphanumeric character except spaces
00438       and \ : / * ? ; | . and it will be truncated if greater than 15
00439       characters. The resulting name is all uppercase all excepted characters
00440       will be set to '0'.
00441
00442  ***************************************************************************
00443  */
00444 int NetbiosGetNetbiosName(char *netBIOSnamePtr, int netBIOSnameSize);
00445
00446 /*
00447  ***************************************************************************
00448
00449    Enable the NETBIOS name service
00450
00451    Parameters:
00452       name                    - Name to register, NULL pointer disables service
00453       initialRegister         - Send registration packet upon enabling.
00454
00455    Return:
00456       None
00457
00458    Notes:
00459       Enable name service responding to name service queries.
00460       Name can contain any alphanumeric character except spaces
00461       and \ : / * ? ; | . and it will be truncated if greater than 15
00462       characters.
00463
00464  ***************************************************************************
00465  */
00466 void NetbiosEnableNameService(const char *name, BOOL initialRegister);
00467
00468 #endif
```

24.434 netDevice.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NETDEVICE_H_
00006 #define _NETDEVICE_H_
```

```
00007 #include <nettypes.h>
00008 #include <buffers.h>
00009 #include <stdio.h>
00010 #include <device.h>
00011 /*
00012  ****************************************************************************
00013  *
00014  * Include Files
00015  *
00016  ****************************************************************************
00017  */
00018
00019 /*
00020  ****************************************************************************
00021  *
00022  * Debugging
00023  *
00024  ****************************************************************************
00025  */
00026 /* Library debugging switch */
00027 #define NETDEVICE_DEVICE_DEBUG (1)
00028
00029 /*
00030  ****************************************************************************
00031  *
00032  * Constants
00033  *
00034  ****************************************************************************
00035  */
00036 /* Maximum Network Devices */
00037 #define NETDEVICE_DEVICE_MAXIMUM (5)
00038
00039 /*
00040    NetworkAddInterface Returns
00041
00042       NETDEVICE_REQUEST_FAILED        - Was not added by the system
00043       NETDEVICE_MODULE_NOT_FOUND      - Module not detected
00044       NETDEVICE_ALREADY_EXISTS        - Already exists
00045       NETDEVICE_SUPPORT_TASK_FAILED   - Driver task not created
00046
00047 */
00048
00049 #define NETDEVICE_REQUEST_FAILED (0)
00050 #define NETDEVICE_MODULE_NOT_FOUND (-1)
00051 #define NETDEVICE_ALREADY_EXISTS (-2)
00052 #define NETDEVICE_SUPPORT_TASK_FAILED (-3)
00053 #define NETDEVICE_MODULE_NOT_SUPPORTED (-4)
00054
00055 /* List requests */
00056 #define NETDEVICE_SITE_COUNT (1)
00057 #define NETDEVICE_SITE_LIST (2)
00058
00059 /*
00060    Security supplicant actions
00061
00062       NETDEVICE_INITIALIZE             - Initialize
00063       NETDEVICE_UPDATE_KEY             - Update key
00064
00065 */
00066 #define NETDEVICE_INITIALIZE (0)
00067 #define NETDEVICE_UPDATE_PASSPHRASE (1)
00068 #define NETDEVICE_GET_TEMPORAL_KEY (2)
00069
00070 /*
00071  ****************************************************************************
00072  *
00073  * Enumerations
00074  *
00075  ****************************************************************************
00076  */
00077 /*
00078    Network interface controller (NIC)
00079
00080       ControllerNone             - No controller
00081       ControllerPointToPoint     - Point-to-Point data link protocol
00082       ControllerRealtekRtl8711   - Realtek IEEE 802.11g
00083
00084  */
00085 typedef enum _NetDeviceController
00086 {
00087     ControllerNone,
00088     ControllerPointToPoint,
00089     ControllerRealtekRtl8711
00090
00091 } NetDeviceController;
00092
00093 /*
```

```
00094    Network interface controller (NIC) bus type used
00095
00096       BusTypeNone                - No bus
00097       BusTypeMemoryMapped        - External data bus access
00098       BusTypeSdioBusType         - Secure Digital Input Output (SDIO)
00099       BusTypeSdioBusSpiMode      - SDIO using SPI mode
00100       BusTypeSpiBus              - Serial Peripheral Interface (SPI)
00101
00102  */
00103 typedef enum _NetDeviceBusType
00104 {
00105     BusTypeNone,
00106     BusTypeMemoryMapped,
00107     BusTypeSdioBusType,
00108     BusTypeSdioBusSpiMode,
00109     BusTypeSpiBus
00110
00111 } NetDeviceBusType;
00112
00113 /*
00114       Select details (chip select)
00115
00116          Chip select are either managed (asserted/deasserted by the processor
00117          module (e.g. (Q)SPI module) or unmanaged and handled by the driver.
00118
00119          The (Q)SPI module supports connecting additional hardware that using
00120          the (Q)SPI chip selects can select several devices (multiplex) up to 15.
00121
00122       SelectDetailNone                    - No chip select
00123       SelectDetailSpiUnmanaged            - Single signal unmanaged chip select
00124       SelectDetailSpiUnmanagedMultiplex   - Multiplexed unmanaged chip select
00125       SelectDetailSpiManaged              - Single signal managed chip select
00126       SelectDetailSpiManagedMultiplex     - Multiplexed managed chip select
00127       SelectDetailGpioUnmanaged           - Single signal unmanaged chip select
00128
00129       Changing the ordering of these parameters require complete system and platform
00130       rebuilds.
00131
00132 */
00133
00134 typedef enum _NetDeviceSelectDetail
00135 {
00136     SelectDetailNone,
00137     SelectDetailSpiUnmanaged,
00138     SelectDetailSpiUnmanagedMultiplex,
00139     SelectDetailSpiManaged,
00140     SelectDetailSpiManagedMultiplex,
00141     SelectDetailGpioUnmanaged,
00142
00143 } NetDeviceSelectDetail;
00144
00145 /*
00146  ****************************************************************************
00147  *
00148  * C++ definitions
00149  *
00150  ****************************************************************************
00151  */
00152
00153 /*
00154  ****************************************************************************
00155  *
00156  * Routine Prototypes
00157  *
00158  ****************************************************************************
00159  */
00160 /* Device management */
00161 typedef void(NetDeviceProgressFn)(int activity, int progress, const char *detail);
00162 typedef NetDeviceProgressFn *NetDeviceProgressFnPtr;
00163
00164 /* Security supplicant */
00165 typedef BOOL(NetDeviceSupplicantFn)(int action, void *buffer);
00166 typedef NetDeviceSupplicantFn *NetDeviceSupplicantFnPtr;
00167
00168 /*
00169  ****************************************************************************
00170  *
00171  * Classes
00172  *
00173  ****************************************************************************
00174  */
00175 /*
00176  ****************************************************************************
00177  * Network Device Base Class
00178  ****************************************************************************
00179  */
00180 class NetDevice
```

```
00181 {
00182    public:
00183     /*** Constructor ***/
00184     NetDevice(int irq, NetDeviceController controller, NetDeviceBusType busType, int select,
      NetDeviceSelectDetail selectDetails);
00185
00186     /*** Destructor ***/
00187     virtual ~NetDevice();
00188
00189     /*** Copy Constructor ***/
00190     /* Default */
00191
00192     /*** Static Methods ***/
00193     /* Verify platform support, create controller object */
00194     static int add(int irq, NetDeviceController controller, NetDeviceBusType busType, int select,
      NetDeviceSelectDetail selectDetails);
00195
00196     /* Get network device  */
00197     static NetDevice *getNetDevice(int deviceIndex);
00198
00199     /*** Methods ***/
00200     /* Setup bus for controller */
00201     virtual BOOL setupBus(void);
00202
00203     /* Setup ISR */
00204     virtual BOOL setupIsr(DeviceIsrFnPtr isrFunction, int irq);
00205
00206     /* Determine existence of controller on bus */
00207     virtual BOOL probe(void);
00208
00209     /* Initialize controller in idle state, add to network stack */
00210     virtual BOOL open(void);
00211
00212     /* Reset controller to quiescent idle state */
00213     virtual BOOL close(void);
00214
00215     /* Remove controller from network stack */
00216     virtual BOOL remove(void);
00217
00218     /* Configure and retrieve settings */
00219     virtual BOOL configure(int request, ssize_t parameterSize, void *parameterPtr);
00220     virtual BOOL retrieveSetting(int setting, ssize_t settingSize, void *settingPtr);
00221
00222     /* List */
00223     virtual BOOL list(int request, ssize_t &listSize, void *listPtr);
00224
00225     /* Network name */
00226     virtual void getNetworkName(char *name, ssize_t maximumBytes);
00227
00228     /* Survey network */
00229     virtual BOOL surveyNetwork(char *site);
00230
00231     /* Connect */
00232     virtual BOOL connect(const char *identifier = NULL, const char *securityKey = NULL, int
      securityDetail = 0, int mode = 0);
00233
00234     /* Disconnect */
00235     virtual BOOL disconnect(void);
00236
00237     /*** Network stack routines ***/
00238     /* Transmit packet */
00239     virtual BOOL transmitPacket(PoolPtr poolPtr);
00240
00241     /* Set/Reset multicast */
00242     virtual void resetMulticast(MACADDRESS_48 macAddress, BOOL addAddress);
00243
00244     /*** Accessors ***/
00245     /* Access stack interface number */
00246     void setInterfaceNumber(int interfaceNumber);
00247     int getInterfaceNumber(void) const;
00248
00249     /* Get number */
00250     int getNumber(void) const;
00251
00252     /* Get name */
00253     char *getName(void);
00254
00255     /* Get unit */
00256     int getUnit(void) const;
00257
00258     /* Get controller */
00259     NetDeviceController getController(void) const;
00260
00261     /* Get bus type */
00262     NetDeviceBusType getBusType(void) const;
00263
00264     /* Get select */
```

```
00265     int getSelect(void) const;
00266
00267     /* Get select */
00268     NetDeviceSelectDetail getSelectDetail(void) const;
00269
00270     /* Get MAC address */
00271     virtual BOOL getMacAddress(MACADDRESS_48 &macAddress);
00272
00273     /* Connected? */
00274     virtual BOOL isConnected(void);
00275
00276     /* Progress routine */
00277     virtual void setProgressRoutine(NetDeviceProgressFnPtr routinePtr);
00278
00279     /* Security supplicant routine */
00280     virtual void setSupplicant(NetDeviceSupplicantFnPtr routinePtr);
00281
00282     /* Set/Get ticks per second */
00283     void setTicksPerSecond(unsigned short ticksPerSecond);
00284     unsigned int getTicksPerSecond(void) const;
00285
00286     /* Signal interrupt */
00287     virtual void signalInterrupt(void);
00288
00289     /* Valid? */
00290     BOOL isValid(void) const;
00291
00292    protected:
00293     /* Set name */
00294     void setName(const char *namePtr);
00295
00296     /* Set unit */
00297     void setUnit(int unit);
00298
00299     /* Set valid */
00300     void setValid(BOOL valid) const;
00301
00302    private:
00303     /*** Methods ***/
00304     /* None */
00305
00306     /*** Data Members ***/
00307     /* Index in __device */
00308     int __index;
00309
00310     /* Name */
00311     char *__namePtr;
00312
00313     /* Unit */
00314     int __unit;
00315
00316     /* IRQ */
00317     int __irq;
00318
00319     /* Controller */
00320     NetDeviceController __controller;
00321
00322     /* Bus */
00323     NetDeviceBusType __busType;
00324
00325     /* Select */
00326     int __select;
00327
00328     /* Select */
00329     NetDeviceSelectDetail __selectDetail;
00330
00331     /* Interface number */
00332     int __interfaceNumber;
00333
00334     /* Ticks per second (Can be modified by developer) */
00335     unsigned short __ticksPerSecond;
00336
00337     /* Valid */
00338     BOOL __valid;
00339
00340     /*** Static Data Members ***/
00341     /* One time initialization */
00342     static BOOL __isInitialized;
00343
00344     /* Critical section for mutex */
00345     static OS_CRIT __criticalSection;
00346
00347     /* Interface objects */
00348     static NetDevice *__device[NETDEVICE_DEVICE_MAXIMUM];
00349 };
00350
00351 /*
```

```
00352  ***************************************************************************
00353
00354    Verify availability and support for device.
00355
00356
00357       Parameters:
00358          irq         - Fixed level interrupt source
00359          controller  - IEEE 802.11g controller
00360          busType     - Bus used by the controller
00361
00362
00363       Return:
00364          TRUE - Support and available on module, FALSE option not supported
00365
00366        Note:
00367          Does not verify controller is attached and operational.
00368
00369  ***************************************************************************
00370  */
00371 BOOL NetDeviceIsValid(int irq, NetDeviceController controller, NetDeviceBusType busType);
00372
00373 /*
00374  ***************************************************************************
00375
00376    Get device handle
00377
00378
00379       Parameters:
00380          irq         - Fixed level interrupt source
00381          controller  - IEEE 802.11g controller
00382          busType     - Bus used by the controller
00383          select      - Chip select is bus specific
00384                        SDIO using SPI mode (BusTypeSdioBusSpiMode)
00385                        0 - 3 (Q)SPI chip select lines QSPI_CS[0:3]
00386
00387       Return:
00388          Handle - Supported and available on module,
00389                NULL interface not supported
00390
00391        Note:
00392          Verify parameters and platform support for device.
00393          Does not verify controller is attached and operational.
00394
00395  ***************************************************************************
00396  */
00397 NetDevice *NetDeviceGetDevice(int irq,
00398                               NetDeviceController controller,
00399                               NetDeviceBusType busType,
00400                               int select,
00401                               NetDeviceSelectDetail selectDetails);
00402
00403 /* Device Realtek RTL8711 Controller on SDIO bus in SPI mode */
00404 NetDevice *NetDeviceGetDeviceRtl8711SdioSpi(int irq, int select, NetDeviceSelectDetail selectDetails);
00405
00406 #endif /* _NETDEVICE_H_ */
```

24.435 netinterface.h File Reference

NetBurner Network Interface Header File.

```
#include <predef.h>
#include <autoip.h>
#include <config_netobj.h>
#include <config_obj.h>
#include <dhcpclient.h>
#include <discoveryservlet.h>
#include <mDNS.h>
#include <nettypes.h>
#include <buffers.h>
```

Classes

- class InterfaceBlock

  Network interface configuration block class for interface control and configuration.

Functions

- int Removeinterface (int interface)

  Remove a network interface from the system.
- void EnableMulticast (MACADR macAddress, int interface=0)

  Enable Multicast on an existing interface.
- void DisableMulticast (MACADR macAddress, int interface=0)

  Disable Multicast on an existing interface.
- InterfaceBlock * GetInterfaceBlock (int interface=0)

  Get an InterfaceBlock control and configuration object.
- int32_t GetFirstInterface (void)

  Returns the Interface Number of the first registered network interface.
- int32_t GetNextInterface (int lastInterface)

  Returns the Interface Number of the next registered network interface.
- int32_t GetInterfaceNumber (InterfaceBlock *pifb)

  Returns the Interface Number of the specified network interface InterfaceBlock.
- int32_t GetInterfaceForMyAddress4 (IPADDR4 ip)

  Returns the Interface Number of the specified network interface IPv4 address.
- bool GetInterfaceLink (int ifn)

  Returns the network interface link status.
- MACADR InterfaceMAC (int interface)

  Returns the MAC address of the specified network interface.
- bool InterfaceLinkActive (int interface)

  Returns the link status of the specified network interface.
- int InterfaceLinkSpeed (int interface)

  Returns the 10/100 link speed of the specified network interface.
- bool InterfaceLinkDuplex (int interface)

  Returns the full or half link duplex of the specified network interface.
- const char * InterfaceName (int interface)

  Returns the interface name of the specified network interface.
- IPADDR4 InterfaceIP (int interface)

  Returns the IPv4 IP address of the specified network interface.
- IPADDR4 InterfaceAutoIP (int interface)

  Returns the IPv4 IP AutoIP address of the specified network interface.
- IPADDR4 InterfaceDNS (int interface)

  Returns the IPv4 DNS address of the specified network interface.
- IPADDR4 InterfaceDNS2 (int interface)

  Returns the second IPv4 DNS address of the specified network interface.
- IPADDR4 InterfaceMASK (int interface)

  Returns the IPv4 network mask of the specified network interface.
- IPADDR4 InterfaceGate (int interface)

  Returns the IPv4 gateway address of the specified network interface.

24.435.1 Detailed Description

NetBurner Network Interface Header File.
All network interfaces and functions are accessed through the network interface API including:

- Ethernet
- Wifi
- AutoIP
- Multihome
- PPP

24.436 netinterface.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00040 #ifndef _NB_NETIF_H
00041 #define _NB_NETIF_H
00042
00043 #include <predef.h>
00044 #include <autoip.h>
00045 #include <config_netobj.h>
00046 #include <config_obj.h>
00047 #include <dhcpclient.h>
00048 #include <discoveryservlet.h>
00049 #include <mDNS.h>
00050 #include <nettypes.h>
00051 #include <buffers.h>
00052
00053
00054 /*-------------------------------------------------------------------------
00055  * Definitions
00056  *-------------------------------------------------------------------------*/
00057
00058 //  PPP Interface Identifier
00059 #define ARP_PPP_INTERFACE (255)
00060
00061 /*-------------------------------------------------------------------------
00062  *
00063  * MAC Address for second and subsequent interfaces
00064  *    For a second and subsequent NICs the MSB(it) of the 5th octet is XOR'ed
00065  *    with 1 to support prior released tools.
00066  *
00067  *                              6 bytes
00068  *   <---------------------------------------------------------------------->
00069  *        0           1           2           3           4           5
00070  *   ----------------------------------------------------------------------
00071  *  | 6th byte  | 5th byte  | 4th byte  | 3rd byte  | 2nd byte  | 1st byte  |
00072  *  | 1st octet | 2nd octet | 3rd octet | 4th octet | 5th octet | 6th octet |
00073  *   ----------------------------------------------------------------------
00074  *
00075  *                3 bytes                           3 bytes
00076  *   <---------------------------------> <--------------------------------->
00077  *   ----------------------------------------------------------------------
00078  *  | Organizationally Unique Identifier |   Network Interface Controller    |
00079  *  |           |   (OUI)   |           |           |  (NIC)    |           |
00080  *  |    0x00   |    0x03   |   0xF4    |    0xYX   |   0xZX    |   0xXX    |
00081  *   ----------------------------------------------------------------------
00082  *  |            \   OUI is NetBurner's      Y is IIXX XXXX binary
00083  *  |             \                          I is the interface defined as
00084  *  |              \                            bits in NB_MAX_NIC... below
00085  *  |               |                                 Z is WXXX binary
00086  *  V               V                                   W ix XOR'd bit
00087  *  |1|2|3|4|5|6|7|8| (7) 0 global, 1 local, (8) 0 unicast, 1 multicast (IEE)
00088  *
00089  * Notes:
00090  *    0-5 are C language offsets, 6th byte through 1st byte is Ethernet
00091  *    transmission order and octet notation is from IEEE 802 MAC-48 address.
00092  *    Derivation is needed if the NIC does not have a non-volatile MAC address.
00093  *
00094  *-------------------------------------------------------------------------*/
00095 #ifndef NB_MAC_ADDRESS
00096 #define NB_MAC_ADDRESS
00097
00098 /* 4th Octet */
00099 #define NB_MAC_NIC_INTERFACE        (0x00)
00100 #define NB_MAC_NIC_SECOND_INTERFACE (0x80)
00101 #define NB_MAC_NIC_THIRD_INTERFACE  (0x40)
00102 #define NB_MAC_NIC_FOURTH_INTERFACE (0xC0)
00103 #define NB_MAC_NIC_SECOND_INTERFACE_MASK \
00104     (NB_MAC_NIC_INTERFACE | NB_MAC_NIC_SECOND_INTERFACE | NB_MAC_NIC_THIRD_INTERFACE |
      NB_MAC_NIC_FOURTH_INTERFACE)
00105 #define NB_MAC_OCTET_4      (3)
00106 #define NB_MAC_OCTET_5      (4)
00107 #define NB_MAC_OCTET_5_XOR  (0x80)
00108
00109 #endif /* NB_MAC_ADDRESS */
00110
00111
00112 /*-------------------------------------------------------------------------
00113  * Callback Function Prototypes
00114  *-------------------------------------------------------------------------*/
00115
00116 /*
00117  ***************************************************************************
```

```
00118
00119    Process Internet Protocol Packet (IP)
00120
00121    Parameters:
00122       poolPtr           - Packet from pool
00123       ethernetFramePtr  - Ethernet frame
00124       checksum          - IP frame checksum
00125
00126    Return:
00127       None
00128
00129    Notes;
00130       Called for every IP packet received and meets this criteria.
00131          1)The packet is an IP packet.
00132          2)The IP header checksum is valid.
00133          3)The packet was addressed to us or to a broadcast address.
00134       Before the callback is called the flags member of the buffer is setup to
00135       indicate physical broadcast etc....
00136
00137  ***************************************************************************
00138  */
00139 typedef void(ProcessPacketFunc)(PoolPtr poolPtr, PEFRAME ethernetFramePtr, uint16_t checksum);
00140
00141 /*
00142  ***************************************************************************
00143
00144    Process Address Resolution Packet (ARP)
00145
00146    Parameters:
00147       poolPtr           - Packet from pool
00148       ethernetFramePtr  - Ethernet frame
00149
00150    Return:
00151       None
00152
00153    Notes;
00154       Called for every ARP packet received. Before the callback is called the
00155       flags member of the buffer is setup to indicate physical broadcast etc...
00156
00157  ***************************************************************************
00158  */
00159 typedef void(ProcessArpFunc)(PoolPtr poolPtr, PEFRAME ethernetFramePtr);
00160
00161 /*
00162  ***************************************************************************
00163
00164    Process Address LLDP
00165
00166    Parameters:
00167       poolPtr           - Packet from pool
00168       ethernetFramePtr  - Ethernet frame
00169
00170    Return:
00171       None
00172
00173    Notes;
00174       If filled in called for every LLDP frame
00175       flags member of the buffer is setup to indicate physical broadcast etc...
00176
00177  ***************************************************************************
00178  */
00179 typedef void (*ProcessLLDPptr)(PoolPtr poolPtr);
00180
00181 extern ProcessLLDPptr pProcessLLDP;
00182
00183 /*
00184  ***************************************************************************
00185
00186    Returns current link active status
00187
00188    Parameters:
00189       None
00190
00191    Return:
00192       TRUE active, FALSE all else.
00193
00194    Notes:
00195       Interface driver must minimize latency.
00196
00197  ***************************************************************************
00198  */
00199 typedef BOOL(LinkActiveFunc)(void);
00200
00201
00202 /*
00203  ***************************************************************************
00204  *
```

```
00205  * Data Structures
00206  *
00207  ******************************************************************************
00208  */
00209
00210 #define INTF_CSUM_IP    (0x01)
00211 #define INTF_CSUM_ICMP  (0x02)
00212 #define INTF_CSUM_UDP   (0x04)
00213 #define INTF_CSUM_TCP   (0x08)
00214 #define INTF_CSUM_IP6   (0x10)
00215 #define INTF_CSUM_ICMP6 (0x20)
00216 #define INTF_CSUM_UDP6  (0x40)
00217 #define INTF_CSUM_TCP6  (0x80)
00218
00219 #define INTF_CSUM_ALL4 (INTF_CSUM_IP | INTF_CSUM_ICMP | INTF_CSUM_UDP | INTF_CSUM_TCP)
00220 #define INTF_CSUM_ALL6 (INTF_CSUM_IP6 | INTF_CSUM_ICMP6 | INTF_CSUM_UDP6 | INTF_CSUM_TCP6)
00221
00222
00223 /*
00224    Interface Configuration and Control Block
00225
00226       Ethernet
00227          theMac            - Interface ethernet MAC address
00228
00229       TCP/IP
00230          netIP             - Interface IP address
00231          netIpMask         - Interface network IP mask
00232          netIpGate         - Interface gateway IP address
00233          netMss            - Interface maximum segment size (0 default)
00234
00235       Network Processing
00236          send_func         - Transmit packet function
00237          kill_if           - Kill interface function
00238          enab_multicast    - Enable/Disable multicast addresses
00239
00240       Identification
00241          InterfaceName     - Name
00242          config_num        - Configuration record number
00243
00244       Multi-Home Feature
00245          bMultiHome        - Multi-home?
00246          root_if           - Physical interface number
00247
00248       Routing Support
00249          LinkActiveFuncPtr - Link active function
00250
00251       Accelerator Support
00252           checksumOffload  - Bitfield indicating TX checksums the
00253                              interface automatically generates
00254
00255    Notes:
00256       Should not change data sizes and always add to the end to avoid breaking
00257       use for interface configuration.
00258  */
00259
00260
00261 class InterfaceBlock : public config_obj
00262 {
00263    public:
00264     I4Record ip4{"IPv4"};   // The IPv4 configuration for "this" interface, see config_netobj.h for
      details
00265 #ifdef IPV6
00266     I6Record ip6{"IPv6"};   // The IPv6 configuration for "this" interface, see config_netobj.h for
      details
00267 #endif
00268     config_MACADR MAC{"00:00:00:00:00:00", "MAC"};            // Interface MAC address. May be 0/null
      for interfaces such as PPP
00269     config_string device_name{"", "DeviceName"};             // Name of interface. Used to register
      for DNS and NetBIOS
00270     config_int    dhcp_discover_secs{1, "DhcpDiscoverSec"};   // How long to wait after boot before
      sending a DHCP Discover message
00271     config_string discovery_server{"discover.netburner.com/DevicePost", "DiscoveryReportUrl"};   //
      NetBurner discover server name
00272     config_int    discovery_interval{(15 * 60), "DiscoveryReportInterval"};   // How often to report
      to NetBurner discover server
00273     config_bool   obfuscate_discovery{true, "DiscoveryObfuscate", "Should discovery obfuscate data"};
00274     config_bool   supress_default_responses{false, "SupressDefault", "Supress ping and udp Echo
      ports"};
00275     config_int    vlan_tag_value{0, "VlanTag"};
00276
00277     ConfigEndMarker;   // Marker for end of configuration structure
00278
00279     int32_t    root_if;             // Root interface number. If multihomed, the interface would be a
      child of the root interface
00280     int32_t    my_ifnum;            // This interface number
00281     uint16_t   netMss;              // Network max segment size for this interface
00282     uint8_t    checksumOffload;     // Non-zero if hardware interface support offloading checksum
```

```
    calculation
00303     bool       bHaveSeenLink;       // Have seen a link at once before
00304     DhcpObject dhcpClient{this};    // DHCP client support
00305     discover_servlet disco_servlet{this};   // the servlet that runs the NetBurner discovery server
    process
00306     void (*MCastLinkNotify)(InterfaceBlock *pBlock, bool bLink);   // A funciton pointer called when
    link changes for multicast.
00307
00308     IPADDR4 previous_addr;
00309     IPADDR4 previous_mask;
00310     IPADDR4 previous_gate;
00311
00312 #ifdef AUTOIP                   // if not defined don't compile anything...
00313     AutoIPClient AutoClient;    // AutoIP client object for this interface
00314 #endif
00315
00316 #if defined MULTIHOME
00317     BOOL bMultiHome;           // True if this is a multihomed interface
00318     InterfaceBlock *pChild;    // Parrent points to list, children are linked list of pchilds.
00319 #endif
00320
00327     InterfaceBlock(const char *name, const char *desc = NULL);
00328
00336     InterfaceBlock(config_obj &owner, const char *name, const char *desc = NULL);
00337
00345     void RegisterInterface();
00346
00347     virtual void send_func(PoolPtr poolPtr) = 0;
00348     virtual void kill_if();
00349
00360     virtual void EnableMulticast(MACADR macAddress, BOOL addAddress) = 0;
00361
00369     virtual bool LinkActive() = 0;
00370
00378     virtual int LinkSpeed() = 0;
00379
00387     virtual bool LinkDuplex() = 0;
00388
00394     void InterfaceLinkChange(bool link);
00395
00403     virtual const char *GetInterfaceName();
00404
00412     int GetInterfaceNumber() { return my_ifnum; }
00413
00419     inline bool IsRootInterface() { return root_if == 0; }
00420
00421     // Move values from the IPV4 settings to cur stuff.
00422     // Starts /Shutdowns DHCP if necessary
00423     // Stores previous values in  previous_addr,_mask,_gate;
00424     // Returns true if the address changed.
00425     bool ProcessIPConfigChange(uint16_t ticks_to_wait);
00426
00427     //Used if the web page changes IP values to swap so one can complete socket.
00428     void SwapOldAndCurrentAddr();
00429
00430     virtual bool bNeedsArp() { return true; };
00431     void fdShowInterfaceValues(int fd);
00432     inline void ShowInterfaceValues() { fdShowInterfaceValues(1); };
00433     void SetDefaultFlags();
00434     // Called by DHCP client when we get new settings...
00435     void DHCPNotify();
00436
00437 #if defined MULTIHOME
00438     int RegisterMultiHomeInterface(InterfaceBlock &UnderBlock);
00439 #endif
00440 };
00441
00442
00443
00444 // Adds additional features to an InterfaceBlock. Defaults to most features disabled
00445 class EtherLikeInterface : public InterfaceBlock
00446 {
00447    public:
00448     ConfigEndMarker;
00449     EtherLikeInterface(const char *name, const char *desc = NULL) : InterfaceBlock(name, desc)
00450     {
00451         ip4.SetFlag(fConfigHidden);
00452 #ifdef IPV6
00453         ip6.SetFlag(fConfigHidden);
00454 #endif
00455         MAC.SetFlag(fConfigHidden);
00456         SetFlag(fConfigHidden);
00457     }
00458
00459     EtherLikeInterface(config_obj &owner, const char *name, const char *desc = NULL) :
    InterfaceBlock(owner, name, desc)
00460     {
```

```
00461         ip4.SetFlag(fConfigHidden);
00462 #ifdef IPV6
00463         ip6.SetFlag(fConfigHidden);
00464 #endif
00465         MAC.SetFlag(fConfigHidden);
00466         SetFlag(fConfigHidden);
00467     }
00468
00469     virtual void Enable()
00470     {
00471         ClrFlag(fConfigHidden);
00472         ip4.ClrFlag(fConfigHidden);
00473 #ifdef IPV6
00474         ip6.ClrFlag(fConfigHidden);
00475 #endif
00476         MAC.ClrFlag(fConfigHidden);
00477         MAC.SetFlag(fConfigReadOnly);
00478     }
00479 };
00480
00481 /* Global receive processing definition */
00482 int NetDoRX(PoolPtr pp, uint16_t count, int if_num);
00483
00484 /*
00485  ****************************************************************************
00486  *
00487  * External Definitions
00488  *
00489  ****************************************************************************
00490  */
00491 /*
00492    System ARP Process Function
00493 */
00494 extern ProcessArpFunc *pArpFunc;
00495
00496 #ifdef IPV6
00497
00498 /*
00499  ****************************************************************************
00500
00501    IP6  Network Packet
00502
00503    Parameters:
00504       poolPtr           - Packet from pool
00505
00506    Return:
00507       None
00508
00509  ****************************************************************************
00510  */
00511 typedef void(ProcessIp6Func)(PoolPtr poolPtr);
00512
00513 extern ProcessIp6Func *pIp6Func;
00514
00515 #endif
00516
00517 /*
00518    System IP Packet Process Function
00519 */
00520 extern ProcessPacketFunc *pPacketfunc;
00521
00522 /*
00523  ****************************************************************************
00524  *
00525  * Routine Prototypes
00526  *
00527  ****************************************************************************
00528  */
00529
00530 /*
00531  ****************************************************************************
00532
00533    Selects Packet Transmission for the Interface
00534
00535    Parameters:
00536       poolPtr        - Packet from pool
00537       interface      - Interface number
00538
00539
00540    Return:
00541       None
00542
00543    Notes:
00544       Frees buffer if interface is not valid
00545
00546  ****************************************************************************
00547  */
```

```
00548 void TransmitBuffer(PoolPtr poolPtr, int interface);
00549
00550 /*
00551  ***************************************************************************
00552
00553    Extracts Data (Ethernet Frame) Pointer from Pool Packet
00554
00555    Parameters:
00556       poolPtr        - Packet from pool
00557
00558
00559    Return:
00560       Pointer to data segment (ethernet frame) of pool packet.
00561
00562    Notes:
00563       Does not check for validity
00564
00565  ***************************************************************************
00566  */
00567 inline PEFRAME GetEframe(PoolPtr pp)
00568 {
00569     return (PEFRAME)pp->pData;
00570 }
00571
00572 inline PVLEFRAME GetVLEframe(PoolPtr pp)
00573 {
00574     return (PVLEFRAME)pp->pData;
00575 }
00576
00577
00587 int Removeinterface(int interface);
00588
00589
00600 void EnableMulticast(MACADR macAddress, int interface = 0);
00601
00613 void DisableMulticast(MACADR macAddress, int interface = 0);
00614
00615
00624 InterfaceBlock *GetInterfaceBlock(int interface = 0);
00625
00626
00635 int32_t GetFirstInterface(void);
00636
00637
00645 int32_t GetNextInterface(int lastInterface);
00646
00647
00657 int32_t GetInterfaceNumber(InterfaceBlock *pifb);
00658
00659
00667 int32_t GetInterfaceForMyAddress4(IPADDR4 ip);
00668
00669
00679 bool GetInterfaceLink(int ifn);
00680
00681
00682 /*
00683  ***************************************************************************
00684  *
00685  * Runtime Libraries Routine Prototypes
00686  *
00687  ***************************************************************************
00688  */
00689
00690 /*
00691  ***************************************************************************
00692
00693    Initialize the first Ethernet Interface
00694
00695    Parameters:
00696       processPacketFuncPtr    - IP Packet process function
00697       processArpFuncPtr       - ARP Packet process function
00698
00699
00700    Return:
00701       TRUE success, else problems
00702
00703    Notes:
00704       Should be called once in the system to create first interface.
00705
00706  ***************************************************************************
00707  */
00708 BOOL InitializeNetwork(ProcessPacketFunc *processPacketFuncPtr, ProcessArpFunc *processArpFuncPtr);
00709
00710 /*
00711  ***************************************************************************
00712
```

```
00713    Disables network.
00714
00715    Parameters:
00716       None
00717
00718    Return:
00719       None
00720
00721    Notes:
00722       None
00723
00724  ***************************************************************************
00725  */
00726 void StopNetworks(void);
00727
00728 /*
00729  ***************************************************************************
00730
00731    Get current network settings.
00732
00733    Parameters:
00734       interface      - Interface number
00735
00736    Return:
00737       Specific setting requested for the interface, 0 if invalid interface.
00738
00739    Notes:
00740       None
00741
00742  ***************************************************************************
00743  */
00744
00752 MACADR InterfaceMAC(int interface);
00753
00763 bool InterfaceLinkActive(int interface);
00764
00765
00775 int InterfaceLinkSpeed(int interface);
00776
00777
00787 bool InterfaceLinkDuplex(int interface);
00788
00798 const char *InterfaceName(int interface);
00799
00800
00801 void UnWrapVlan(PoolPtr pp, int len);
00802
00803
00804
00805
00826 IPADDR4 InterfaceIP(int interface);
00827
00835 IPADDR4 InterfaceAutoIP(int interface);
00836
00844 IPADDR4 InterfaceDNS(int interface);
00845
00853 IPADDR4 InterfaceDNS2(int interface);
00854
00862 IPADDR4 InterfaceMASK(int interface);
00863
00871 IPADDR4 InterfaceGate(int interface);
00872
00879 #endif /* _NB_NETIF_H */
00880
00881
```

24.437 netrx.h File Reference

Header file for callback functions to add custom Ethenet handlers and pass Ethernet frames in the bottom of the TCP/IP stack.

```
#include <predef.h>
#include <buffers.h>
```

Typedefs

- typedef int(∗ netDoRXFunc) (PoolPtr, uint16_t, int)

  Typedef interface for all network rx processing functions.

Functions

- netDoRXFunc SetCustomNetDoRX (netDoRXFunc customFunc)

  Registers a new custom ethernet handler to run prior to the primary handler.
- netDoRXFunc ClearCustomNetDoRX ()

  Clears the custom ethernet handler, resetting the handler to NULL.
- int NetDoRX (PoolPtr pp, uint16_t ocount, int if_num)

  Entry function for Ethernet frames into the system TCP/IP stack.

24.437.1 Detailed Description

Header file for callback functions to add custom Ethenet handlers and pass Ethernet frames in the bottom of the TCP/IP stack.

24.438 netrx.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00028 #ifndef _NETRX_H
00029 #define _NETRX_H
00030
00031 #include <predef.h>
00032 #include <buffers.h>
00033
00034 #ifdef ALLOW_CUSTOM_NET_DO_RX
00050 typedef int (*netDoRXFunc)(PoolPtr, uint16_t, int);
00051
00052 extern netDoRXFunc CustomNetDoRX;
00053
00061 inline netDoRXFunc SetCustomNetDoRX(netDoRXFunc customFunc)
00062 {
00063     netDoRXFunc ret = CustomNetDoRX;
00064     CustomNetDoRX = customFunc;
00065     return ret;
00066 }
00067
00073 inline netDoRXFunc ClearCustomNetDoRX()
00074 {
00075     return SetCustomNetDoRX(NULL);
00076 }
00077 #endif
00078
00088 int NetDoRX(PoolPtr pp, uint16_t ocount, int if_num);
00089
00090 #endif /* ----- #ifndef _NETRX_H  ----- */
00091
00092
```

24.439 nettimer.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /* Definitions for various IP definitions and structures. */
00006 #ifndef _NB_NETTIMER_H
00007 #define _NB_NETTIMER_H
00008 #include <buffers.h>
00009 #include <utils.h>
00010 //#define  NETTIMER_DIAG (1)
00011
00012 class TimeOutManager;
00013
00014 class TimeOutElement
00015 {
00016    private:
00017     TimeOutElement *volatile pNext;
00018
00019     TimeOutElement *volatile pPrev;
00020
00021 protected:
00022     TimeOutManager *volatile pOwner;
```

```
00023
00024     volatile uint32_t NextTime;
00025     uint32_t Interval;
00026
00027 #ifdef NETTIMER_DIAG
00028     uint32_t seq_check;
00029     uint32_t sentinel;
00030 #endif
00031
00032    public:
00033     virtual void TimeElementEvent() = 0;
00034     TimeOutElement();    // Constructor
00035     ~TimeOutElement();   // Destructor
00036     inline uint32_t GetNextTime() { return NextTime; };
00037     inline void ChangeNextInterval(uint32_t nt) { Interval = nt; };
00038     inline uint32_t GetInterval() { return Interval; }
00039     void SetNextTime(uint32_t NextTime);
00040     friend class TimeOutManager;
00041 } __attribute__((packed));
00042
00043 class TimeOutManager
00044 {
00045     OS_CRIT ManagerCrit;
00046     TimeOutElement *volatile pElHead;
00047     TimeOutElement *volatile pElTail;
00048     volatile uint32_t NextTime;
00049     volatile uint32_t LastRun;
00050     OS_TCB * OwningTaskTcb;
00051
00052     void Remove(TimeOutElement &te);
00053     void Insert(TimeOutElement &te);
00054     void CheckTime(TimeOutElement &te);
00055
00056 #ifdef NETTIMER_DIAG
00057     void CheckList(const char *msg, TimeOutElement *pte);
00058     void DumpList(uint32_t n);
00059     volatile uint32_t entries;
00060 #endif
00061    public:
00062     TimeOutManager();   // Constructor
00063
00064     void InitTaskOwner();
00065
00066 #ifdef NETTIMER_DIAG
00067     void Report();
00068 #endif
00069
00070     // Cause a repetitive event to happen every TickInterval ticks...
00071     // OK to spread allows the system to spread out things like 1 sec ticks uniformly among tick
      intervals.
00072     void RegisterInterval(TimeOutElement &te, uint16_t TickInterval, const char *fromwhere, bool
      OkToSpread = false);
00073
00074     // Cause an event to trigger at a specific time.
00075     void RegisterTriggerAt(TimeOutElement &te, uint32_t TriggerTime);
00076
00077     void RegisterTriggerAt(TimeOutElement &te, TickTimeout & tt);
00078
00079
00080     // Remove either a At or interval event from the time manager...
00081     void DeRegister(TimeOutElement &te);
00082
00083     // Returns the time ticks till the next event...
00084     // Ok to call this function when no events are due...
00085
00086     uint32_t ProcessEvents();
00087
00088     void CoreProcessEvents();
00089 };
00090
00091 inline void TimeOutElement::SetNextTime(uint32_t NextTime)
00092 {
00093     if (pOwner) pOwner->RegisterTriggerAt(*this, NextTime);
00094 }
00095
00096
00097
00098 typedef void(ActionFunction)();
00099
00100 extern TimeOutManager NetTimeOutManager;
00101
00102 class IntervalAction : public TimeOutElement
00103 {
00104    private:
00105     ActionFunction *m_pFunc;
00106
00107    public:
```

```
00108     IntervalAction(ActionFunction *pf);
00109     virtual void TimeElementEvent();
00110 };
00111
00112 typedef void(DHCPProcessFunction)(PoolPtr p);
00113 extern DHCPProcessFunction *pDHCPProcessFunction;
00114
00115 typedef void(MULTICastProcessFunction)(PoolPtr p, uint16_t csum);
00116 extern MULTICastProcessFunction *pMultiCastFunc;
00117
00118 #ifdef IPV6
00119 #include <udp.h>
00120 typedef void(DHCPv6ProcessFunction)(UDPPacket *pkt);
00121 extern DHCPv6ProcessFunction *pDHCPv6ClientProcessFunction;
00122 #endif
00123 #endif
```

24.440 nettypes.h File Reference

NetBurner IPADDR4 Class. See the IPADDR4 Class page for complete documentation.

```
#include <predef.h>
#include <basictypes.h>
#include <ipv6/ipv6_addr.h>
```

Classes

- class MACADR

  Used to store and manipulate MAC addresses.
- class IPADDR4

  Used to store and manipulate IPv4 addresses in dual stack mode.

Typedefs

- typedef class MACADR MACADR

  Used to store and manipulate MAC addresses.
- typedef IPADDR6 IPADDR

  IPADDR Object Type (either v4 or v6)

Functions

- bool operator== (const MACADR &i, const MACADR &j)

  Check MAC equality.
- bool operator!= (const MACADR &i, const MACADR &j)

  Check MAC inequality.
- bool operator> (const MACADR &i, const MACADR &j)

  Check MAC greater than.

24.440.1 Detailed Description

NetBurner IPADDR4 Class. See the IPADDR4 Class page for complete documentation.

24.441 nettypes.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00025 #ifndef NB_NET_TYPES_H
00026 #define NB_NET_TYPES_H
00027
```

```
00028 #include <predef.h>
00029 #include <basictypes.h>
00030
00031 typedef int(PutCharsFunction)(void *data, const char *chars, int len);
00032
00033 /*
00034  ******************************************************************************
00035  *
00036  * Definitions
00037  *
00038  ******************************************************************************
00039  */
00040
00041 #define PPP_TYPE (0x0021)  /* These are Big Endian constants! */
00042 #define EIP_TYPE (0x0800)  /* These are Big Endian constants! */
00043 #define EIP6_TYPE (0x86DD) /* These are Big Endian constants! */
00044 #define HARD_ENET (0x0001) /* These are Big Endian constants! */
00045 #define VLAN_TYPE (0x8100)
00046 #define EARP_TYPE (0x0806)
00047 #define ARP_REQUEST (0x01)
00048 #define ARP_RESPONSE (0x02)
00049
00050 /*
00051  * Media Access Control (MAC) address size in 8 bit bytes and 16 bit words
00052  */
00053 #define MACADDRESS_OCTETS_48 (6)
00054 #define MACADDRESS_WORDS_48 (3)
00055 typedef struct _MACADDRESS_48
00056 {
00057     uint8_t octet[MACADDRESS_OCTETS_48];
00058 } __attribute__((packed)) MACADDRESS_48;
00059
00060 class MACADR;   // Forward
00061
00066 typedef class MACADR
00067 {
00068    public:
00069     beuint16_t phywadr[MACADDRESS_WORDS_48] = {0};
00074     inline bool IsNull() { return ((phywadr[2] == 0) && (phywadr[1] == 0) && (phywadr[0] == 0)); };
00079     inline bool IsMultiCast() { return (phywadr[0] & 0x0100); };
00084     inline bool IsBroadCast() { return ((phywadr[0] == 0xFFFF) && (phywadr[1] == 0xFFFF) &&
      (phywadr[2] == 0xFFFF)); };
00089     uint8_t GetByte(int n) const
00090     {
00091         //#if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00092         //       if (n&1) { return ((phywadr[n/2]»8)&0xFF); }
00093         //       else     { return (phywadr[n/2]&0xFF); }
00094         //#else
00095         if (n & 1) { return (phywadr[n / 2] & 0xFF); }
00096         else
00097         {
00098             return ((phywadr[n / 2] » 8) & 0xFF);
00099         }
00100         //#endif
00101     };
00106     inline void SetFromBytes(const uint8_t *pb)
00107     {
00108         uint8_t *pto = (uint8_t *)phywadr;
00109         for (int i = 0; i < 6; i++)
00110         {
00111             pto[i] = pb[i];
00112         }
00113     };
00118     void fdprint(int fd);
00123     inline void print() { fdprint(1); };
00128     MACADR operator+(uint32_t rhs)
00129     {
00130         MACADR ret = *this;
00131         uint32_t tmp;
00132         tmp = ret.phywadr[2] + rhs;
00133         do {
00134             ret.phywadr[2] = tmp & 0xffff;
00135             tmp »= 16;
00136             tmp = phywadr[1] + tmp;
00137             ret.phywadr[1] = tmp & 0xffff;
00138             tmp »= 16;
00139             tmp = phywadr[0] + tmp;
00140             ret.phywadr[0] = tmp & 0xffff;
00141             tmp »= 16;
00142         } while (tmp && ((tmp = ret.phywadr[2] + rhs)));
00143         return ret;
00144     }
00145 } __attribute__((packed)) MACADR;
00146
00151 inline bool operator==(const MACADR &i, const MACADR &j)
00152 {
00153     if (i.phywadr[0] != j.phywadr[0]) return FALSE;
```

```
00154     if (i.phywadr[1] != j.phywadr[1]) return FALSE;
00155     if (i.phywadr[2] != j.phywadr[2]) return FALSE;
00156     return TRUE;
00157 }
00158
00163 inline bool operator!=(const MACADR &i, const MACADR &j)
00164 {
00165     if (i.phywadr[0] != j.phywadr[0]) return TRUE;
00166     if (i.phywadr[1] != j.phywadr[1]) return TRUE;
00167     if (i.phywadr[2] != j.phywadr[2]) return TRUE;
00168     return FALSE;
00169 }
00170
00175 inline bool operator>(const MACADR &i, const MACADR &j)
00176 {
00177     if (i.phywadr[0] > j.phywadr[0])
00178         return true;
00179     else if (i.phywadr[0] < j.phywadr[0])
00180         return false;
00181
00182     if (i.phywadr[1] > j.phywadr[1])
00183         return true;
00184     else if (i.phywadr[1] < j.phywadr[1])
00185         return false;
00186
00187     if (i.phywadr[2] > j.phywadr[2]) return true;
00188
00189     return false;
00190 }
00191
00192 extern MACADR ENET_BCAST;
00193 extern MACADR ENET_ZERO;
00194
00195 // Forward declaration
00196 class CUR_IPADDR4;
00197
00205 class IPADDR4
00206 {
00207    private:
00208     beuint32_t ip_val;
00209
00210    public:
00215     IPADDR4() = default;
00216
00223     IPADDR4(const IPADDR4 &v) = default;
00224
00225     IPADDR4 &operator=(const IPADDR4 &v)
00226     {
00227         ip_val = v.ip_val;
00228         return *this;
00229     };
00230
00231     volatile IPADDR4 &operator=(const IPADDR4 &v) volatile
00232     {
00233         ip_val = v.ip_val;
00234         return *this;
00235     };
00236
00237     IPADDR4 &operator=(const uint32_t v)
00238     {
00239         ip_val = v;
00240         return *this;
00241     };
00242
00243     volatile IPADDR4 &operator=(const uint32_t v) volatile
00244     {
00245         ip_val = v;
00246         return *this;
00247     };
00248
00249     bool IsEmbeddedIPV4() const { return true; };
00250     IPADDR4 Extract4() const { return *this; };
00251     operator uint32_t() const { return (uint32_t)ip_val; };
00252
00260     inline bool IsNull() const { return ip_val == 0; };
00261
00269     inline bool NotNull() const { return !IsNull(); };
00270
00276     inline void SetNull() { ip_val = 0; };
00277
00285     inline bool IsLoopBack() const { return ((ip_val & 0xFF000000) == 0x7F000000); };
00286
00294     inline bool IsMultiCast() const { return ((ip_val & 0xF0000000) == 0xE0000000); };   // 224. to
      239. E0....EF.
00295
00303     inline bool IsmDns() { return (ip_val==0xE00000FB); };
00304
```

```
00305
00313     inline bool IsGlobalBroadCast() const { return ip_val == 0xffffffff; };
00314
00322     inline bool IsAutoIP() { return ((ip_val & 0xFFFF0000) == 0xA9FE0000); };
00323
00324     //    IPADDR4() {ip_val=0; };
00325     IPADDR4(uint32_t v) { ip_val = v; };
00326     IPADDR4(uint8_t a, uint8_t b, uint8_t c, uint8_t d) { ip_val = (a << 24) | (b << 16) | (c << 8) | d;
    }
00327
00328     // Return the MAC address for multicasts on at this address, NULL if not multicast
00329     inline MACADR McastMac() const
00330     {
00331         uint32_t ipDst = ip_val;
00332         ipDst &= 0x007FFFFF;
00333         MACADR ma;
00334         ma.phywadr[0] = 0x0100;
00335         ma.phywadr[1] = 0x5E00 | (uint16_t)((ipDst >> 16) & 0x7F);
00336         ma.phywadr[2] = (uint16_t)(ipDst & (0xFFFF));
00337         return ma;
00338     };
00339
00344     inline static IPADDR4 NullIP()
00345     {
00346         IPADDR4 i4;
00347         i4.ip_val = 0;
00348         return i4;
00349     };
00350
00355     inline static IPADDR4 GlobalBroadCast()
00356     {
00357         IPADDR4 i4;
00358         i4.ip_val = 0xFFFFFFFF;
00359         return i4;
00360     };
00361
00366     void print() const;
00367
00374     void fdprint(int fd) const;
00375
00385     int sprintf(char *cp, int maxl) const;
00386
00392     void SetFromAscii(const char *cp);
00393
00394     int GetPrintLen(bool compact);
00395     int PrintHelper(PutCharsFunction *pf, void *data, bool compact);
00396
00397     bool IsBcastNetMask(IPADDR4 intfIP, IPADDR4 mask) const
00398     {
00399         return ((ip_val & mask.ip_val) == (intfIP.ip_val & mask.ip_val)) && ((ip_val & ~mask.ip_val)
    == (~mask.ip_val));
00400     }
00401
00402     // Helpers for === and != tests....
00403     friend bool operator==(const IPADDR4 i, const IPADDR4 j);
00404     friend bool operator!=(const IPADDR4 i, const IPADDR4 j);
00405     friend bool operator>(const IPADDR4 i, const IPADDR4 j);
00406     friend bool operator<(const IPADDR4 i, const IPADDR4 j);
00407
00408     friend bool operator==(const uint32_t i, const IPADDR4 j);
00409     friend bool operator!=(const uint32_t i, const IPADDR4 j);
00410     friend bool operator>(const uint32_t i, const IPADDR4 j);
00411     friend bool operator<(const uint32_t i, const IPADDR4 j);
00412
00413     friend bool operator==(const IPADDR4 i, const uint32_t j);
00414     friend bool operator!=(const IPADDR4 i, const uint32_t j);
00415     friend bool operator>(const IPADDR4 i, const uint32_t j);
00416     friend bool operator<(const IPADDR4 i, const uint32_t j);
00417
00418     friend IPADDR4 IPV4FromConst(uint32_t d);
00419     friend IPADDR4 IPV4FromConst(beuint32_t d);
00420
00421     friend IPADDR4 LocalBroadCast(IPADDR4 ifip, IPADDR4 ipmask);
00422
00423     friend class CUR_IPADDR4;
00424
00425     // friend BE32<uint32_t>::BE32(IPADDR4 rhs);
00426     // friend BE32<int32_t>::BE32(IPADDR4 rhs);
00427
00428 } __attribute__((packed));
00429
00430 // Helpers for === and != tests....
00431 inline bool operator==(const IPADDR4 i, const IPADDR4 j)
00432 {
00433     return i.ip_val == j.ip_val;
00434 }
00435 inline bool operator!=(const IPADDR4 i, const IPADDR4 j)
```

```
00436 {
00437     return i.ip_val != j.ip_val;
00438 }
00439 inline bool operator>(const IPADDR4 i, const IPADDR4 j)
00440 {
00441     return i.ip_val > j.ip_val;
00442 }
00443 inline bool operator<(const IPADDR4 i, const IPADDR4 j)
00444 {
00445     return i.ip_val < j.ip_val;
00446 }
00447
00448 inline bool operator==(const uint32_t i, const IPADDR4 j)
00449 {
00450     return i == j.ip_val;
00451 }
00452 inline bool operator!=(const uint32_t i, const IPADDR4 j)
00453 {
00454     return i != j.ip_val;
00455 }
00456 inline bool operator>(const uint32_t i, const IPADDR4 j)
00457 {
00458     return i > j.ip_val;
00459 }
00460 inline bool operator<(const uint32_t i, const IPADDR4 j)
00461 {
00462     return i < j.ip_val;
00463 }
00464
00465 inline bool operator==(const IPADDR4 i, const uint32_t j)
00466 {
00467     return i.ip_val == j;
00468 }
00469 inline bool operator!=(const IPADDR4 i, const uint32_t j)
00470 {
00471     return i.ip_val != j;
00472 }
00473 inline bool operator>(const IPADDR4 i, const uint32_t j)
00474 {
00475     return i.ip_val > j;
00476 }
00477 inline bool operator<(const IPADDR4 i, const uint32_t j)
00478 {
00479     return i.ip_val < j;
00480 }
00481
00482 inline IPADDR4 IPV4FromConst(uint32_t d)
00483 {
00484     IPADDR4 i4;
00485     i4.ip_val = d;
00486     return i4;
00487 };
00488
00489 inline IPADDR4 LocalBroadCast(IPADDR4 ifip, IPADDR4 ipmask)
00490 {
00491     return IPV4FromConst(ifip.ip_val | ~(ipmask.ip_val));
00492 };
00493
00494 /*
00495
00496
00497  ********************************************************************************
00498  *
00499  * Data Structures
00500  *
00501  ********************************************************************************
00502  */
00503
00504 /*
00505  * MAC address
00506  *    octet - address bytes
00507  */
00508
00509 /*
00510  * Definition for an Ethernet frame
00511  */
00512 typedef struct
00513 {
00514     MACADR dest_addr;
00515     MACADR src_addr;
00516     beuint16_t eType;
00517     uint8_t pData[];
00518 } EFRAME;
00519
00520 typedef EFRAME *PEFRAME;
00521
00522 typedef struct
```

```
00523 {
00524     MACADR dest_addr;
00525     MACADR src_addr;
00526     beuint16_t eVlType;
00527     beuint16_t eTag;
00528     beuint16_t eType;
00529     uint8_t pData[];
00530 } VLEFRAME;
00531
00532 typedef VLEFRAME *PVLEFRAME;
00533
00534 #ifdef IPV6
00535 #include <ipv6/ipv6_addr.h>
00536
00541 typedef IPADDR6 IPADDR;
00542
00543 #define GetNullIP() IPADDR::NullIP()
00544
00545 #else
00546
00547 #ifndef IPV4ONLY
00548 #error Got to pick an IP version
00549 #endif
00550 typedef IPADDR4 IPADDR;
00551 #define GetNullIP() IPADDR4::NullIP()
00552 #endif
00553
00554
00555 /* Any address */
00556 #define INADDR_ANY4 IPADDR4::NullIP()
00557
00558 #ifdef IPV6
00559 #define INADDR_ANY IPADDR::NullIP()
00560 #else
00561 #define INADDR_ANY INADDR_ANY4
00562 #endif
00563
00564 #endif   // #ifndef NB_NET_TYPES_H
00565
```

24.442 pop3.h File Reference

NetBurner POP3 API.
`#include <nettypes.h>`

Macros

- #define **POP_OK** (0)

  No errors occurred.
- #define **POP_TIMEOUT** (-1)

  Time out.
- #define **POP_PASSWORDERROR** (-2)

  Password error.
- #define **POP_CONNECTFAIL** (-3)

  Connection failed.
- #define **POP_COMMANDFAIL** (-4)

  Command failed.
- #define **POP_BADSESSION** (-5)

  Session error.
- #define **POP_NETWORKERROR** (-6)

  Network error.
- #define **POP_BUFFER_FULL** (-7)

  Receive buffer full.

Functions

- int POPGetResultCode (int fd, uint32_t timeout)

  Returns the result code of the previous POP3 operation.
- int POP3_InitializeSession (IPADDR server_address, uint16_t port, PCSTR UserName, PCSTR Password, uint32_t timeout)

  Create a connection to the POP3 server and log in.
- int POP3_CloseSession (int session)

  Close a POP3 session.
- int POP3_StatCmd (int session, uint32_t *num_messages, uint32_t *total_bytes, uint32_t timeout)

  Returns the status of the mailstore on the POP3 server.
- int POP3_ListCmd (int session, uint32_t message_number, uint32_t *total_bytes, uint32_t timeout)

  Get the size of a message on the server.
- int POP3_DeleteCmd (int session, uint32_t message_number, uint32_t timeout)

  Delete a pending message on the server.
- int POP3_RetrieveMessage (int session, uint32_t message_number, char *buffer, char **subject_ptr, char **body_ptr, int max_bufferlen, uint32_t timeout)

  Retrieve a message from the server.
- PCSTR GetPOPErrorString (int err)

  Returns the error text for an error code.

24.442.1 Detailed Description

NetBurner POP3 API.

24.443 pop3.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00014 #ifndef _POP3_H_
00015 #define _POP3_H_
00016
00017 #include <nettypes.h>
00018
00019 // #define POP3_DEBUG                  ( 1 )    // Library debug switch
00020
00021 #ifdef POP3_DEBUG
00022 #define POP3_DEBUG_IPRINTF(...)                  \
00023     {                                            \
00024         iprintf("%s:%d", __FUNCTION__, __LINE__); \
00025         iprintf(__VA_ARGS__);                    \
00026         iprintf("\r\n");                         \
00027     }
00028 #else /* #ifdef POP3_DEBUG */
00029 #define POP3_DEBUG_IPRINTF(...) ((void)0)
00030 #endif /* #ifdef POP3_DEBUG */
00031
00035 #define POP_OK (0)
00036 #define POP_TIMEOUT (-1)
00037 #define POP_PASSWORDERROR (-2)
00038 #define POP_CONNECTFAIL (-3)
00039 #define POP_COMMANDFAIL (-4)
00040 #define POP_BADSESSION (-5)
00041 #define POP_NETWORKERROR (-6)
00042 #define POP_BUFFER_FULL (-7)
00054 int POPGetResultCode(int fd, uint32_t timeout);
00055
00069 int POP3_InitializeSession(IPADDR server_address, uint16_t port, PCSTR UserName, PCSTR Password,
      uint32_t timeout);
00070
00079 int POP3_CloseSession(int session);
00080
00092 int POP3_StatCmd(int session, uint32_t *num_messages, uint32_t *total_bytes, uint32_t timeout);
00093
00105 int POP3_ListCmd(int session, uint32_t message_number, uint32_t *total_bytes, uint32_t timeout);
00106
```

```
00119 int POP3_DeleteCmd(int session, uint32_t message_number, uint32_t timeout);
00120
00139 int POP3_RetrieveMessage(int session,
00140                          uint32_t message_number,
00141                          char *buffer,
00142                          char **subject_ptr,
00143                          char **body_ptr,
00144                          int max_bufferlen,
00145                          uint32_t timeout);
00146
00155 PCSTR GetPOPErrorString(int err);
00156
00157 #endif /* #ifndef _POP3_H_ */
00158
```

24.444 ppp.h File Reference

PPP - Point to Point Protocol.

```
#include <nettypes.h>
#include <buffers.h>
#include <netinterface.h>
#include <vjhc.h>
```

Macros

- #define **ERR_PPP_SUCCESS** (0)

  Success.
- #define **ERR_PPP_ALREADY_OPEN** (-1)

  A session is already open.
- #define **ERR_PPP_NO_DIALTONE** (-2)

  No dial tone.
- #define **ERR_PPP_NO_ANSWER** (-3)

  The remote client did not answer.
- #define **ERR_PPP_BUSY** (-4)

  The remote client is sending a busy signal.
- #define **ERR_PPP_FAIL** (-5)

  The attempted action has failed.
- #define **ERR_PPP_PASSFAIL** (-6)

  Pass/Fail.
- #define **ERR_PPP_LOSTCARRIER** (-7)

  Lost connection carrier signal.
- #define **ERR_PPP_NO_MODEM** (-8)

  No modem detected.
- #define **ERR_PPP_LCP_FAILED** (-9)

  LCP negotiation has failed.
- #define **ERR_PPP_CHAPFAIL** (-10)

  CHAP negotiation has failed.

Enumerations

- enum enum_PPPState {
  eClosed , eInitializingModem , eDialing , eWait4Ring ,
  eAnswering , eWaitForTrain , eLCPNegotiate , ePAPAuthenticate ,
  eCHAPAuthenticate , eNCPNegotiate , eOpen , eClosing }

  PPP States.

24.444.1 Detailed Description

PPP - Point to Point Protocol.

24.445 ppp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00029 #ifndef _NB_PPP_H
00030 #define _NB_PPP_H
00031
00032 #include <nettypes.h>
00033 #include <buffers.h>
00034 #include <netinterface.h>
00035 #include <vjhc.h>
00036
00037 class PPPInterface;   // forward declaration
00038
00039 class ahdlc
00040 {
00041    private:
00042     /* User controlled variables. */
00043     uint8_t tx_accm[32], rx_accm[32];
00044
00045     /* Internal state data. */
00046     unsigned char *rx_bufp;
00047     PoolPtr rx_pool;
00048     beuint16_t rx_crc;
00049     char escaped;
00050     OS_SEM TransmitSemaphore;
00051
00052     void initialize_rxbuffer();
00053     PPPInterface *m_pIf;
00054
00055    public:
00056     ahdlc(PPPInterface *pIf);
00057     ~ahdlc();
00058     void clear();
00059     void receive(char *buffer, int count);
00060     void sendone(int fd, char c);
00061     void transmit(int fd, char *buffer, int n);
00062     void set_tx_accm(uint32_t tx_accm);
00063     void set_rx_accm(uint32_t rx_accm);
00064 };
00065
00075 typedef enum
00076 {
00077     eClosed,
00078     eInitializingModem,
00079     eDialing,
00080     eWait4Ring,
00081     eAnswering,
00082     eWaitForTrain,
00083     eLCPNegotiate,
00084     ePAPAuthenticate,
00085     eCHAPAuthenticate,
00086     eNCPNegotiate,
00087     eOpen,
00088     eClosing
00089 } enum_PPPState;
00096 #define ERR_PPP_SUCCESS (0)
00097 #define ERR_PPP_ALREADY_OPEN (-1)
00098 #define ERR_PPP_NO_DIALTONE (-2)
00099 #define ERR_PPP_NO_ANSWER (-3)
00100 #define ERR_PPP_BUSY (-4)
00101 #define ERR_PPP_FAIL (-5)
00102 #define ERR_PPP_PASSFAIL (-6)
00103 #define ERR_PPP_LOSTCARRIER (-7)
00104 #define ERR_PPP_NO_MODEM (-8)
00105 #define ERR_PPP_LCP_FAILED (-9)
00106 #define ERR_PPP_CHAPFAIL (-10)
00109 /* Definition for a PPP Frame */
00110 typedef struct
00111 {
00112     uint8_t unused[10];
00113     beuint16_t PPPAddr;
00114     beuint16_t PPPFrameType;
00115     uint8_t code;
00116     uint8_t idval;
00117     beuint16_t length;
00118     beuint16_t pData[4];
```

```
00119 } __attribute__((packed)) PPPFRAME;
00120
00121 class LCPState_Class
00122 {
00123    protected:
00124     int m_nState;
00125     uint16_t m_nOptions;
00126     uint32_t options_ok_mask;
00127     uint32_t options_nak_mask;
00128     uint32_t options_rej_mask;
00129     uint16_t m_TicksLeft;
00130     uint16_t m_toCount;
00131     uint16_t m_FrameType;
00132     uint16_t m_nRequests_Sent;
00133     LCPState_Class *pNextLayerUp;
00134     LCPState_Class *pNextLayerDn;
00135     uint8_t m_Last_ReqSent;
00136     uint8_t m_Last_ReqRespond;
00137     PPPInterface *m_ifc;
00138
00139    public:
00140     void RecieveMsg(PPPFRAME *pPPP, PoolPtr pp);
00141     virtual void TestOption(puint8_t option_start, uint32_t maskv) = 0;
00142
00143     void Up();
00144     void Down();
00145     void Open();
00146     void Close();
00147     void Tick();
00148     void Initialize(PPPInterface *ppi, uint16_t frame_type, LCPState_Class *next_layer, LCPState_Class
      *pre_layer);
00149     int GetState();
00150     void SendPPPBuffer(PoolPtr ppSend);
00151     void SendProtoReject(PPPFRAME *pPPP, PoolPtr pp);
00152
00153    protected:
00154     void SetState(int i);
00155     void do_irc();
00156     void do_zrc();
00157     virtual void do_scr() = 0;
00158     virtual void do_RCN(PPPFRAME *pPPP, int reject) = 0;
00159     virtual void do_wereup();
00160     virtual void do_weredown();
00161     void do_sca(PPPFRAME *pPPP, PoolPtr pp);
00162     void do_scn(PPPFRAME *pPPP, PoolPtr pp);
00163     void CopyOptions(PPPFRAME *pPPPOut, PPPFRAME *pPPPin, uint32_t option_mask);
00164     void do_str();
00165     void do_sta(PPPFRAME *pPPP, PoolPtr pp);
00166     void do_ser(PPPFRAME *pPPP, PoolPtr pp);
00167     void do_abortconn();
00168 };
00169
00170 class LCP_Class : public LCPState_Class
00171 {
00172    public:
00173     virtual void TestOption(puint8_t option_start, uint32_t maskv);
00174     virtual void do_scr();
00175     virtual void do_RCN(PPPFRAME *pPPP, int reject);
00176     virtual void do_wereup();
00177     virtual void do_weredown();
00178 };
00179
00180 class IPCP_Class : public LCPState_Class
00181 {
00182    public:
00183     virtual void TestOption(puint8_t option_start, uint32_t maskv);
00184     virtual void do_scr();
00185     virtual void do_RCN(PPPFRAME *pPPP, int reject);
00186     virtual void do_wereup();
00187     virtual void do_weredown();
00188 };
00189
00190 class PPPInterface : public InterfaceBlock
00191 {
00192    protected:
00193     PPPInterface(const char *name = "PPP", const char *pDesc = "PPP Interface");
00194     PPPInterface *pNext;
00195
00196    public:
00197     config_int Restart_Interval{
00198         3, "Restart_Interval",
00199         "Seconds to wait to restart failed negotiation"};
00200     config_int Max_Terminate{2, "Max_Terminate",
00201                              "Max times to send terminate request before giving up"};
00202     config_int Max_Configure{
00203         25, "Max_Configure",
00204         "Max number of config request loops before giving up"};
```

```
00205     config_int Max_Failure{25, "Max_Failure",
00206                            "The max number of config failure before giving up"};
00207     config_bool DNSRequest{true, "DNSRequest", "Request DNS from the other side of the connection"};
00208     config_bool CHAPenable{true, "CHAPEnable", "Enable CHAP authentication"};
00209     config_bool RestartOnClose{false, "RestartOnclose", "Restart PPP automatically from close"};
00210     config_pass UserName{"nburn", "User", "PPP Username for this interface"};
00211     config_pass PassWord{"nburn", "Password", "PPP Password for this interface"};
00212     uint32_t TX_DESIRED_ACCM;
00213     uint32_t RX_DESIRED_ACCM;
00214     config_IPADDR4 OfferedDNSAddress{"0.0.0.0", "OfferedDNS", "Offer a DNS address to the other
      side"};
00215     ConfigEndMarker;
00216
00217     uint32_t dwflags;
00218     uint16_t MRU;
00219     uint32_t TX_ACCM;
00220     uint32_t RX_ACCM;
00221
00222     volatile bool if_up;
00223     volatile int abort_ppp;
00224     volatile enum_PPPState cur_State;
00225     volatile bool abortornot;
00226     volatile bool m_bDataSeen;
00227     volatile uint32_t NextTick;
00228
00229     int fdserial;
00230
00231     OS_SEM state_semaphore;
00232
00233     int ppp_nError;
00234
00235     volatile uint16_t PPPIds;
00236
00237     const char *hang_string;
00238     int hang_state;
00239
00240     bool bTO;   // Did we time out
00241     int pap_to;
00242     int chap_to;
00243     BOOL client;   // Are we a client?
00244
00245     uint8_t ChallengeID;
00246     uint32_t CurrentRand[4];
00247
00248     bool DoVJCompress;
00249     slcompress VJHCStruct;
00250     ahdlc ahdlc_obj{this};
00251     LCP_Class lcp_automaton;
00252     IPCP_Class ipcp_automaton;
00253     bool pap_complete;
00254     bool chap_complete;
00255
00256     virtual void send_func(PoolPtr poolPtr);
00257     virtual bool LinkActive();
00258     virtual int LinkSpeed();
00259     virtual bool LinkDuplex();
00260     virtual bool bNeedsArp();
00261
00262     volatile enum_PPPState GetPPPState();
00263
00264     void EnableMulticast(MACADR macAddress, BOOL addAddress);
00265     void processppp();
00266     void ClosePPPSesion();
00267     void SendPAP();
00268     void PAPTick();
00269     void ProcessPAP(PPPFRAME *pPPP, PoolPtr pp);
00270     void ProcessPPPIP(PPPFRAME *pPPP, PoolPtr pp);
00271     void SendCHAP(uint8_t code, uint8_t id = 0);   // Sends a CHAP packet to the PPP connected device
00272     void SendCHAPChallenge();
00273     void CHAPTick();
00274     void ProcessCHAP(PPPFRAME *pPPP, PoolPtr pp);
00275     void process_rx(PoolPtr pp);
00276     void UpdateTimeout();
00277
00278     int StartIf(int fd);
00279     void Restart();
00280
00281     static void PPPDaemonTask(void *p);
00282     static PPPInterface *pListHead;
00283 };
00284
00285 class PPPServer : public PPPInterface
00286 {
00287    public:
00288     PPPServer(const char *name = "PPPS", const char *pDesc = "PPP Server Interface");
00289 };
00290
```

```
00291 class PPPClient : public PPPInterface
00292 {
00293    public:
00294     PPPClient(const char *name = "PPPC", const char *pDesc = "PPP Server Interface");
00295 };
00296
00297 #endif
00298
```

24.446 examples/JSON/DemoNetBurner/overload/nbrtos/include/predef-overload.h

```
00001 #define INCLUDE_HTTP_DIAG  (1)
00002 #define INCLUDE_TLS_DIAG   (1)
```

24.447 examples/MultiHome/overload/nbrtos/include/predef-overload.h

```
00001 #define MULTIHOME (1)
00002 #define NUM_MULTI_INTERFACES (10)
```

24.448 examples/OverloadDirectory/overload/nbrtos/include/predef-overload.h

```
00001 #define NBRTOS_TIME           (1)
```

24.449 examples/PlatformSpecific/SAME70/MODM7AE70/MODM7AE70↩ FactoryApp/overload/nbrtos/include/predef-overload.h

```
00001 #define SUPPORT_LEGACY_IPSETUP (1)
```

24.450 examples/Profile/overload/nbrtos/include/predef-overload.h

```
00001 #define NBRTOS_TIME           (1)
```

24.451 examples/SSH/SecureSerToEthFactory↩ App/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED       (1) // Enable SSH on this example
00002
00003 // Enable legacy IPSetup and NBFind for MCF5441X platforms to support EnableAutoUpdate features.
00004 #if (defined MCF5441X)
00005     #define SUPPORT_LEGACY_IPSETUP (1)
00006     #define SUPPORT_LEGACY_FIND    (1)
00007 #endif
```

24.452 examples/SSH/sshMinimalClient/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED   (1) // Enable SSH on this example
```

24.453 examples/SSH/sshMinimalServer/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED   (1) // Enable SSH on this example
```

24.454 examples/SSH/sshServerUser↩ Auth/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED   (1) // Enable SSH on this example
```

24.455 examples/SSH/SshServerUser↩ Key/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED   (1) // Enable SSH on this example
```

24.456 examples/SSL/FTPSServer/overload/nbrtos/include/predef-overload.h

```
00001 #define FTPD_SSL_SUPPORT (1)
```

24.457 examples/SSL/HttpsUploadCert/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED   (1) // Enable SSH on this example
```

24.458 examples/StackProtection/overload/nbrtos/include/predef-overload.h

```
00001 #define NBRTOS_STACKCHECK    (1)
00002
00003 // Coldfire chips can only monitor a single memory range, must pick to monitor overflow or underflow
00004 #ifdef COLDFIRE
00005 #define NBRTOS_STACKOVERFLOW (1)
00006 //  #define NBRTOS_STACKUNDERFLOW (1)
00007 #endif
00008
00009 // Cortex M7 can monitor 2 memory ranges, so can watch for overflow AND underflows simultaneously.
00010 #ifdef CORTEX_M7
00011 #define NBRTOS_STACKOVERFLOW (1)
00012 #define NBRTOS_STACKUNDERFLOW (1)
00013 #endif
```

24.459 examples/telnetcmd/overload/nbrtos/include/predef-overload.h

```
00001 #define NB_SSH_SUPPORTED   (1) // Enable SSH on this example
```

24.460 examples/VLan/overload/nbrtos/include/predef-overload.h

```
00001 #define MULTIHOME
00002 #define NUM_MULTI_INTERFACES (10)
```

24.461 nbrtos/include/predef-overload.h

```
00001 /* This file intentionally left blank
00002  *
00003  * It is used to modify values in predef.h from an overload project directory
00004  *
00005  * */
```

24.462 predef.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
```

```
00004
00005 #ifndef _PREDEF_H_
00006 #define _PREDEF_H_
00007
00008 /* This release  Build on: $Date: 2012/03/28 14:35:22 $ */
00009 /* This build revision tag: $Name:  $ */
00010
00011 /*
00012  ******************************************************************************
00013  *
00014  * Features
00015  *       NBRTOS_PRIO_PROMOTION  - Priority Inheritance upon Priority Inversion
00016  *                                of OS_CRIT ownership
00017  *       NBRTOS_SCANF_FLOAT     - Enable float support on scanf. Can be disabled
00018  *                                to save ~1k flash image size
00019  *
00020  ******************************************************************************
00021  */
00022 /* #define NBRTOS_PRIO_PROMOTION  (1) */
00023 #define NBRTOS_SCANF_FLOAT (1)
00024
00025 /*
00026  ******************************************************************************
00027  *
00028  * Debugging
00029  *
00030  *    Use these constants to turn debug features on and off
00031  *       NBRTOS_STACKCHECK      - Stack integrity
00032  *       NBRTOS_STACKOVERFLOW   - Real time stack overflow protection
00033  *                                * Reduces all stack sizes by 256 bytes, creating
00034  *                                    a canary zone which will be watched for writes
00035  *                                * Use compiler option -fstack-check
00036  *                                * Enable in application with EnableOSStackProtector()
00037  *                                * Coldfire modules can enable either overflow or underflow checking
00038  *                                * ARM modules can enable both overflow and underflow checking
     simultaneously
00039  *       NBRTOS_STACKUNDERFLOW  - Real time stack underflow protection
00040  *                                * Reduces all stack sizes by 256 bytes, creating
00041  *                                    a canary zone which will be watched for writes
00042  *                                * Use compiler option -fstack-check
00043  *                                * Enable in application with EnableOSStackProtector()
00044  *                                * Coldfire modules can enable either overflow or underflow checking
00045  *                                * ARM modules can enable both overflow and underflow checking
     simultaneously
00046  *       NBRTOS_TASKLIST        - Task diagnostics
00047  *       NBRTOS_TASK_LOG        - Task change callback with new prio, IRQ context.
00048  *       NBRTOS_TIME            - Task time counting
00049  *       BUFFER_DIAG            - Buffer integrity and use
00050  *       BUFFER_DIAG_LOG        - Buffer integrity and use logging, via SysLog
00051  *       BUFFER_SANITY          - Buffer Pool sanity traps
00052  *       USER_CRITICAL_SANITY   - Enables an assert if an illegal pend occurs within a
     USER_ENTER_CRITICAL() block
00053  *       _DEBUG_PRINT           - Enables DBPRINT Macros in release builds
00054  *       ENABLE_SMARTTRAP       - Enables extra debugging information during traps
00055  ******************************************************************************
00056  */
00057 /* #define NBRTOS_STACKCHECK     (1) */
00058 /* #define NBRTOS_STACKOVERFLOW  (1) */
00059 /* #define NBRTOS_STACKUNDERFLOW (1) */
00060 /* #define NBRTOS_TASKLIST       (1) */
00061 /* #define NBRTOS_TASK_LOG       (1) */
00062 /* #define NBRTOS_TIME           (1) */
00063 /* #define BUFFER_DIAG           (1) */
00064 /* #define BUFFER_DIAG_LOG       (1) */
00065 /* #define BUFFER_SANITY         (1) */
00066 /* #define USER_CRITICAL_SANITY  (1) */
00067 /* #define _DEBUG_PRINT          (1) */
00068 #define ENABLE_SMARTTRAP (1)
00069
00070 /*
00071  ******************************************************************************
00072  *
00073  * Utility
00074  *
00075  *  Development features potentially undesirable in final release
00076  *      ALLOW_NBID_REBOOT       - Adds single UDP message reboot to device
00077  ******************************************************************************
00078  */
00079 /* #define ALLOW_NBID_REBOOT (1) */
00080
00081 /*
00082  ******************************************************************************
00083  *
00084  * Multihome
00085  *
00086  *    Uncomment to enable multihoming operation
00087  *
```

```
00088  ***************************************************************************
00089  */
00090 //#define MULTIHOME
00091 #ifdef MULTIHOME
00092 #define NUM_MULTI_INTERFACES (10)
00093 #else
00094 #define NUM_MULTI_INTERFACES (0)
00095 #endif
00096
00097 /*
00098  ***************************************************************************
00099  *
00100  * Interface limits
00101  *
00102  *    MAX_NET_INTERFACES     - Maximum number of registered network interfaces
00103  *
00104  ***************************************************************************
00105  */
00106 #define MAX_NET_INTERFACES (4 + NUM_MULTI_INTERFACES)
00107
00108 /***************************************************************************
00109  *
00110  * IPv6 Config
00111  *
00112  * Select dual stack mode or IPv4 only mode
00113  *
00114  ***************************************************************************
00115  */
00116 #define IPV6 (1)   // Dual stack IPv4/IPv6 mode
00117
00118 #define IPV6_COUNTERS (1)   // add counters to IPV6
00119 //#define IPV4ONLY (1)  // IPv4 only mode
00120
00121 /*
00122  ***************************************************************************
00123  *
00124  * Auto-IP
00125  *
00126  *  Comment out this line to disable the Auto-IP virtual interface
00127  *
00128  ***************************************************************************
00129  */
00130
00131 #define AUTOIP
00132
00133 /*
00134  ***************************************************************************
00135  *
00136  * Custom Ethernet Handlers
00137  *
00138  *  Uncomment this line to enable Custom Ethernet Handlers
00139  *
00140  ***************************************************************************
00141  */
00142 /* #define ALLOW_CUSTOM_NET_DO_RX */
00143
00144 /*
00145  ***************************************************************************
00146  *
00147  * TCP No Copy Mode
00148  *
00149  *  Uncomment these lines to enable TCP No Copy features
00150  *  These reduce buffer space efficiency in order to eliminate the secondary
00151  *  copy when performing TCP transactions.
00152  *
00153  ***************************************************************************
00154  */
00155 #define TCP_NOCOPY_TX (1)
00156
00157 /*
00158  ***************************************************************************
00159  *
00160  * TCP Socket Throughput Information
00161  *
00162  *  Uncomment this line to enable TCP Throughput Information
00163  *  This will add statistics counters that will track the total and payload
00164  *  bytes sent and received by individual TCP sockets.
00165  *
00166  ***************************************************************************
00167  */
00168 /* #define TCP_THROUGHPUT_INFO_ENABLED (1) */
00169
00170 /*
00171  ***************************************************************************
00172  *
00173  * Multi-home
00174  *
```

```
00175  *    Uncomment to enable UDP fragments
00176  *
00177  ******************************************************************************
00178  */
00179 /* #define UDP_FRAGMENTS  ( 4 ) */
00180
00181 /*
00182  ******************************************************************************
00183  *
00184  * Random value
00185  *
00186  *    Comment out this line to eliminate the random value support in the
00187  *       library. It was commented out up until and including Rel2.4 Rc3
00188  *
00189  ******************************************************************************
00190  */
00191 #define GATHER_RANDOM (1)
00192
00193 /*
00194  ******************************************************************************
00195  *
00196  * SSL and/or SSH support
00197  *
00198  *    Needs to be uncommented to support these features
00199  *
00200  ******************************************************************************
00201  */
00202 /*
00203  * SSL Supported
00204  *    Should be defined when SSL is included in library
00205  *
00206  */
00207 #define NB_SSL_SUPPORTED (1)
00208 /* #define SSL_V3_DISABLED    ( 1 ) */
00209
00210 /******************************************************************************
00211  * Optional SSL Features
00212  *  SSL_TLS_SUPPORT - TLS is supported by default with SSL.
00213  *  WEB_CLIENT_SSL_SUPPORT - Whether the webclient library recognizes 'https://'
00214  *  SSL_DEFAULT_MAX_SESSION_AGE_TICKS
00215  *      - Maximum time to allow a sessions to be reused after it is created
00216  *      - A value of 0 will force a negotiation on every connection
00217  ******************************************************************************/
00218
00219 #ifdef NB_SSL_SUPPORTED
00220 #define SSL_TLS_SUPPORT (1)
00221 #define WEB_CLIENT_SSL_SUPPORT (1)
00222 #define SSL_DEFAULT_MAX_SESSION_AGE_TICKS (3600 * TICKS_PER_SECOND)   // 1 Hour
00223 #define TLS_CACHE_PEER_CERT_VALIDATIONS (1)
00224 #define ENABLE_ED25519 (1)
00225 //#define ENABLE_AUTOCERT_REGEN (1) // Enable if autogenerated certificates should renew automatically
      when expired
00226 #ifdef ENABLE_AUTOCERT_REGEN
00227 #define AUTO_CERT_GEN_CHECK (60 * TICKS_PER_SECOND)   // Every min
00228 #endif
00229
00230 /******************************************************************************
00231  * SSL uses 1.3 by default and will downgrade to support older versions
00232  * Minimum version can be set here
00233  *
00234  * 0 - TLS 1.3 (No downgrade allowed)
00235  * 1 - TLS 1.2
00236  * 2 - TLS 1.1
00237  * 3 - TLS 1.0
00238  * 4 - SSL 3.0 (Oldest, all versions will be supported)
00239  ******************************************************************************/
00240 #define SSL_MINIMUM_VERSION (1)
00241
00242 // Used to enable more robust ECC curves, at the expense of handshake speed
00243 #define ENABLE_ECC384 (1)
00244 //#define ENABLE_ECC521 (1)
00245
00246 // Used to enable RSA 4K Keys
00247 //#define ENABLE_RSA_4K (1)
00248
00249 /******************************************************************************
00250  * Use custom static malloc/free functions for SSL to speed up TLS performance
00251  *
00252  * This will claim a static 15KB chunk of memory from the memory specified
00253  *
00254  * Commented out/Undefined - Use default system malloc/free
00255  * 1 - Use custom malloc/free, utilizing fastest available memory
00256  * 2 - Use custom malloc/free, utilizing TCM memory (if available)
00257  * 3 - Use custom malloc/free, utilizing SRAM memory
00258  * 4 - Use custom malloc/free, utilizing SDRAM memory
00259  *
00260  ******************************************************************************/
```

```
00261 #define SSL_CUSTOM_MALLOC (1)
00262 // #define SSL_CUSTOM_MALLOC (2)
00263 // #define SSL_CUSTOM_MALLOC (3)
00264 // #define SSL_CUSTOM_MALLOC (4)
00265
00266 #endif
00267
00268 /*
00269  * FTPS Support
00270  *
00271  * Uncomment to enable SSL use for control and data channels
00272  */
00273 #ifdef NB_SSL_SUPPORTED
00274 #define FTPD_SSL_SUPPORT (1)
00275 #define FTPD_CLIENT_SSL_SUPPORT (1)   // Data and connection ports used are defined in
      SslClientSession.cpp
00276 #endif                                /* #ifdef NB_SSL_SUPPORTED */
00277
00278 /*
00279  * SSL client certificate checking supported
00280  *    Should be defined when client certificate checking is required
00281  *
00282  */
00283 #ifdef NB_SSL_SUPPORTED
00284 /* #define NB_SSL_CLIENT_CERTIFICATE_CHECKING_ENABLED       ( 1 ) */
00285 #endif /* #ifdef NB_SSL_SUPPORTED */
00286
00287 /*
00288  * SSH Supported
00289  *    Should be defined when SSH is included in library
00290  */
00291
00292 //#define NB_SSH_SUPPORTED   ( 1 )
00293
00294 /*
00295  * Security Random Number Support is required for SSL and SSH
00296  *
00297  */
00298 #if defined(NB_SSL_SUPPORTED) || defined(NB_SSH_SUPPORTED)
00299 #ifndef GATHER_RANDOM
00300 #define GATHER_RANDOM (1)
00301 #endif
00302 #endif
00303
00304 /*
00305  ******************************************************************************
00306  *
00307  * User QSPI driver
00308  *
00309  *    Uncomment to enable user QSPI driver defined in qspi.h
00310  *       Enabling the user QSPI driver disables the joint use of the QSPI by
00311  *       WLAN and SD/MMC.
00312  *
00313  ******************************************************************************
00314  */
00315 #define NB_ENABLE_USER_QSPI (1)
00316
00317 /*
00318  ******************************************************************************
00319  *
00320  * Enable Legacy config records for the primary interface
00321  *
00322  * Uncomment to enable
00323  *
00324  ******************************************************************************
00325  */
00326 #define SUPPORT_LEGACY_FIND (1)
00327 //#define SUPPORT_LEGACY_IPSETUP (1)
00328
00329 /*
00330  ******************************************************************************
00331  *
00332  * For 5441X products MOD5441X, SB800EX, Nano5441X.
00333  * update the Legacy config records for ethernet interface(s)
00334  *
00335  * This will keep the alternate monitor in sync for acces if one sets static IP Addresses etc..
00336  * This has NO effect on the any other platform.
00337  *
00338  ******************************************************************************
00339  */
00340 #define KEEP_LEGACY_CONFIG_UPDATED (1)
00341 /*
00342  ******************************************************************************
00343  *
00344  * For 5441X products MOD5441X, SB800EX, Nano5441X.
00345  * update the Legacy config boot uart when config is changed
00346  *
```

```
00347  * This will keep the alternate monitor in sync
00348  * This has NO effect on the any other platform.
00349  *
00350  ******************************************************************************
00351  */
00352 #define SET_LEGACY_CONFIG_UART  (1)
00353
00354
00355 //Should we run a DNS cache?
00356 //#define  DNS_CACHE (1)
00357
00358
00359 /*
00360  ******************************************************************************
00361  *
00362  * Disable Symetric Routing
00363  *
00364  * Uncomment to prevent all non-local incoming packet routes from being put
00365  * in the arp cache.
00366  *
00367  ******************************************************************************
00368  */
00369 /*  #define NO_SYMETRIC_ROUTING (1)  */
00370
00371 /*
00372  ******************************************************************************
00373  *
00374  * FEC ISR Error Counters
00375  *
00376  *    Uncomment to enable FEC error counters in ethernet.cpp and etherprint.cpp
00377  *
00378  ******************************************************************************
00379  */
00380 /*#define FEC_ISR_ERROR_COUNTERS (1) */
00381
00382 /*
00383  ******************************************************************************
00384  *
00385  * Library Constants
00386  *
00387  *    Please do not modify any definitions below this comment.
00388  *
00389  ******************************************************************************
00390  */
00391 /*
00392  * Library Versions
00393  *
00394  */
00395
00396 #define WARN_STUB (1)
00397
00398 #define NNDK_MAJOR
00399 #define NNDK_MINOR
00400 #define NNDK_PATCH
00401
00402 #define NB_VERSION_TEXT "3.3"
00403
00404 #include <predef-overload.h>
00405
00406 #endif /* #ifndef _PREDEF_H_ */
```

24.463 qspi.h File Reference

NetBurner MCF5441x QSPI API (Queued Serial Peripheral Interface)

```
#include <basictypes.h>
#include <nbrtos.h>
```

Classes

- struct QSPI_Record

  This struct contains the major variables/configurations used for a QSPI transfer.

Macros

- #define **QSPI_OK** (0)

  QSPI is busy.

- #define **QSPI_BUSY** (1)

  QSPI is ready.

Functions

- uint8_t QSPIInit (uint32_t baudRateInBps=2000000, uint8_t transferSizeInBits=8, uint8_t peripheralChip↩ Selects=0x0F, uint8_t chipSelectPolarity=1, uint8_t clockPolarity=0, uint8_t clockPhase=1, BOOL dout↩ Hiz=TRUE, uint8_t csToClockDelay=0, uint8_t delayAfterTransfer=0)

  Initialize Queued Serial Peripheral Interface (QSPI)
- uint8_t QSPIStart (puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount, OS_SEM *finishedSem=NULL)

  Start QSPI Data Transfer.
- BOOL QSPIdone (void)

  Can be called after QSPIStart(). Returns TRUE when transfer is complete. This is an alternative to using a semaphore.

24.463.1 Detailed Description

NetBurner MCF5441x QSPI API (Queued Serial Peripheral Interface)

24.464 qspi.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00064 #ifndef _QSPI_H
00065 #define _QSPI_H
00066 #include <basictypes.h>
00067 #include <nbrtos.h>
00068
00069 /*
00070  ****************************************************************************-
00071  *
00072  * Queued Serial Peripheral Interface (QSPI)
00073  *
00074  *    Interface consists of data output (QSPI_DOUT), data input (QSPI_DIN),
00075  *       serial clock (QSPI_CLK) & peripheral chip selects (QSPI_CS[3:0]).
00076  *
00077  *    SPI parameters are used to match the master (processor) with the slave
00078  *       device. Chip select can be used to multiplex devices.
00079  *
00080  *    The specific hardware initialization for MFCXXXX and MODXXXX are
00081  *       microprocessor and module specific and need to be correctly
00082  *       implemented in the module specific library for each SPI device.
00083  *
00084  *    More than one SPI device requires careful coordination of hardware use
00085  *       and synchronization of use.
00086  *
00087  ****************************************************************************-
00088  */
00089
00094 #define QSPI_OK (0)
00095 #define QSPI_BUSY (1)
00140 uint8_t QSPIInit(uint32_t baudRateInBps = 2000000,
00141                  uint8_t transferSizeInBits = 8,
00142                  uint8_t peripheralChipSelects = 0x0F,
00143                  uint8_t chipSelectPolarity = 1,
00144                  uint8_t clockPolarity = 0,
00145                  uint8_t clockPhase = 1,
00146                  BOOL doutHiz = TRUE,
00147                  uint8_t csToClockDelay = 0,
00148                  uint8_t delayAfterTransfer = 0);
00149
00165 uint8_t QSPIStart(puint8_t transmitBufferPtr, volatile uint8_t *receiveBufferPtr, uint32_t byteCount,
      OS_SEM *finishedSem = NULL);
00166
00178 BOOL QSPIdone(void);
00179
00185 struct QSPI_Record
00186 {
00187     volatile uint8_t *pQSPIRxbuf;
```

```
00188     volatile uint8_t *pQSPITxbuf;
00189     uint8_t BitsPerQueue;
00190     uint32_t QSPI_SizeLeft;
00191     uint16_t Command_Mask;
00192     OS_SEM *QSPI_Sem;
00193 };
00194
00195 #endif
00196
```

24.465 qspiBsp.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _QSPI_BSP_H_
00006 #define _QSPI_BSP_H_
00007 #include <basictypes.h>
00008 /*
00009  ******************************************************************************-
00010  *
00011  * Definitions
00012  *
00013  ******************************************************************************-
00014  */
00015 /*
00016  * Interrupt controller settings for:
00017  *      MOD5270
00018  *      SB70
00019  *      MOD5213
00020  *      PK70
00021  *      MOD5282
00022  *      MOD5234
00023  */
00024 /* Interrupt source (vector) */
00025 #define QSPI_INTERRUPT_SOURCE (18)
00026
00027 /* Interrupt level */
00028 #define QSPI_INTERRUPT_LEVEL (2)
00029
00030 /* Interrupt level */
00031 #define QSPI_INTERRUPT_PRIORITY (7)
00032
00033 /*
00034  * RAM addresses
00035  */
00036 #define QSPI_RAM_ENTRIES (16)
00037 #define QSPI_RAM_TRANSMIT_START (0x00)
00038 #define QSPI_RAM_RECEIVE_START (0x10)
00039 #define QSPI_RAM_COMMAND_START (0x20)
00040
00041 /*
00042  ******************************************************************************-
00043  *
00044  * Structures
00045  *
00046  ******************************************************************************-
00047  */
00048 /*
00049   QSPI Mode Register (QMR)
00050      mstr           - Master mode, must be one
00051      dohie          - DOUT high impedance or driven when idle?
00052      bits           - Transfer size 8 through 16 (1000-1111 & 0000 )
00053      cpol           - Clock polarity (is inactive level)
00054      cpha           - Clock phase of data capture, 0 leading, 1 falling
00055      baud           - Baud rate divider 0 disables, 2-255
00056
00057  */
00058 typedef struct _QmrFields
00059 {
00060     uint16_t mstr : 1;
00061     uint16_t dohie : 1;
00062     uint16_t bits : 4;
00063     uint16_t cpol : 1;
00064     uint16_t cpha : 1;
00065     uint16_t baud : 8;
00066
00067 } __attribute__((packed)) QmrFields;
00068
00069 typedef union _Qmr
00070 {
00071     QmrFields field;
00072     uint16_t content;
00073
```

```
00074 } __attribute__((packed)) Qmr;
00075
00076 /*
00077   QSPI Delay Register (QDLYR)
00078      spe             - Enable
00079      qcd             - Chip select to valid clock (dsck bit in command RAM)
00080      dtl             - Delay after transfer (dt bit in command RAM)
00081
00082  */
00083 typedef struct _QdlyrFields
00084 {
00085     uint16_t spe : 1;
00086     uint16_t qcd : 7;
00087     uint16_t dtl : 8;
00088
00089 } __attribute__((packed)) QdlyrFields;
00090
00091 typedef union _Qdlyr
00092 {
00093     QdlyrFields field;
00094     uint16_t content;
00095
00096 } __attribute__((packed)) Qdlyr;
00097
00098 /*
00099   QSPI Wrap Register (QWR)
00100      halt            - Halts transfer
00101      wren            - Wrap around enabled
00102      wrto            - Wrap around location
00103      csiv            - Chip select inactive level
00104      endqp           - End of queue pointer
00105      cptqp           - Completed queue entry pointer (R)
00106      newqp           - Start of queue pointer
00107
00108  */
00109 typedef struct _QwrFields
00110 {
00111     uint16_t halt : 1;
00112     uint16_t wren : 1;
00113     uint16_t wrto : 1;
00114     uint16_t csiv : 1;
00115     uint16_t endqp : 4;
00116     uint16_t cptqp : 4;
00117     uint16_t newqp : 4;
00118
00119 } __attribute__((packed)) QwrFields;
00120
00121 typedef union _Qwr
00122 {
00123     QwrFields field;
00124     uint16_t content;
00125
00126 } __attribute__((packed)) Qwr;
00127
00128 /*
00129   QSPI Interrupt Register (QIR)
00130      wcefb           - Write collision access error enable
00131      abrtb           - Abort access error enable
00132      abrtl           - Abort lockout
00133      wcefe           - Write collision interrupt enable
00134      abrte           - Abort interrupt enable
00135      spife           - Finished interrupt enable
00136      wcef            - Write collision error flag
00137      abrt            - Abort flag
00138      spif            - Finished flag
00139
00140  */
00141 typedef struct _QirFields
00142 {
00143     uint16_t wcefb : 1;
00144     uint16_t abrtb : 1;
00145     uint16_t mbz_13 : 1;
00146     uint16_t abrtl : 1;
00147     uint16_t wcefe : 1;
00148     uint16_t abrte : 1;
00149     uint16_t mbz_09 : 1;
00150     uint16_t spife : 1;
00151     uint16_t mbz_04_07 : 4;
00152     uint16_t wcef : 1;
00153     uint16_t abrt : 1;
00154     uint16_t mbz_01 : 1;
00155     uint16_t spif : 1;
00156
00157 } __attribute__((packed)) QirFields;
00158
00159 typedef union _Qir
00160 {
```

```
00161     QirFields field;
00162     uint16_t content;
00163
00164 } __attribute__((packed)) Qir;
00165
00166 /*
00167   QSPI Command Register(s) (QCR)
00168      cont           - Continuous (0-stop, 1-go)
00169      bitse          - Eight bits or qmr.field.bits
00170      dt             - Delay after transfer (0-default, 1-qdlyr.dtl)
00171      dsck           - Chip select to valid clock (0-1/2 clock, 1-qdlyr.qcd)
00172      qspi_cs        - Chip select mask [3:0]
00173
00174  */
00175 typedef struct _QcrFields
00176 {
00177     uint16_t cont : 1;
00178     uint16_t bitse : 1;
00179     uint16_t dt : 1;
00180     uint16_t dsck : 1;
00181     uint16_t qspi_cs : 4;
00182     uint16_t mbz_00_07 : 8;
00183
00184 } __attribute__((packed)) QcrFields;
00185
00186 typedef union _Qcr
00187 {
00188     QcrFields field;
00189     uint16_t content;
00190
00191 } __attribute__((packed)) Qcr;
00192
00193 /*
00194  ******************************************************************************
00195  *
00196  * Routines
00197  *
00198  ******************************************************************************
00199  */
00200
00201 /*
00202  ******************************************************************************
00203
00204    Interrupt service routine (ISR)
00205
00206    Parameters:
00207       None
00208
00209    Return:
00210       None
00211
00212    Notes:
00213       None
00214
00215  ******************************************************************************
00216  */
00217 typedef void(QspiIsr)(void);
00218
00219 /*
00220  ******************************************************************************
00221
00222    Setup QSPI module
00223
00224    Parameters:
00225       setHighDrive               - Drive strength TRUE high, FALSE low
00226
00227    Return:
00228       0 - OK, all else problems
00229
00230    Notes:
00231       Set assigned pins QSPI_DOUT, QSPI_DIN, QSPI_CLK and drive strength install
00232       BSP Isr.
00233
00234  ******************************************************************************
00235  */
00236 int QspiSetupHardware(BOOL setHighDrive);
00237
00238 /*
00239  ******************************************************************************
00240
00241    Attach chip select to QSPI module
00242
00243    Parameters:
00244       controlledChipSelects      - Chip selects [3:0] respectively
00245
00246    Return:
00247       0 - OK, all else problems
```

```
00248
00249    Notes:
00250       Driver user can let the module assert/de-assert chip select by attaching
00251       or use QspiSelectChip and QspiDeselectChip based on device requirements.
00252       The last call to QspiAttachChipSelects/QspiDetachChipSelects is the one
00253       controlling.
00254
00255  ***************************************************************************
00256  */
00257 int QspiAttachChipSelects(uint8_t controlledChipSelects);
00258
00259 /*
00260  ***************************************************************************
00261
00262    Detach chip selects from QSPI module set as GPIO pins
00263
00264    Parameters:
00265       controlledChipSelects      - Chip selects [3:0] respectively
00266
00267    Return:
00268       0 - OK, all else problems
00269
00270    Notes:
00271       User can assert/de-assert chip select by detaching and calling
00272       QspiAssertChipSelects and QspiDeassertChipSelects.
00273       The last call to QspiAttachChipSelects/QspiDetachChipSelects is the one
00274       controlling.
00275
00276  ***************************************************************************
00277  */
00278 int QspiDetachChipSelects(uint8_t controlledChipSelects);
00279
00280 /*
00281  ***************************************************************************
00282
00283    Asserts chip selects
00284
00285    Parameters:
00286       controlledChipSelects   - Chip selects [3:0] respectively
00287       isChipSelectActiveLow   - Chips select phase
00288                                 TRUE Active low, inactive high
00289                                 TRUE Active high, inactive low
00290
00291
00292    Return:
00293       Notes
00294
00295    Notes:
00296       Driver must call QspiAttachChipSelects/QspiDetachChipSelects.
00297
00298  ***************************************************************************
00299  */
00300 void QspiAssertChipSelects(uint8_t controlledChipSelects, BOOL isChipSelectActiveLow);
00301
00302 /*
00303  ***************************************************************************
00304
00305    Deasserts chip selects
00306
00307    Parameters:
00308       controlledChipSelects   - Chip selects [3:0] respectively
00309       isChipSelectActiveLow   - Chips select phase
00310                                 TRUE Active low, inactive high
00311                                 TRUE Active high, inactive low
00312
00313
00314    Return:
00315       Notes
00316
00317    Notes:
00318       Driver must call QspiAttachChipSelects/QspiDetachChipSelects.
00319
00320  ***************************************************************************
00321  */
00322 void QspiDeassertChipSelects(uint8_t controlledChipSelects, BOOL isChipSelectActiveLow);
00323
00324 /*
00325  ***************************************************************************
00326
00327    Installs interrupt service routine
00328
00329    Parameters:
00330       isr                       - "C" interrupt service routine
00331
00332    Return:
00333       None
00334
```

```
00335    Notes:
00336       None
00337
00338  ***************************************************************************
00339  */
00340 void QspiSetupIsr(QspiIsr isr);
00341
00342 /*
00343  ***************************************************************************
00344
00345    Get baud rate setting
00346
00347    Parameters:
00348       baudRateInMhz              - Baud rate requested in Mhz
00349
00350    Return:
00351       Lowest near integral setting within processor/module limits.
00352
00353    Notes:
00354       Base on processor and clock speed.
00355
00356  ***************************************************************************
00357  */
00358 uint8_t QspiGetBaudSetting(unsigned long baudRateInMhz);
00359
00360 /*
00361  ***************************************************************************
00362
00363    Get current baud rate setting
00364
00365    Parameters:
00366       None
00367
00368    Return:
00369       Baud rate setting in Mhz
00370
00371    Notes:
00372       Base on processor and clock speed.
00373
00374  ***************************************************************************
00375  */
00376 unsigned long QspiGetCurrentBaudSetting(void);
00377
00378 /*
00379  ***************************************************************************
00380
00381    Enable module interrupt
00382
00383    Parameters:
00384       None
00385
00386    Return:
00387       None
00388
00389    Notes:
00390       None
00391
00392  ***************************************************************************
00393  */
00394 void QspiEnableIsr(void);
00395
00396 /*
00397  ***************************************************************************
00398
00399    Disable module interrupt
00400
00401    Parameters:
00402       None
00403
00404    Return:
00405       None
00406
00407    Notes:
00408       None
00409
00410  ***************************************************************************
00411  */
00412 void QspiDisableIsr(void);
00413
00414 /*
00415  ***************************************************************************
00416
00417    Get module registers
00418
00419    Parameters:
00420       None
00421
```

```
00422    Return:
00423       None
00424
00425    Notes:
00426       None
00427
00428  ***************************************************************************
00429  */
00430 void QspiGetRegisters(volatile uint16_t **qmrPtr,
00431                       volatile uint16_t **qdlyrPtr,
00432                       volatile uint16_t **qwrPtr,
00433                       volatile uint16_t **qirPtr,
00434                       volatile uint16_t **qarPtr,
00435                       volatile uint16_t **qdrPtr);
00436
00437 #endif /* _QSPI_BSP_H_ */
```

24.466 qspiShared.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _QSPI_SHARED_H_
00006 #define _QSPI_SHARED_H_
00007 #include <nbrtos.h>
00008 /*
00009  ***************************************************************************
00010  *
00011  * Queued Serial Peripheral Interface (QSPI)
00012  *
00013  *    Interface consists of data output (QSPI_DOUT), data input (QSPI_DIN),
00014  *       serial clock (QSPI_CLK) & optionally peripheral chip selects
00015  *       (QSPI_CS[3:0]).
00016  *
00017  *    SPI parameters are used to match the master (processor) with the slave
00018  *       device. Chip select can be used to multiplex devices.
00019  *
00020  *    The specific hardware initialization for MFCXXXX and MODXXXX are
00021  *       microprocessor and module specific and need to be correctly
00022  *       implemented in the module specific library for each SPI device, for
00023  *       example Nburn\MOD5270\system\wifi70_bsp.cpp.
00024  *
00025  *    More than one SPI device requires careful coordination of hardware use
00026  *       and synchronization of use.
00027  *
00028  ***************************************************************************
00029  */
00030
00031 /*
00032  ***************************************************************************
00033  *
00034  * Definitions
00035  *
00036  ***************************************************************************
00037  */
00038 /* Chip selects
00039  *    Can connect to QSPI module as supported by the processor.
00040  */
00041 #define QSPI_CHIP_SELECT_0 (0x1)
00042 #define QSPI_CHIP_SELECT_1 (0x2)
00043 #define QSPI_CHIP_SELECT_2 (0x4)
00044 #define QSPI_CHIP_SELECT_3 (0x8)
00045
00046 /* Sets of settings available */
00047 #define QSPI_SETTINGS_MAXIMUM (4)
00048
00049 /* Block size in bytes */
00050 #define QSPI_BLOCK_SIZE_IN_BYTES (16)
00051
00052 /*
00053  ***************************************************************************
00054  *
00055  * Enumerations
00056  *
00057  ***************************************************************************
00058  */
00059 /*
00060  * Transfer size in bits, must match settings for qmr.bits
00061  */
00062 typedef enum _SpiTransferSize
00063 {
00064     TransferSize_16_bits = 0x0,
00065     TransferSize_8_bits = 0x8
00066
```

```
00067 } SpiTransferSize;
00068
00069 /*
00070  ***************************************************************************
00071  *
00072  * Runtime Library Routines
00073  *
00074  ***************************************************************************
00075  */
00076
00077 /*
00078  ***************************************************************************
00079  *
00080  * Initialize shared QSPI driver.
00081  *
00082  *    Parameters:
00083  *       isDriveStrengthHigh     - Output drive strength TRUE high, FALSE low
00084  *       isDoutHighImpedance     - DOUT high impedance between transactions
00085  *                                 TRUE high impedance, FALSE driven
00086  *       isClockInactiveHigh     - Clock inactive state
00087  *                                 TRUE high, FALSE low
00088  *       isDataChangeOnLeading   - Clock phase
00089  *                                 TRUE Data changed on leading edge,
00090  *                                    captured on the following edge.
00091  *                                 FALSE Data captured on leading edge,
00092  *                                    changed on the following edge.
00093  *       isChipSelectActiveLow   - Chips select phase
00094  *                                 TRUE Active low, inactive high
00095  *                                 TRUE Active high, inactive low
00096  *       moduleChipSelects       - Chip selects to integrate with QSPI module.
00097  *
00098  *    Return:
00099  *       None
00100  *
00101  *    Notes:
00102  *       Must be called at least once before any use of the SPI bus.
00103  *
00104  ***************************************************************************
00105  */
00106 extern void QspiInitialize(BOOL isDriveStrengthHigh,
00107                            BOOL isHighImpedance,
00108                            BOOL isClockInactiveHigh,
00109                            BOOL isDataChangeOnLeading,
00110                            BOOL isChipSelectActiveLow,
00111                            uint8_t moduleChipSelects);
00112
00113 /*
00114  ***************************************************************************
00115  *
00116  * Allocate SPI bus
00117  *
00118  *    Parameters:
00119  *       spiSetting              - Settings handle
00120  *       timeout                 - Timeout in ticks
00121  *
00122  *    Return:
00123  *       OS_NO_ERR OK, else probably timed out.
00124  *
00125  *    Notes:
00126  *       Must release when idle to support sharing. Will suspend task until
00127  *       granted.
00128  *
00129  ***************************************************************************
00130  */
00131 extern uint8_t QspiAllocate(int spiSetting, uint16_t timeout);
00132
00133 /*
00134  ***************************************************************************
00135  *
00136  * Releases SPI bus
00137  *
00138  *    Parameters:
00139  *       None
00140  *
00141  *    Return:
00142  *       None
00143  *
00144  *    Notes:
00145  *       Will support granting it to any waiting tasks.
00146  *
00147  ***************************************************************************
00148  */
00149 extern void QspiRelease(void);
00150
00151 /*
00152  ***************************************************************************
00153  *
```

```
00154  * Creates bus parameters profile used for transfers.
00155  *
00156  *    Parameters:
00157  *       baudRateInMhz           - Baud rate in Mhz
00158  *       qspiTransferSize        - Bits transfered for each entry in queue
00159  *                                 Currently 8 and 16 bits
00160  *       delayFromCstoClockValid - Delay from assertion of the chip selects to
00161  *                                 valid clock transition
00162  *       delayBetweenTransfers   - Delay after the serial transfer
00163  *       chipSelects             - Chips select mask
00164  *       idleValue               - Value while driving idle bus
00165  *
00166  *    Return:
00167  *       >=0 handle assigned to these settings.
00168  *       <0 Problems with settings.
00169  *
00170  *    Notes:
00171  *       Baud rate has minimum and maximum rates depending on the module.
00172  *
00173  ****************************************************************************
00174  */
00175 extern int QspiCreateSetting(int baudRateInMhz,
00176                              SpiTransferSize qspiTransferSize,
00177                              uint16_t delayFromCstoClockValid,
00178                              uint16_t delayBetweenTransfers,
00179                              uint8_t chipSelects,
00180                              uint16_t idleValue);
00181
00182 /*
00183  ****************************************************************************
00184  *
00185  * Sets bus parameters used for transfers.
00186  *
00187  *    Parameters:
00188  *       spiSetting              - Settings handle
00189  *       baudRateInMhz           - Baud rate in Mhz
00190  *
00191  *    Return:
00192  *       None
00193  *
00194  *    Notes:
00195  *       Can be done any time, only used after SpiSet
00196  *
00197  ****************************************************************************
00198  */
00199 extern void QspiChangeBaudRate(int spiSettings, int baudRateInMhz);
00200
00201 /*
00202  ****************************************************************************
00203  *
00204  * Gets current baud rate in Mhz for the current active setting.
00205  *
00206  *    Parameters:
00207  *       None
00208  *
00209  *    Return:
00210  *       Actual baud rate in Mhz
00211  *
00212  *    Notes:
00213  *       Can be done any time, only used after SpiSet
00214  *
00215  ****************************************************************************
00216  */
00217 extern unsigned long QspiGetBaudRate(void);
00218
00219 /*
00220  ****************************************************************************
00221  *
00222  * Change bus setup.
00223  *
00224  *    Parameters:
00225  *       spiSetting              - Settings handle
00226  *
00227  *    Return:
00228  *       None
00229  *
00230  *    Notes:
00231  *       None
00232  *
00233  ****************************************************************************
00234  */
00235 extern void QspiChangeSetting(int spiSetting);
00236
00237 /*
00238  ****************************************************************************
00239  *
00240  * Send, receive or exchange bytes (8 bits) with slave
```

```
00241  *
00242  *    Parameters:
00243  *       transmitBufferPtr       - Buffer of bytes to send, NULL for receiving
00244  *       receiveBufferPtr        - Buffer to receive data, NULL for sending
00245  *       byteCount               - Count of bytes to send or receive
00246  *       finishedSemPtr          - Optional semaphore to set when completed
00247  *
00248  *    Return:
00249  *       None
00250  *
00251  *    Notes:
00252  *       None
00253  *
00254  ***************************************************************************
00255  */
00256 extern void QspiExchangeBytes(uint8_t *transmitBufferPtr, uint8_t *receiveBufferPtr, uint32_t
      byteCount, OS_SEM *finishedSemPtr);
00257
00258 /*
00259  ***************************************************************************
00260  *
00261  * Send, receive or exchange data word with slave (9-16 bits)
00262  *
00263  *    Parameters:
00264  *       transmitBufferPtr       - Buffer of words to send, NULL for receiving
00265  *       receiveBufferPtr        - Buffer to receive data, NULL for sending
00266  *       byteCount               - Count of bytes to send or receive
00267  *       finishedSemPtr          - Optional semaphore to set when completed
00268  *
00269  *    Return:
00270  *       None
00271  *
00272  *    Notes:
00273  *       None
00274  *
00275  ***************************************************************************
00276  */
00277 extern void QspiExchangeWords(uint16_t *transmitBufferPtr, uint16_t *receiveBufferPtr, uint32_t
      byteCount, OS_SEM *finishedSemPtr);
00278
00279 /*
00280  ***************************************************************************
00281  *
00282  * Setup chip select(s) as user not module controlled
00283  *
00284  *    Parameters:
00285  *       chipSelects             - Chips select mask
00286  *
00287  *    Return:
00288  *       None
00289  *
00290  *    Notes:
00291  *       Polarity set by isChipSelectActiveLow parameter of QspiInitialize.
00292  *
00293  ***************************************************************************
00294  */
00295 extern void QspiUserChipSelects(uint8_t chipSelects);
00296
00297 /*
00298  ***************************************************************************
00299  *
00300  * Select SPI device(s) by asserting chip select(s)
00301  *
00302  *    Parameters:
00303  *       chipSelects             - Chips select mask
00304  *
00305  *    Return:
00306  *       None
00307  *
00308  *    Notes:
00309  *       None
00310  *
00311  ***************************************************************************
00312  */
00313 extern void QspiSelectDevice(uint8_t chipSelects);
00314
00315 /*
00316  ***************************************************************************
00317  *
00318  * Release SPI device(s) by deasserting chip select(s)
00319  *
00320  *    Parameters:
00321  *       chipSelects             - Chips select mask
00322  *
00323  *    Return:
00324  *       None
00325  *
```

```
00326  *    Notes:
00327  *       None
00328  *
00329  ***************************************************************************
00330  */
00331 extern void QspiReleaseDevice(uint8_t chipSelects);
00332
00333 /*
00334  ***************************************************************************
00335  *
00336  * SDIO in the SPI Mode Functions (Driver use only)
00337  *
00338  ***************************************************************************
00339  */
00340 /*
00341  ***************************************************************************
00342  *
00343  * Send command, process acknowledgement and send or receive data
00344  *
00345  *    Parameters:
00346  *       commandBufferPtr        - Buffer containing SDIO command (6 bytes)
00347  *       r5ResponsePtr           - Buffer containing two bytes for SPI response
00348  *       sendData                - TRUE send data else receive data
00349  *       dataBufferPtr           - Buffer of bytes to send or recieve data
00350  *       byteCount               - Count of bytes to send or receive
00351  *       responseTimeout         - Response delay in ticks
00352  *
00353  *    Return:
00354  *       TRUE - success, FALSE response timeout
00355  *
00356  *    Notes:
00357  *       SDIO using the SPI mode.
00358  *
00359  ***************************************************************************
00360  */
00361 extern BOOL QspiSdioDataExchange(uint8_t *commandBufferPtr,
00362                                  uint8_t *r5ResponsePtr,
00363                                  BOOL sendData,
00364                                  uint8_t *dataBufferPtr,
00365                                  uint32_t byteCount,
00366                                  uint32_t responseTimeout);
00367
00368 #endif /* #ifndef _QSPI_SHARED_H_ */
```

24.467 random.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef RANDOM_H
00006 #define RANDOM_H
00007
00008 #include <basictypes.h>
00009
00010 int RandomValid();
00011 uint8_t GetRandomByte();
00012 uint16_t GetRandomWord();
00013 uint32_t GetRandomDword();
00014 #endif
```

24.468 randseed.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_RANDOM_H
00006 #define _NB_RANDOM_H
00007
00008 #ifdef GATHER_RANDOM
00009 // ENTROP_POOL_SIZE *MUST* be a multiple of 16
00010 #define ENTROPY_POOL_SIZE 256
00011
00012 extern volatile uint32_t rnd_cnt;
00013 extern volatile uint16_t rnd_collect[ENTROPY_POOL_SIZE];
00014 #else
00015 #endif
00016 #endif
```

24.469 remoteconsole.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef NB_RMTCONSOLE_H
00005 #define NB_RMTCONSOLE_H
00006 void EnableRemoteConsole();
00007 #endif
```

24.470 sdio.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SDIO_H_
00006 #define _SDIO_H_
00007 #include <basictypes.h>
00008 #include <sys/types.h>
00009
00010 /*
00011  ****************************************************************************
00012  *
00013  * Reference
00014  *    SDIO Simplified Specification, SD Specification, Part E1, Version 2.00
00015  *       February 8, 2007, Technical Committee, SD Card Association
00016  *
00017  *    Physical Layer Simplified Specification, SD Specification, Part 1,
00018  *       Version 2.00, September 25, 2006, Technical Committee
00019  *       SD Card Association
00020  *
00021  ****************************************************************************
00022  */
00023 /*
00024  ****************************************************************************
00025  *
00026  * Debugging
00027  *
00028  *    Needs to be uncommented to support these features
00029  *
00030  ****************************************************************************
00031  */
00032 /* Library debugging switch */
00033 /* #define SDIO_DEBUG                  ( 1 ) */
00034
00035 /*
00036  ****************************************************************************
00037  *
00038  * Runtime Library Definitions
00039  *
00040  ****************************************************************************
00041  */
00042 /* Bus type */
00043 #define SDIO_SPI_BUS (1)
00044
00045 /*
00046  ****************************************************************************
00047  *
00048  * Command and Response Definitions
00049  *
00050  ****************************************************************************
00051  */
00052 /* Command size in bytes */
00053 #define SDIO_COMMAND_SIZE (6)
00054
00055 /* Block size */
00056 #define SDIO_BLOCK_SIZE (512)
00057
00058 /* Direction */
00059 #define SDIO_CARD_TO_HOST (0x0)
00060 #define SDIO_HOST_TO_CARD (0x1)
00061
00062 /* Command Index */
00063 #define SDIO_GO_IDLE (0x00)
00064 #define SDIO_IO_SEND_OPERATIONAL_CONDITION (0x05)
00065 #define SDIO_IO_RW_DIRECT (0x34)
00066 #define SDIO_IO_RW_EXTENDED (0x35)
00067 #define SDIO_CRC_ON_OFF (0x3B)
00068
00069 /* Read/Write */
00070 #define SDIO_READ (0)
00071 #define SDIO_WRITE (1)
00072
```

```
00073 /* Function Number (I/O card dependent) */
00074 #define SDIO_FUNCTION_0 (0)
00075 #define SDIO_FUNCTION_1 (1)
00076
00077 /* Read after write */
00078 #define SDIO_READ_OR_WRITE_ONLY (0)
00079 #define SDIO_RAW (1)
00080
00081 /* Block Mode */
00082 #define SDIO_BYTE_MODE (0)
00083 #define SDIO_BLOCK_MODE (1)
00084
00085 /* OP code */
00086 #define SDIO_FIXED_ADDRESS (FALSE)
00087 #define SDIO_INCREMENTING_ADDRESS (TRUE)
00088
00089 /* Block size */
00090 #define SDIO_BLOCK_SIZE_NONE (0)
00091
00092 /*
00093  ******************************************************************************
00094  *
00095  * Common I/O Area (CIA) register addresses (Function 0)
00096  *
00097  ******************************************************************************
00098  */
00099 /* Card Common Control Registers (CCCR) */
00100 #define SDIO_CIA_CCCR_CCCR_SDIO_REVISION (0x00000)
00101 #define SDIO_CIA_CCCR_SD_SPEC_REVISION (0x00001)
00102 #define SDIO_CIA_CCCR_IO_ENABLE (0x00002)
00103 #define SDIO_CIA_CCCR_IO_READY (0x00003)
00104 #define SDIO_CIA_CCCR_INTERRUPT_ENABLE (0x00004)
00105 #define SDIO_CIA_CCCR_INTERRUPT_PENDING (0x00005)
00106 #define SDIO_CIA_CCCR_IO_ABORT (0x00006)
00107 #define SDIO_CIA_CCCR_BUS_INTERFACE_CONTROL (0x00007)
00108 #define SDIO_CIA_CCCR_CARD_CAPABILITY (0x00008)
00109 #define SDIO_CIA_CCCR_COMMON_CIS_POINTER_LSB (0x00009)
00110 #define SDIO_CIA_CCCR_COMMON_CIS_POINTER_CSB (0x0000A)
00111 #define SDIO_CIA_CCCR_COMMON_CIS_POINTER_MSB (0x0000B)
00112 #define SDIO_CIA_CCCR_BUS_SUSPEND (0x0000C)
00113 #define SDIO_CIA_CCCR_FUNCTION_SELECT (0x0000D)
00114 #define SDIO_CIA_CCCR_EXEC_FLAGS (0x0000E)
00115 #define SDIO_CIA_CCCR_READY_FLAGS (0x0000F)
00116 #define SDIO_CIA_CCCR_FN0_BLOCK_SIZE_LSB (0x00010)
00117 #define SDIO_CIA_CCCR_FN0_BLOCK_SIZE_MSB (0x00011)
00118 #define SDIO_CIA_CCCR_POWER_CONTROL (0x00012)
00119
00120 /*
00121  ******************************************************************************
00122  *
00123  * Common I/O Area (CIA) register addresses (Function 1)
00124  *
00125  ******************************************************************************
00126  */
00127 #define SDIO_CIA_CCCR_FN1_BLOCK_SIZE_LSB (0x00110)
00128 #define SDIO_CIA_CCCR_FN1_BLOCK_SIZE_MSB (0x00111)
00129 /*
00130  ******************************************************************************
00131  *
00132  * Runtime Library Enumerations and Structures
00133  *
00134  ******************************************************************************
00135  */
00136 /*
00137    SDIO Bus type
00138       Sdio                  - SD 4-bit mode
00139       SdioSpiMode           - SPI mode
00140
00141  */
00142 typedef enum _SdioBusType
00143 {
00144     Sdio,
00145     SdioSpiMode
00146
00147 } SdioBusType;
00148
00149 /*
00150    SDIO Response type
00151       R4                    - IO_SEND_OP_COND Response
00152       R5                    - IO_RW_DIRECT Response
00153
00154  */
00155 typedef enum _SdioResponseType
00156 {
00157     R4,
00158     R5
00159
```

```
00160 } SdioResponseType;
00161
00162 /*
00163    Bus token for user
00164       butType              - Bus type in use for this token
00165
00166       *** SPI ***
00167       spiSetting           - Shared SPI setting
00168       spiConnectTimeout    - Bus allocation timeout period in ticks
00169       spiResponseTimeout   - Command response timeout period in ticks
00170       idleByteCount        - Bytes required to idle bus
00171       idleFillValue        - Data to transmit that idles the bus
00172
00173       *** SDIO Future ***
00174
00175  */
00176 typedef struct _SdioBusToken
00177 {
00178     SdioBusType busType;
00179     int spiSetting;
00180     uint32_t spiConnectTimeout;
00181     uint32_t spiResponseTimeout;
00182     ssize_t idleByteCount;
00183     uint16_t idleFillValue;
00184
00185 } __attribute__((packed)) SdioBusToken;
00186
00187 /*
00188  ******************************************************************************
00189  *
00190  * Command and Response Structures
00191  *
00192  ******************************************************************************
00193  */
00194 /*
00195    Start bit, direction and command
00196       startBit             - Start bit, always 0
00197       direction            - 0 card to host, 1 host to card
00198       index                - Command index
00199
00200  */
00201 typedef struct _CommandIndex
00202 {
00203     uint8_t startBit : 1;
00204     uint8_t direction : 1;
00205     uint8_t index : 6;
00206
00207 } __attribute__((packed)) CommandIndex;
00208
00209 /*
00210    CRC and end
00211       crc7                 - 7 bits of CRC data
00212       endBit               - End bit, always 1
00213
00214  */
00215 typedef struct _Crc
00216 {
00217     uint8_t crc7 : 7;
00218     uint8_t endBit : 1;
00219
00220 } __attribute__((packed)) Crc;
00221
00222 /*
00223    Response Flags IAW SD Physical Specification
00224       comCrcError          - 1 yes else 0
00225       illegalCommand       - 1 yes else 0
00226       ioCurrentState       - Current I/O state
00227       error                - Unknown error
00228       rfu                  - RFU always 0
00229       functionNumberError  - 1 yes else 0
00230       outOfRange           - 1 yes else 0
00231
00232  */
00233 typedef struct _ResponseFlags
00234 {
00235     uint8_t comCrcError : 1;
00236     uint8_t illegalCommand : 1;
00237     uint8_t ioCurrentState : 2;
00238     uint8_t error : 1;
00239     uint8_t rfu : 1;
00240     uint8_t functionNumberError : 1;
00241     uint8_t outOfRange : 1;
00242
00243 } __attribute__((packed)) ResponseFlags;
00244
00245 /*
00246    Go idle state (CMD0)
```

```
00247       commandIndex         - Start, direction and command index
00248       stuff                - MBZ
00249       crc                  - CRC and end bit
00250
00251  */
00252 typedef struct _GoIdleStateCommand_CMD0
00253 {
00254     CommandIndex commandIndex;
00255     uint8_t stuff[4];
00256     Crc crc;
00257
00258 } __attribute__((packed)) GoIdleStateCommand_CMD0;
00259
00260 /*
00261    I/O Send Operational Condition
00262       crc                  - CRC and end bit
00263       ocr                  - Operations condition register
00264       stuffBits            - MBZ
00265       commandIndex         - Start, direction and command index
00266
00267  */
00268 typedef struct _IoSendOpCondCommand_CMD5
00269 {
00270     CommandIndex commandIndex;
00271     uint8_t stuffBits;
00272     uint8_t ocr[3];
00273     Crc crc;
00274
00275 } __attribute__((packed)) IoSendOpCondCommand_CMD5;
00276
00277 /*
00278    CRC toggle command (CMD59)
00279       crc                  - CRC and end bit
00280       ocr                  - Operations condition register
00281       stuffBits            - MBZ
00282       commandIndex         - Start, direction and command index
00283
00284  */
00285 typedef struct _CrcOnOffCommand_CMD59
00286 {
00287     CommandIndex commandIndex;
00288     uint8_t stuffBits;
00289     uint8_t ocr[3];
00290     Crc crc;
00291
00292 } __attribute__((packed)) CrcOnOffCommand_CMD59;
00293
00294 /*
00295    Modified R1 IAW SD Physical Specification
00296       start                - Start bit, always 0
00297       parameterError       - 1 yes else 0
00298       rfu                  - RFU always 0
00299       functionNumberError  - 1 yes else 0
00300       comCrcError          - 1 yes else 0
00301       illegalCommand       - 1 yes else 0
00302       rfu_1                - RFU always 0
00303       idleState            - Idle state, always 1
00304  */
00305 typedef struct _ModifiedR1
00306 {
00307     uint8_t startBit : 1;
00308     uint8_t parameterError : 1;
00309     uint8_t rfu : 1;
00310     uint8_t functionNumberError : 1;
00311     uint8_t comCrcError : 1;
00312     uint8_t illegalCommand : 1;
00313     uint8_t rfu_1 : 1;
00314     uint8_t idleState : 1;
00315
00316 } __attribute__((packed)) ModifiedR1;
00317
00318 /*
00319    Operate bit, number of I/O functions and memory present
00320       operate              - 1 card is ready to operate
00321       ioFunctions          - Number of I/O functions supported
00322       memoryPresent        - 1 SD memory else 0
00323       stuffBits            - MBZ
00324
00325  */
00326 typedef struct _NumberIoFunctions
00327 {
00328     uint8_t operate : 1;
00329     uint8_t ioFunctions : 3;
00330     uint8_t memoryPresent : 1;
00331     uint8_t stuffBits : 3;
00332
00333 } __attribute__((packed)) NumberIoFunctions;
```

```
00334
00335 /*
00336    I/O Send Operational Condition Response in SPI mode (R4)
00337       modifiedR1           - Modified R1
00338       numberIoFunctions    - Number of I/O functions supported
00339       ocr                  - Operations condition register
00340
00341  */
00342 typedef struct _IoSendOpCondResponseSpiMode_R4
00343 {
00344     ModifiedR1 modifiedR1;
00345     NumberIoFunctions numberIoFunctions;
00346     uint8_t ocr[3];
00347
00348 } __attribute__((packed)) IoSendOpCondResponseSpiMode_R4;
00349
00350 /*
00351    I/O Send Operational Condition Response  (R4)
00352       startBit             - Start bit, always 0
00353       direction            - 0 card to host
00354       reserved_02_07       - 1s
00355       numberIoFunctions    - Number of I/O functions supported
00356       ocr                  - Operations condition register
00357       reserved_45_46       - 1S
00358       endBit               - End bit, always 1
00359
00360  */
00361 typedef struct _IoSendOpCondResponse_R4
00362 {
00363     uint8_t startBit : 1;
00364     uint8_t direction : 1;
00365     uint8_t reserved_02_07 : 6;
00366     NumberIoFunctions numberIoFunctions;
00367     uint8_t ocr[3];
00368     uint8_t reserved_45_46 : 7;
00369     uint8_t endBit : 1;
00370
00371 } __attribute__((packed)) IoSendOpCondResponse_R4;
00372
00373 /*
00374    R/W flag, function number and raw flag
00375       rwFlag               - 0 read, 1 write
00376       functionNumber       - I/O card area
00377       rawFlag              - Read after write, 1 yes
00378       stuff                - MBZ
00379       registerAddress_bits_16thru15  - Register address bits 15 and 16
00380
00381  */
00382 typedef struct _Function
00383 {
00384     uint8_t rwFlag : 1;
00385     uint8_t functionNumber : 3;
00386     uint8_t rawFlag : 1;
00387     uint8_t stuff : 1;
00388     uint8_t registerAddress_bits_16thru15 : 2;
00389
00390 } __attribute__((packed)) Function;
00391
00392 /*
00393    Register address
00394       address              - Register address bits 0 through 14
00395       stuff                - MBZ
00396
00397  */
00398 typedef struct _RegisterAddress
00399 {
00400     unsigned short address : 15;
00401     unsigned short stuff : 1;
00402
00403 } __attribute__((packed)) RegisterAddress;
00404
00405 /*
00406    I/O read/write direct command
00407       commandIndex         - Start, direction and command index
00408       function             - R/W flag, function number and raw flag
00409       registerAddress      - Register address bits 0 through 14
00410       dataOrStuff          - Write data byte else MBZ
00411       stuffBits            - MBZ
00412
00413  */
00414 typedef struct _IoRwDirectCommand_CMD52
00415 {
00416     CommandIndex commandIndex;
00417     Function function;
00418     RegisterAddress registerAddress;
00419     uint8_t dataOrStuff;
00420     Crc crc;
```

```
00421
00422 } __attribute__((packed)) IoRwDirectCommand_CMD52;
00423
00424 /*
00425    I/O read/write direct response
00426       commandIndex         - Start, direction and command index
00427       stuff                - MBZ
00428       responseFlags        - Status of SDIO card
00429       readOrWriteData      - RAW read else ignore
00430       data                 - Read data
00431       crc                  - CRC
00432
00433  */
00434 typedef struct _IoRwDirectResponse_R5
00435 {
00436     CommandIndex commandIndex;
00437     uint8_t stuff[2];
00438     ResponseFlags responseFlags;
00439     uint8_t readOrWriteData;
00440     uint8_t data;
00441     Crc crc;
00442
00443 } __attribute__((packed)) IoRwDirectResponse_R5;
00444
00445 /*
00446    I/O read/write direct response
00447       modifiedR1           - Modified R1
00448       data                 - Read data
00449
00450  */
00451 typedef struct _IoRwDirectResponseSpiMode_R5
00452 {
00453     ModifiedR1 modifiedR1;
00454     uint8_t data;
00455
00456 } __attribute__((packed)) IoRwDirectResponseSpiMode_R5;
00457
00458 /*
00459    R/W flag, function number and raw flag
00460       rwFlag               - 0 read, 1 write
00461       functionNumber       - I/O card area
00462       blockMode            - 1 block mode else bytes
00463       opCode               -  1 incrementing, 0 fixed location
00464       registerAddress_bits_16thru15  - Register address bits 15 and 16
00465
00466  */
00467 typedef struct _FunctionBlock
00468 {
00469     uint8_t rwFlag : 1;
00470     uint8_t functionNumber : 3;
00471     uint8_t blockMode : 1;
00472     uint8_t opCode : 1;
00473     uint8_t registerAddress_bits_16thru15 : 2;
00474
00475 } __attribute__((packed)) FunctionBlock;
00476
00477 /*
00478    Register extended address
00479       address              - Register address bits 0 through 14
00480       byteCount_bit_8       - Byte or block count msb
00481
00482  */
00483 typedef struct _RegisterExtended
00484 {
00485     unsigned short address : 15;
00486     unsigned short byteCount_bit_8 : 1;
00487
00488 } __attribute__((packed)) RegisterExtended;
00489
00490 /*
00491    I/O read/write extended command
00492       commandIndex         - Start, direction and command index
00493       functionBlock        - R/W flag, function number, block and increment
00494       registerExtended     - Register address bits 0 through 14, msb block count
00495       byteCount            - Byte count bits 0 through 7
00496       stuffBits            - MBZ
00497
00498  */
00499 typedef struct _IoRwExtendedCommand_CMD53
00500 {
00501     CommandIndex commandIndex;
00502     FunctionBlock functionBlock;
00503     RegisterExtended registerExtended;
00504     uint8_t byteCount;
00505     Crc crc;
00506
00507 } __attribute__((packed)) IoRwExtendedCommand_CMD53;
```

```
00508
00509 /*
00510    Register Value ( Host order, Big Endian )
00511       wValue 16bit value
00512       bValue 8 bit value
00513
00514 */
00515
00516 #define HIGH_BYTE (0)
00517 #define LOW_BYTE (1)
00518
00519 typedef union _RegisterValue
00520 {
00521     uint16_t wValue;
00522     uint8_t bValue[2];
00523
00524 } __attribute__((packed)) RegisterValue;
00525
00526 /*
00527  ******************************************************************************
00528  *
00529  * CIA CCCR Register field structures
00530  *
00531  ******************************************************************************
00532  */
00533 /*
00534    CCCR/SDIO Revision Register (SDIO_CIA_CCCR_CCCR_SDIO_REVISION)
00535       sdio                          - SDIO revision
00536       cccr                          - CCCR revision
00537
00538  */
00539 typedef struct _CccrSdioRevision
00540 {
00541     uint8_t sdio : 4;
00542     uint8_t cccr : 4;
00543
00544 } __attribute__((packed)) CccrSdioRevision;
00545
00546 /*
00547    SDIO Specification Revision Register (SDIO_CIA_CCCR_SD_SPEC_REVISION)
00548       rfu                           - Reserved for future use
00549       sd                            - Revision
00550
00551  */
00552 typedef struct _SdioSpecRevision
00553 {
00554     uint8_t rfu : 4;
00555     uint8_t sd : 4;
00556
00557 } __attribute__((packed)) SdioSpecRevision;
00558
00559 /*
00560    I/O Enable or Ready Register (SDIO_CIA_CCCR_IO_[ENABLE|READY])
00561       io7                           - Function 7
00562       io6                           - Function 6
00563       io5                           - Function 5
00564       io4                           - Function 4
00565       io3                           - Function 3
00566       io2                           - Function 2
00567       io1                           - Function 1
00568       rfu                           - Reserved for future use
00569
00570  */
00571 typedef struct _IoFunctions
00572 {
00573     uint8_t io7 : 1;
00574     uint8_t io6 : 1;
00575     uint8_t io5 : 1;
00576     uint8_t io4 : 1;
00577     uint8_t io3 : 1;
00578     uint8_t io2 : 1;
00579     uint8_t io1 : 1;
00580     uint8_t rfu : 1;
00581
00582 } __attribute__((packed)) IoFunctions;
00583
00584 /*
00585    Interrupt Enable or Pending Register
00586          (SDIO_CIA_CCCR_INTERRUPT_[ENABLE|PENDING])
00587       int7                          - Interrupt for function 7
00588       int6                          - Interrupt for function 6
00589       int5                          - Interrupt for function 5
00590       int4                          - Interrupt for function 4
00591       int3                          - Interrupt for function 3
00592       int2                          - Interrupt for function 2
00593       int1                          - Interrupt for function 1
00594       ienm                          - Master enable
```

```
00595
00596  */
00597 typedef struct _IntFunctions
00598 {
00599     uint8_t int7 : 1;
00600     uint8_t int6 : 1;
00601     uint8_t int5 : 1;
00602     uint8_t int4 : 1;
00603     uint8_t int3 : 1;
00604     uint8_t int2 : 1;
00605     uint8_t int1 : 1;
00606     uint8_t ienm : 1;
00607
00608 } __attribute__((packed)) IntFunctions;
00609
00610 /*
00611    Bus Interface Control (SDIO_CIA_CCCR_BUS_INTERFACE_CONTROL)
00612       cdDisable                    - Card detection
00613       scsi                         - Support continuous SPI interrupt
00614       ecsi                         - Enable continuous SPI interrupt
00615       width                        - SDIO data bus width
00616
00617  */
00618 typedef struct _BusInterfaceControl
00619 {
00620     uint8_t cdDisable : 1;
00621     uint8_t scsi : 1;
00622     uint8_t ecsi : 1;
00623     uint8_t rfu : 3;
00624     uint8_t width : 2;
00625
00626 } __attribute__((packed)) BusInterfaceControl;
00627
00628 /*
00629    Card Capability (SDIO_CIA_CCCR_CARD_CAPABILITY)
00630       s4bls                        - 4-bit support for low speed cards
00631       lsc                          - Low speed card
00632       e4mi                         - Enable inter-block interrupts 4 bit mode
00633       s4mi                         - Supports inter-block interrupts 4 bit mode
00634       sbs                          - Supports suspend/resume
00635       srw                          - Supports read wait
00636       smb                          - Supports multiblock
00637       sdc                          - Supports direct commands
00638
00639  */
00640 typedef struct _CardCapability
00641 {
00642     uint8_t s4bls : 1;
00643     uint8_t lsc : 1;
00644     uint8_t e4mi : 1;
00645     uint8_t s4mi : 1;
00646     uint8_t sbs : 1;
00647     uint8_t srw : 1;
00648     uint8_t smb : 1;
00649     uint8_t sdc : 1;
00650
00651 } __attribute__((packed)) CardCapability;
00652
00653 /*
00654    Power Control (SDIO_CIA_CCCR_POWER_CONTROL)
00655       empc                         - Enable master power control
00656       smpc                         - Supports master power control
00657
00658  */
00659 typedef struct _PowerControl
00660 {
00661     uint8_t rfu : 6;
00662     uint8_t empc : 1;
00663     uint8_t smpc : 1;
00664
00665 } __attribute__((packed)) PowerControl;
00666
00667 /*
00668    Register settings and transfer byte union
00669       <lower case>                 - Register bits defined as structure
00670       content                      - Transfer byte
00671
00672  */
00673 typedef union _CccrRegister
00674 {
00675     CccrSdioRevision cccrSdioRevision;
00676     SdioSpecRevision sdioSpecRevision;
00677     IoFunctions ioFunctions;
00678     IntFunctions intFunctions;
00679     BusInterfaceControl busInterfaceControl;
00680     CardCapability cardCapability;
00681     PowerControl powerControl;
```

```
00682     uint8_t content;
00683
00684 } __attribute__((packed)) CccrRegister;
00685
00686 /*
00687  ***************************************************************************
00688  *
00689  * Classes
00690  *
00691  ***************************************************************************
00692  */
00693 /*
00694  ***************************************************************************
00695  *
00696  * SDIO Base Class (CMD0)
00697  *
00698  ***************************************************************************
00699  */
00700 class SdioCommand
00701 {
00702    public:
00703     /*** Constructor ***/
00704     SdioCommand(uint8_t commandIndex);
00705
00706     /*** Destructor ***/
00707     virtual ~SdioCommand() { return; };
00708
00709     /*** Methods ***/
00710     /* Reset command */
00711     void reset(void);
00712
00713     /* Prepare command to send */
00714     virtual void prepare(void);
00715
00716     /* Generate CRC 7 */
00717     uint8_t generateCrc7(void);
00718
00719     /* Display using iprintf */
00720     virtual void display(void) { return; };
00721
00722     /*** Accessors ***/
00723     /* Get command data pointer */
00724     puint8_t getPtr(void);
00725
00726     /* Get response size */
00727     uint32_t getSize(void);
00728
00729    protected:
00730     /* None */
00731
00732    private:
00733     /*** Methods ***/
00734     /* None */
00735
00736     /*** Data Members ***/
00737     /* Command index */
00738     uint8_t __command_index;
00739
00740     /* CRC 7 */
00741     uint8_t __crc7;
00742
00743     /* Command data buffer */
00744     uint8_t __commandData[SDIO_COMMAND_SIZE];
00745
00746     /* Command 0 pointer */
00747     GoIdleStateCommand_CMD0 *__commandPtr;
00748 };
00749
00750 /*
00751  ***************************************************************************
00752  *
00753  * SDIO I/O Send Operational Condition (CMD5)
00754  *
00755  ***************************************************************************
00756  */
00757 class SdioCMD5 : public SdioCommand
00758 {
00759    public:
00760     /*** Constructor ***/
00761     SdioCMD5();
00762
00763     /*** Destructor ***/
00764     ~SdioCMD5() { return; };
00765
00766     /*** Methods ***/
00767     /* Prepare command to send */
00768     void prepare(uint32_t ocr);
```

```
00769
00770     /*** Accessors ***/
00771     /* None */
00772
00773    protected:
00774     /* None */
00775
00776    private:
00777     /*** Methods ***/
00778     /* Reset command */
00779     /* None */
00780
00781     /*** Data Members ***/
00782     /* Command 5 pointer */
00783     IoSendOpCondCommand_CMD5 *__commandPtr;
00784 };
00785
00786 /*
00787  ****************************************************************************
00788  *
00789  * SDIO I/O Read/Write Direct (CMD52)
00790  *
00791  ****************************************************************************
00792  */
00793 class SdioCMD52 : public SdioCommand
00794 {
00795    public:
00796     /*** Constructor ***/
00797     SdioCMD52(void);
00798
00799     /*** Destructor ***/
00800     ~SdioCMD52() { return; };
00801
00802     /*** Methods ***/
00803     /* Prepare command to send */
00804     void prepare(int readWriteFlag, uint8_t function, int rawFlag, uint32_t registerAddress, uint8_t
      writeData);
00805
00806     /* Display using iprintf */
00807     void display(void);
00808
00809     /*** Accessors ***/
00810     /* None */
00811
00812    protected:
00813     /* None */
00814
00815    private:
00816     /*** Methods ***/
00817     /* Reset command */
00818     /* None */
00819
00820     /*** Data Members ***/
00821     /* Command 52 pointer */
00822     IoRwDirectCommand_CMD52 *__commandPtr;
00823 };
00824
00825 /*
00826  ****************************************************************************
00827  *
00828  * SDIO I/O Read/Write Extended (CMD53)
00829  *
00830  ****************************************************************************
00831  */
00832 class SdioCMD53 : public SdioCommand
00833 {
00834    public:
00835     /*** Constructor ***/
00836     SdioCMD53(void);
00837
00838     /*** Destructor ***/
00839     ~SdioCMD53() { return; };
00840
00841     /*** Methods ***/
00842     /* Prepare command to send */
00843     void prepare(int readWriteFlag,
00844                  uint8_t function,
00845                  int blockMode,
00846                  BOOL isIncrementingAddress,
00847                  uint32_t registerAddress,
00848                  uint32_t byteCount,
00849                  uint16_t blockSize);
00850
00851     /* Display using iprintf */
00852     void display(void);
00853
00854     /*** Accessors ***/
```

```
00855     /* Get object */
00856     IoRwExtendedCommand_CMD53 &getCommand(void);
00857
00858    protected:
00859     /* None */
00860
00861    private:
00862     /*** Methods ***/
00863     /* Reset command */
00864     /* None */
00865
00866     /*** Data Members ***/
00867     /* Command 53 pointer */
00868     IoRwExtendedCommand_CMD53 *__commandPtr;
00869 };
00870
00871 /*
00872  ******************************************************************************
00873  *
00874  * CRC toggle command (CMD59)
00875  *
00876  ******************************************************************************
00877  */
00878 class SdioCMD59 : public SdioCommand
00879 {
00880    public:
00881     /*** Constructor ***/
00882     SdioCMD59(void);
00883
00884     /*** Destructor ***/
00885     ~SdioCMD59() { return; };
00886
00887     /*** Methods ***/
00888     /* Prepare command to send */
00889     void prepare(BOOL isOn);
00890
00891     /*** Accessors ***/
00892     /* None */
00893
00894    protected:
00895     /* None */
00896
00897    private:
00898     /*** Methods ***/
00899     /* Reset command */
00900     /* None */
00901
00902     /*** Data Members ***/
00903     /* Command 5 pointer */
00904     CrcOnOffCommand_CMD59 *__commandPtr;
00905 };
00906
00907 /*
00908  ******************************************************************************
00909  *
00910  * SDIO Response Base Class
00911  *
00912  ******************************************************************************
00913  */
00914 class SdioResponse
00915 {
00916    public:
00917     /*** Constructor ***/
00918     SdioResponse(SdioBusType busType, SdioResponseType responseType);
00919
00920     /*** Destructor ***/
00921     virtual ~SdioResponse() = 0;
00922
00923     /*** Methods ***/
00924     /* Is response OK */
00925     virtual BOOL isOk(void) = 0;
00926
00927     /* Display using iprintf */
00928     virtual void display(void) = 0;
00929
00930     /*** Accessors ***/
00931     /* Get response type */
00932     SdioResponseType getType(void);
00933
00934     /* Get response type */
00935     SdioBusType getBusType(void);
00936
00937     /* Get remaining response */
00938     virtual puint8_t getPtr(void) = 0;
00939
00940     /* Get response size */
00941     virtual uint32_t getSize(void) = 0;
```

```
00942
00943    protected:
00944     /* None */
00945
00946    private:
00947     /*** Methods ***/
00948     /* Synchronize data */
00949     virtual void synchronize(void) = 0;
00950
00951     /*** Data Members ***/
00952     /* Type */
00953     SdioResponseType __type;
00954
00955     SdioBusType __busType;
00956 };
00957
00958 /*
00959  ****************************************************************************
00960  *
00961  * I/O Send Operational Condition response (R4)
00962  *
00963  ****************************************************************************
00964  */
00965 class SdioResponseR4 : public SdioResponse
00966 {
00967    public:
00968     /*** Constructor ***/
00969     SdioResponseR4(SdioBusType busType);
00970
00971     /*** Destructor ***/
00972     ~SdioResponseR4();
00973
00974     /*** Methods ***/
00975     /* Is response OK */
00976     BOOL isOk(void);
00977
00978     /* Display using iprintf */
00979     void display(void);
00980
00981     /*** Accessors ***/
00982     /* Get response pointer */
00983     IoSendOpCondResponse_R4 *getResponsePtr(void);
00984
00985     /* Get remaining response */
00986     puint8_t getPtr(void);
00987
00988     /* Get response size */
00989     uint32_t getSize(void);
00990
00991     /* Get response pointer */
00992     IoSendOpCondResponse_R4 &getResponse(void);
00993
00994     /* Get response size */
00995     uint32_t getNumberIoFunctions(void);
00996
00997     /* Get Operations Condition Register (OCR) */
00998     uint32_t getOcr(void);
00999
01000    protected:
01001     /* None */
01002
01003    private:
01004     /*** Methods ***/
01005     /* Synchronize data */
01006     void synchronize(void);
01007
01008     /*** Data Members ***/
01009     /* Response pointer and size */
01010     puint8_t __responsePtr;
01011     ssize_t __responseSize;
01012
01013     /* Response */
01014     IoSendOpCondResponse_R4 __response;
01015
01016     /* Modified R1 (SPI Mode) */
01017     IoSendOpCondResponseSpiMode_R4 __responseSpiMode;
01018 };
01019
01020 /*
01021  ****************************************************************************
01022  *
01023  * I/O Read/Write Direct response (R5)
01024  *
01025  ****************************************************************************
01026  */
01027 class SdioResponseR5 : public SdioResponse
01028 {
```

```
01029    public:
01030     /*** Constructor ***/
01031     SdioResponseR5(SdioBusType busType);
01032
01033     /*** Destructor ***/
01034     ~SdioResponseR5();
01035
01036     /*** Methods ***/
01037     /* Is response OK */
01038     BOOL isOk(void);
01039
01040     /* Is card idle */
01041     BOOL isIdle(void);
01042
01043     /* Display using iprintf */
01044     void display(void);
01045
01046     /*** Accessors ***/
01047     /* Get response pointer */
01048     IoRwDirectResponse_R5 *getResponsePtr(void);
01049
01050     /* Get remaining response */
01051     puint8_t getPtr(void);
01052
01053     /* Get response size */
01054     uint32_t getSize(void);
01055
01056     /* Get response reference */
01057     IoRwDirectResponse_R5 &getResponse(void);
01058
01059     /* Get data byte */
01060     uint8_t getData(void);
01061
01062    protected:
01063     /* None */
01064
01065    private:
01066     /*** Methods ***/
01067     /* Synchronize data */
01068     void synchronize(void);
01069
01070     /*** Data Members ***/
01071     /* Response pointer and size */
01072     puint8_t __responsePtr;
01073     ssize_t __responseSize;
01074
01075     /* IO_RW_DIRECT Response */
01076     IoRwDirectResponse_R5 __response;
01077
01078     /* IO_RW_DIRECT Response (SPI Mode) */
01079     IoRwDirectResponseSpiMode_R5 __responseSpiMode;
01080 };
01081
01082 #endif /* _SDIO_H_ */
```

24.471 sdioBsp.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SDIOBSP_H
00006 #define _SDIOBSP_H
00007
00008 #include <basictypes.h>
00009 /*
00010  ******************************************************************************
00011  *
00012  * Debugging
00013  *
00014  ******************************************************************************
00015  */
00016 #define SDIO_DEBUG_INT (1)
00017
00018 /*
00019  ******************************************************************************
00020  *
00021  * C Platform Functions
00022  *
00023  ******************************************************************************
00024  */
00025
00026 /*
00027  ******************************************************************************-
00028  *
```

```
00029  * Set up access
00030  *
00031  *    Parameters:
00032  *       None
00033  *
00034  *    Return:
00035  *       TRUE if OK, FALSE otherwise
00036  *
00037  *    Notes:
00038  *       None
00039  *
00040  ****************************************************************************-
00041  */
00042 BOOL SdioBspSetupHardware(void);
00043
00044 /*
00045  ****************************************************************************-
00046  *
00047  * Assert SDIO chip select/deselect
00048  *
00049  *    Parameters:
00050  *       pin               - Pin number, 0 none
00051  *
00052  *    Return:
00053  *       None
00054  *
00055  *    Notes:
00056  *       Implemented for MOD-DEV-70 and MOD-DEV-100
00057  *
00058  ****************************************************************************-
00059  */
00060 void SdioAssertChipSelect(int pin);
00061 void SdioDeassertChipSelect(int pin);
00062
00063 #endif /* _SDIOBSP_H */
```

24.472 sdioBus.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SDIOBUS_H_
00006 #define _SDIOBUS_H_
00007
00008 #include <basictypes.h>
00009 #include <sys/types.h>
00010 #include <netDevice.h>
00011 #include <sdio.h>
00012 /*
00013  ****************************************************************************
00014  *
00015  * Debugging
00016  *
00017  ****************************************************************************
00018  */
00019 /* Library debugging switch */
00020 #define SDIO_BUS_DEBUG (1)
00021
00022 #ifdef SDIO_BUS_DEBUG
00023 #define SDIO_DEBUG_IPRINTF(...)                    \
00024     {                                              \
00025         iprintf("%s:%d", __FUNCTION__, __LINE__); \
00026         iprintf(__VA_ARGS__);                      \
00027         iprintf("\r\n");                           \
00028     }
00029 #else /* #ifdef SDIO_BUS_DEBUG  */
00030 #define SDIO_DEBUG_IPRINTF(...) ((void)0)
00031 #endif /* #ifdef SDIO_BUS_DEBUG  */
00032
00033 /*
00034  ****************************************************************************
00035  *
00036  * Runtime Library Definitions
00037  *
00038  ****************************************************************************
00039  */
00040 /* Bus type */
00041 #define SDIO_SPI_BUS (1)
00042
00043 /*
00044  ****************************************************************************
00045  *
00046  * Typedefs
00047  *
```

```
00048  ***************************************************************************
00049  */
00050 /*
00051  ***************************************************************************
00052
00053    Controller interrupt service routine (ISR)
00054
00055    Parameters:
00056       None
00057
00058    Return:
00059       None
00060
00061    Notes:
00062       None
00063
00064  ***************************************************************************
00065  */
00066 extern "C" typedef void (*SdioInterruptService)(void);
00067
00068 /*
00069  ***************************************************************************
00070  *
00071  * SDIO Bus Base Class
00072  *
00073  ***************************************************************************
00074  */
00075 class SdioBus
00076 {
00077    public:
00078     /*** Constructor ***/
00079     SdioBus(uint32_t speed, uint32_t connectTimeout, uint32_t responseTimeout);
00080
00081     /*** Destructor ***/
00082     virtual ~SdioBus();
00083
00084     /*** Methods ***/
00085     /* Attach to bus */
00086     virtual BOOL attachBus(void) = 0;
00087
00088     /* Acquire bus */
00089     virtual BOOL acquireBus(void) = 0;
00090
00091     /* Release bus */
00092     virtual void releaseBus(void) = 0;
00093
00094     /* Select card on bus */
00095     virtual void selectCard(void) = 0;
00096
00097     /* Release card */
00098     virtual void releaseCard(void) = 0;
00099
00100    /* Send command */
00101    virtual BOOL sendCommand(SdioCommand &command) = 0;
00102
00103    /* Receive response */
00104    virtual BOOL receiveResponse(SdioResponse &response, BOOL idle = TRUE) = 0;
00105
00106    /* Send data */
00107    virtual BOOL sendData(const puint8_t dataPtr, ssize_t dataLength) = 0;
00108
00109    /* Receive data */
00110    virtual BOOL receiveData(puint8_t dataPtr, ssize_t dataLength) = 0;
00111
00112    /* Idle bus */
00113    virtual void idleBus(void) = 0;
00114
00115    /* Execute extended command (e.g. CMD53) */
00116    virtual BOOL executeExtendedCommand(SdioCommand &command,
00117                                        SdioResponse &response,
00118                                        BOOL writeData,
00119                                        puint8_t dataPtr,
00120                                        ssize_t dataLength) = 0;
00121
00122    /*** Accessors ***/
00123    /* Bus speed in Hz */
00124    uint32_t getSpeed(void);
00125
00126    /* Connect timeout */
00127    uint32_t getConnectTimeout(void);
00128
00129    /* Connect timeout */
00130    uint32_t getResponseTimeout(void);
00131
00132    /* Valid ? */
00133    BOOL isValid(void);
00134
```

```
00135    protected:
00136     /* None */
00137
00138    private:
00139     /*** Methods ***/
00140     /* None */
00141
00142     /*** Data Members ***/
00143     /* Bus speed in Hz */
00144     uint32_t __speed;
00145
00146     /* Connect timeouts in ticks */
00147     uint32_t __connectTimeout;
00148
00149     /* Response timeouts in ticks */
00150     uint32_t __responseTimeout;
00151
00152     /* Valid? */
00153     BOOL __valid;
00154 };
00155
00156 /*
00157  ****************************************************************************
00158  *
00159  * SDIO Bus SPI Mode
00160  *
00161  ****************************************************************************
00162  */
00163 class SdioBusSpiMode : public SdioBus
00164 {
00165    public:
00166     /*** Constructor ***/
00167     SdioBusSpiMode(uint32_t speed,
00168                    uint32_t connectTimeout,
00169                    uint32_t responseTimeout,
00170                    ssize_t idleByteCount,
00171                    uint16_t idleFillValue,
00172                    int chipSelectMask,
00173                    NetDeviceSelectDetail chipSelectDetail);
00174
00175     /*** Destructor ***/
00176     ~SdioBusSpiMode();
00177
00178     /*** Methods ***/
00179     /* Attach to bus */
00180     BOOL attachBus(void);
00181
00182     /* Acquire bus */
00183     BOOL acquireBus(void);
00184
00185     /* Release bus */
00186     void releaseBus(void);
00187
00188     /* Select card on bus */
00189     void selectCard(void);
00190
00191     /* Release card */
00192     void releaseCard(void);
00193
00194     /* Send command */
00195     BOOL sendCommand(SdioCommand &command);
00196
00197     /* Receive response */
00198     BOOL receiveResponse(SdioResponse &response, BOOL idleBus = TRUE);
00199
00200     /* Send data */
00201     BOOL sendData(const puint8_t dataPtr, ssize_t dataLength);
00202
00203     /* Receive data */
00204     BOOL receiveData(puint8_t dataPtr, ssize_t dataLength);
00205
00206     /* Idle bus */
00207     void idleBus(void);
00208
00209     /* Execute extended command (e.g. CMD53) */
00210     BOOL executeExtendedCommand(SdioCommand &command, SdioResponse &response, BOOL writeData, puint8_t
      dataPtr, ssize_t dataLength);
00211
00212    protected:
00213     /* None */
00214
00215    private:
00216     /*** Methods ***/
00217     /* Wait for and the get data received token */
00218     uint8_t waitForDataReceived(void);
00219
00220     /* Waits for data start token */
```

```
00221     BOOL waitForData(void);
00222
00223     /*** Data Members ***/
00224     /* SPI setting handle*/
00225     int __spiSetting;
00226
00227     /* SPI chip select mask */
00228     int __chipSelectMask;
00229
00230     /* SPI chip select detail */
00231     NetDeviceSelectDetail __chipSelectDetail;
00232
00233     /* SPI chip select mask */
00234     int __chipSelectPin;
00235
00236     /* Bytes to idle bus */
00237     ssize_t __idleByteCount;
00238
00239     /* Value used to idle bus */
00240     uint16_t __idleFillValue;
00241 };
00242
00243 #endif /* _SDIOBUS_H_ */
```

24.473 serial.h File Reference

NetBurner Serial API.

```
#include <basictypes.h>
```

Macros

- #define **SERIAL_ERR_NOSUCH_PORT** (-1)

  Port number does not exist.
- #define **SERIAL_ERR_PORT_NOTOPEN** (-2)

  Port is not open.
- #define **SERIAL_ERR_PORT_ALREADYOPEN** (-3)

  Port is already open.
- #define **SERIAL_ERR_PARAM_ERROR** (-4)

  Parameter error.
- #define **ADDR_ESCAPE_CHAR** (0xFF)

  Address escape character.
- #define SimpleOpenSerial(p, b) OpenSerial(p, b, 1, 8, eParityNone)

  Simple open a serial port.

Enumerations

- enum parity_mode {
  eParityNone , eParityOdd , eParityEven , eParityMulti ,
  eParityMultiOdd , eParityMultiEven }

  Serial Parity Modes.

Functions

- int OpenSerial (int portnum, unsigned int baudrate, int stop_bits, int data_bits, parity_mode parity)

  Open a serial port.
- int OpenDefaultSerial ()

  Opens the Default serial port as defined by the Boot Config settings.
- void SerialExpandRxBuffer (int fd, int nb)

  Expand the received serial buffer.
- int SerialClose (int portnum)

  Close a serial port.

- void SerialEnableTxFlow (int port, int enab)

  Enable transmit software flow control on the specified UART.
- void SerialEnableRxFlow (int port, int enab)
- void SerialEnableHwTxFlow (int port, int enab)
- void SerialEnableHwRxFlow (int port, int enab)
- void Serial485HalfDupMode (int port, int enab)
- void SendBreak (int port, uint32_t time)
- int serwriteaddress (int fd, const char c)
- int GetUartErrorReg (int fd)
- void SetRTS (int port, bool val)
- BOOL SerialSendComplete (int fd)

24.473.1 Detailed Description

NetBurner Serial API.

24.474 serial.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00014 #ifndef _NB_SERIAL_H
00015 #define _NB_SERIAL_H
00016
00017 #include <basictypes.h>
00018
00022 #define SERIAL_ERR_NOSUCH_PORT (-1)
00023 #define SERIAL_ERR_PORT_NOTOPEN (-2)
00024 #define SERIAL_ERR_PORT_ALREADYOPEN (-3)
00025 #define SERIAL_ERR_PARAM_ERROR (-4)
00028 // Special UART characters
00029 #define ADDR_ESCAPE_CHAR (0xFF)
00030
00031 /*
00032  * Definitions used to represent bits of the UART error register.  These are
00033  * associated with the GetUartErrorReg() function, which is only applicable to
00034  * the MOD5270, SB70 and SB72.
00035  */
00036 #define UART_ERR_BREAK (0x08)
00037 #define UART_ERR_FRAME (0x04)
00038 #define UART_ERR_PARITY (0x02)
00039 #define UART_ERR_OVERRUN (0x01)
00040
00045 typedef enum
00046 {
00047     eParityNone,
00048     eParityOdd,
00049     eParityEven,
00050     eParityMulti,
00051     eParityMultiOdd,
00052     eParityMultiEven
00053 } parity_mode;
00054
00078 int OpenSerial(int portnum, unsigned int baudrate, int stop_bits, int data_bits, parity_mode parity);
00079
00091 #define SimpleOpenSerial(p, b) OpenSerial(p, b, 1, 8, eParityNone)
00092
00106 int OpenDefaultSerial();
00107
00116 void SerialExpandRxBuffer(int fd, int nb);
00117
00128 int SerialClose(int portnum);
00129
00141 void SerialEnableTxFlow(int port, int enab);
00142
00154 void SerialEnableRxFlow(int port, int enab);
00155
00175 void SerialEnableHwTxFlow(int port, int enab);
00176
00192 void SerialEnableHwRxFlow(int port, int enab);
00193
00209 void Serial485HalfDupMode(int port, int enab);
00210
00227 void SendBreak(int port, uint32_t time);
```

```
00228
00244 int serwriteaddress(int fd, const char c);
00245
00268 int GetUartErrorReg(int fd);
00269
00279 void SetRTS(int port, bool val);
00280
00281 #if (defined CB34EX || defined SB700EX || defined SB72EX || defined SB800EX)
00291 BOOL GetCD(int port);
00292
00302 BOOL GetRI(int port);
00303
00313 BOOL GetDSR(int port);
00314
00324 void SetDTR(int port, BOOL val);
00325 #endif /* CB34EX/SB700EX/SB72EX */
00326
00333 BOOL SerialSendComplete(int fd);
00334
00335 #if (defined SAME70) || (defined __MIMXRT10xx__) || (defined __MIMXRT11xx__)
00336
00337 typedef void (*serTxCompCallback_t)(int portnum, uint32_t bytesSinceLast);
00344 serTxCompCallback_t RegisterTxEmptyCallback(int fd, serTxCompCallback_t pFunc);
00345
00346 #endif
00347
00348 #endif /* _NB_SERIAL_H */
00349
```

24.475 serial_config_extension.h File Reference

Modal extension to serial config server.

24.475.1 Detailed Description

Modal extension to serial config server.

24.476 serial_config_extension.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 typedef void(modal_serial_rx)(int sock);
00010 extern modal_serial_rx  *pSerialConfigModalProcess;
00011
00012
```

24.477 serial_extensions.h

```
00001
00002 #ifndef NB_SERIAL_EXTENSIONS_H
00003 #define NB_SERIAL_EXTENSIONS_H
00004 class fifo_buffer_storage;
00005
00006 typedef void (*ExtendedCommand)(int fd, fifo_buffer_storage &rest_of_command);
00007
00008 const uint32_t CommandFlagHidden=1;   //Dont show in help
00009 const uint32_t CommandFlagCaseInsensitive=2; //Case insensitive command
00010
00011
00012 struct CommandElement
00013 {
00014     const char *CommandText;
00015     const char *CommandHelpText;
00016     ExtendedCommand pCmd;
00017     uint32_t flag;
00018 };
00019
00020 // Pass in an array of commands to add, last element should have NULL CommandText
00021 extern const CommandElement *pPlatformCommands;
00022 extern const CommandElement *pExtendedCommands;
00023
00024 #endif
```

24.478 serinternal.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_SER_INTERNAL
00006 #define _NB_SER_INTERNAL
00007 #include <buffers.h>
00008
00012 #define UART_INIT (0x0001) /* UART is initialized */
00013
00014 #define UART_TX_EMPTY (0x0002) /* UART Tx buffer is empty */
00015
00016 #define UART_SENT_STOP (0x0004) /* UART has sent XOFF */
00017
00018 #define UART_RX_STOP (0x0008) /* UART has received XOFF */
00019
00020 #define UART_FLOW_TX                          \
00021     (0x0010) /* UART must respond to received \
00022                 flow control */
00023
00024 #define UART_FLOW_RX                          \
00025     (0x0020) /* UART is allowed to send flow \
00026                 control */
00027
00028 #define UART_FLOW_NEED_TOSTOP (0x0040) /* UART needs to send XOFF */
00029
00030 #define UART_FLOW_NEED_TOSTART (0x0080) /* UART needs to send XON */
00031
00032 #define UART_RS485_TX_MODE                    \
00033     (0x0100) /* UART is in RS-485 HD/FD mode \
00034                 (not used) */
00035
00036 #define UART_MULTI_MODE (0x0200) /* UART is in multi-drop mode */
00037
00038 #define UART_FLOW_TXRTSCTS                    \
00039     (0x0400) /* UART has TxCTS hardware flow \
00040                 control enabled */
00041
00042 #define UART_FLOW_TX485FD                     \
00043     (0x0800) /* UART is in RS-485 full-duplex \
00044                 mode */
00045
00046 #define UART_FLOW_TX485HD                     \
00047     (0x1000) /* UART is in RS-485 half-duplex \
00048                 mode */
00049
00050 #define UART_FLOW_RXRTSCTS                    \
00051     (0x2000) /* UART has RxRTS hardware flow \
00052                 control enabled */
00053
00054 #define UART_RS422_MODE (0x4000) /* UART is in RS-422 mode */
00055
00056 #define UART_TX_LAST_BIT                      \
00057     (0x8000) /* UART has transmitted the last \
00058                 stop bit of the last byte */
00059
00063 #define XON (0x11)
00064 #define XOFF (0x13)
00065
00071 #define UART_XOFF_LIMIT (100)
00072 #define UART_XON_LIMIT (200)
00073
00074 typedef int(GetNextCharFunc)(int uartnum);
00075 typedef void(PutNextCharFunc)(int uartnum, uint8_t c);
00076
00077 int BaseGetChar(int num);
00078 void BasePutChar(int num, uint8_t c);
00079
00080 enum dataBits_t
00081 {
00082     DATA_BITS_5 = 0,
00083     DATA_BITS_6 = 1,
00084     DATA_BITS_7 = 2,
00085     DATA_BITS_8 = 3,
00086     DATA_BITS_9 = 4,
00087     DATA_BITS_10 = 5,
00088     DATA_BITS_11 = 6,
00089     DATA_BITS_12 = 7,
00090 };
00091
00092 constexpr inline uint8_t GetBitCount(dataBits_t bits)
00093 {
00094     return bits + 5;
00095 }
00096
```

```
00097 #ifdef SERIAL_USE_RING_BUFFER
00098 class serRingBuf
00099 {
00100     uint8_t *buf;
00101     uint16_t siz;
00102     uint16_t used;
00103     uint16_t wr;
00104     uint16_t rd;
00105     static int idx;
00106
00107     int put(uint8_t d) volatile;
00108     int get(uint8_t *d) volatile;
00109
00110   public:
00111     void Init(uint8_t max_buffers, uint8_t fromisr = 1);
00112     void Reset(uint8_t max_buffers);
00113     int WriteData(uint8_t *d, int len) volatile;
00114     int ReadData(uint8_t *d, int len) volatile;
00115     uint8_t PeekData();
00116     bool Full();
00117     bool Empty();
00118
00119     uint16_t SpaceAvail();
00120     inline void SetMaxBuffers(uint8_t max_buffers) { return; }
00121 };
00122
00123 typedef serRingBuf serialBuf_t;
00124 #else
00125 typedef fifo_buffer_storage serialBuf_t;
00126 #endif
00127
00128 struct UartDataRec
00129 {
00130     serialBuf_t m_FifoRead;
00131     serialBuf_t m_FifoWrite;
00132     GetNextCharFunc *m_pGetCharFunc;
00133     PutNextCharFunc *m_pPutCharFunc;
00134     int m_UartState;
00135     uint8_t m_Errors;
00136     dataBits_t m_dataBits : 3;
00137 };
00138
00139 extern UartDataRec UartData[];
00140
00141 void WakeTx(int x);
00142
00143 #endif /* _NB_SER_INTERNAL */
```

24.479 servlets.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_SERVLET_H
00006 #define _NB_SERVLET_H
00007
00008 #include <iosys.h>
00009 #include <nbrtos.h>
00010 #include <webclient/http_funcs.h>
00011 #include <iosys.h>
00012 #include <nbstring.h>
00013 #include <dns.h>
00014 #include <nettimer.h>
00015
00016
00017
00018
00019 class servlet_list;
00020
00021 class servlet
00022 {
00023 protected:
00024     servlet *pNext;
00025     servlet_list *pOwner;
00026
00027   public:
00028     servlet(servlet_list *whereToAdd);
00029     servlet();
00030     ~servlet();
00031
00032     // Also returns the number of seconds needed for timeout/recycle...
00033     virtual int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set) = 0;
00034     virtual void ProcessSelectResult(fd_set &rd_set, fd_set &wr_set, fd_set &er_set) = 0;
00035     friend class servlet_list;
```

```
00036 };
00037
00038 class servlet_list
00039 {
00040     servlet *pHead;
00041     OS_CRIT ListCritical;
00042     OS_SEM m_Sem;
00043
00044   public:
00045     servlet_list();
00046     ~servlet_list();
00047     void add(servlet *ps);
00048     void remove(servlet *ps);
00049     void run_once_through_select_loop(int max_timeout_ticks);
00050 };
00051
00052
00053 typedef enum {
00054      eReadingHeader,
00055      eBody
00056 }eWebResponse_t;
00057
00058 typedef enum {
00059     eFirstValidInterface,
00060     eTransactionComplete,
00061     eTransactionDnsFail,
00062     eTransactionFail,
00063     eTransactionTimeOut,
00064     eReconnectTime,
00065     eReconnectNow,
00066 } eWebClientAction_t;
00067
00068 typedef enum {
00069     eIdle,
00070     eWaitingIntf,
00071     eDoingDns,
00072     eSendingRequest,
00073     eSendingPayload,
00074     eWaitingForResponse,
00075     eSleeping,
00076     eDisconnected,
00077 } eWebClientState_t;
00078
00079 const int HEADER_BUFFER_SIZE=256;
00080
00081 class WebClientServlet: public servlet, public TimeOutElement
00082 {
00083 protected:
00084 //Servlet virtual functions
00085  virtual int AddToSelectSet(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00086  virtual void ProcessSelectResult(fd_set &rd_set, fd_set &wr_set, fd_set &er_set);
00087 //Timeout element virtual function
00088  virtual void TimeElementEvent();
00089
00090  int m_fd;          //fd to hold DNS Requests and tcp send/rx
00091  eWebClientState_t  m_cur_state; //State variable
00092  eWebResponse_t     m_resp_state;
00093  ParsedURI m_cur_request; //Current URI
00094  buffer_object * pDestinationBuffer;
00095
00096  NBString PostPayload;
00097  NBString AdditionalHeaders;
00098  NBString PostType;
00099
00100
00101  char HeaderBuffer[HEADER_BUFFER_SIZE];
00102  int m_HeaderPos;
00103  int m_rem_content;
00104
00105  uint32_t m_ResultCode;
00106
00107  TickTimeout m_waketime;
00108  TickTimeout m_responsetimeout;
00109
00110
00111
00112  servlet_list * m_pWhoToReJoin;
00113
00114
00115 //Fucntion to handle data to read
00116 //returns true if the function is complete.
00117 virtual bool ResponseReader(int fd);
00118
00119 virtual void StartQuery();
00120
00121
00122 //Called when things happen
```

```
00123 virtual void ActionComplete(eWebClientAction_t action);
00124
00125 protected:
00126     const char* ActionText(eWebClientAction_t action);
00127
00128
00129
00130 void DisconnectTil(TickTimeout & tt); //Maximum interval ~ 3 yrs with default ticks.
00131 void ReConnectNow();
00132
00133 void CoreStart(const char* pUrl,buffer_object * result_buffer, uint32_t timeout);
00134
00135 public:
00136
00137 // If List_To_Join is null joins config server
00138 WebClientServlet(servlet_list * List_To_Join=0);
00139
00140 uint32_t GetResultCode(){return m_ResultCode; };
00141
00142 void    NewGet(const char* pUrl,buffer_object * result_buffer, uint32_t timeout=0,const char*
      additional_header=0);
00143 void    NewPost(const char * pUrl,const char* pType, NBString & payload,buffer_object * result_buffer,
      uint32_t timeout=0,const char* additional_header=0);
00144 void    NewPost(const char * pUrl,const char* pType, const char* payload,buffer_object *
      result_buffer, uint32_t timeout=0,const char* additional_header=0);
00145 };
00146
00147
00148
00149 #endif
```

24.480 sha1.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _SHA1_H_
00006 #define _SHA1_H_
00007
00008 // NB Definitions
00009 #include <predef.h>
00010
00011 // NB Libs
00012 #include <hash.h>
00013
00014 struct SHA1_CTX : public HASH_CTX
00015 {
00016     unsigned long state[5];
00017     unsigned long count[2];
00018     unsigned char buffer[64];
00019 #ifdef SSL_TLS_SUPPORT
00020     unsigned char hmac_inner_pad[64];
00021     unsigned char hmac_outer_pad[64];
00022 #endif
00023     void __Init();
00024     void __Update(const unsigned char *data, unsigned int len);
00025     void __Final(unsigned char *digest);
00026     int __GetDigestLen() const;
00027     int __GetOIDLen() const;
00028     const unsigned char *__GetOID() const;
00029     void ctor();
00030
00031    private:
00032     static __vtable_HASH_CTX_t _s__vtable;
00033 };
00034
00035 typedef unsigned char sha1_digest_t[20];
00036
00037 inline void SHA1Init(SHA1_CTX *context)
00038 {
00039     context->ctor();
00040 }
00041 inline void SHA1Update(SHA1_CTX *context, const unsigned char *data, unsigned int len)
00042 {
00043     context->Update(data, len);
00044 }
00045 inline void SHA1Final(unsigned char digest[20], SHA1_CTX *context)
00046 {
00047     context->Final(digest);
00048 }
00049 #endif /* #ifdef _SHA1_H_ */
```

24.481 ShutDownNotifications.h File Reference

NetBurner Shutdown Notification Functions.

Macros

- #define **SHUTDOWN_CODEUPDATE** (1)

  A code update is requested.
- #define **SHUTDOWN_CONFIGURE_REBOOT** (2)

  Configuration values have been modified with a requested reboot.

Functions

- bool NBApproveShutdown (int reason)

  Approve action that will result in a reboot.
- void NBFaultNotify ()

  Emergency notification of a fault shutdown.

24.481.1 Detailed Description

NetBurner Shutdown Notification Functions.

24.482 ShutDownNotifications.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00042 #ifndef SHUTDOWN_NOTIFY_H
00043 #define SHUTDOWN_NOTIFY_H (1)
00044
00050 #define SHUTDOWN_CODEUPDATE (1)
00051 #define SHUTDOWN_CONFIGURE_REBOOT (2)
00071 bool NBApproveShutdown(int reason);
00072
00097 void NBFaultNotify();
00098
00099 #endif
00100
```

24.483 smarttrap.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef NB_SMARTTRAP_H
00005 #define NB_SMARTTRAP_H
00006 /* Call this function to enable RTOS aware smart trap reports. */
00007 void EnableSmartTraps();
00008 #endif
```

24.484 snmp.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_SNMP_H_
00006 #define _NB_SNMP_H_
00007
00008 /*
00009   This file provides null functions for the SNMP counters and functions
00010   to allow the system to compile without SNMP.
00011 */
00012
00013 #define SNMPCOUNT(x) ((void)0)
```

```
00014 #define SNMPDEC(x) ((void)0)
00015 #define SNMPADD(x, y) ((void)0)
00016
00017 #endif /* #ifndef _NB_SNMP_H_ */
```

24.485 Socks.h File Reference

NetBurner SOCKS5 API.

```
#include <predef.h>
#include <tcp.h>
#include <nbstring.h>
```

Macros

- #define **SOCKS_SUCCESS** (0)

  The connection was successful.
- #define **SOCKS_ERR_TIMEOUT** (-500)

  The connection timed out.
- #define **SOCKS_BAD_ADR_TYPE** (-501)

  Address type specified does not match the address provided.
- #define **SOCKS_CONN_ABORTED** (-502)

  The other side aborted the connection.
- #define **SOCKS_BAD_AUTH_TYPE** (-503)

  The server doesn't support the requested authorization types.
- #define **SOCKS_BAD_UN_PW** (-504)

  The server did not approve the username and password provided.
- #define **SOCKS_UN_TOO_LONG** (-505)

  The specified username is too long.
- #define **SOCKS_PW_TOO_LONG** (-506)

  The specified password is too long.
- #define **SOCKS_BAD_REPLY** (-507)

  The server reply was not successful.
- #define **SOCKS_BAD_DOMAIN** (-508)

  Unable to resolve domain name passed.
- #define **SOCKS_CMD_NOT_SUPPORTED** (-509)

  The requested command aren't supported.
- #define **SOCKS_BAD_PARAM** (-510)

  The passed in parameters are not valid.
- #define **SOCKS_MAX_UNAME_SIZE** 255

  Max character length for usernames. Defined in RFC 1929.
- #define **SOCKS_MAX_PASSWD_SIZE** 255

  Max character length for passwords. Defined in RFC 1929.

Enumerations

- enum SocksAuthType : unsigned char { eSocksAuthTypeGssApi = 0x01 , eSocksAuthTypeUnPw = 0x02 , eSocksAuthTypeNoAuth = 0x04 }

  SOCKS Autherization Types.
- enum SocksClientCmd : unsigned char { eSocksClientCmdConnect = 1 , eSocksClientCmdBind = 2 , eSocksClientCmdUdpAssoc = 3 }

  SOCKS Client Commands.
- enum SocksAdrType : unsigned char { eSocksAdrTypeNone = 0x00 , eSocksAdrTypeIpv4 = 0x01 , eSocksAdrTypeDomain = 0x03 , eSocksAdrTypeIpv6 = 0x04 }

  SOCKS Address Types.

Functions

- bool AuthWithGssApi ()

  A weak function that should be overriden by the developer in order to support GSSAPI authorization.

- void SetSocksProxySettings (SocksProxy *socksProxy)

  Set the system level SOCKS proxy settings object to the one that is passed in. If this object is set and is marked as enabled, calls to CoreConnect() (made from connect(), DoGet(), etc), will all initially connect to the proxy server using the SOCKS5 protocol.

- SocksProxy * GetSocksProxySettings ()

  Get a pointer to the currnetly set Socks proxy settings object.

24.485.1 Detailed Description

NetBurner SOCKS5 API.

24.486 Socks.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 #ifndef _NB_SOCKS_H_
00016 #define _NB_SOCKS_H_
00017
00018 #include <predef.h>
00019 #include <tcp.h>
00020 #include <nbstring.h>
00021
00022 struct socket_struct;
00023 typedef socket_struct SOCKET;
00024 typedef SOCKET *PSOCKET;
00025
00031 #define SOCKS_SUCCESS               (0)
00032 #define SOCKS_ERR_TIMEOUT           (-500)
00033 #define SOCKS_BAD_ADR_TYPE          (-501)
00034 #define SOCKS_CONN_ABORTED          (-502)
00035 #define SOCKS_BAD_AUTH_TYPE         (-503)
00036 #define SOCKS_BAD_UN_PW             (-504)
00037 #define SOCKS_UN_TOO_LONG           (-505)
00038 #define SOCKS_PW_TOO_LONG           (-506)
00039 #define SOCKS_BAD_REPLY             (-507)
00040 #define SOCKS_BAD_DOMAIN            (-508)
00041 #define SOCKS_CMD_NOT_SUPPORTED     (-509)
00042 #define SOCKS_BAD_PARAM             (-510)
00045 #define SOCKS_TIMEOUT (10 * TICKS_PER_SECOND)
00046 #define SOCKS_MAX_UNAME_SIZE 255
00047 #define SOCKS_MAX_PASSWD_SIZE 255
00048
00053 enum SocksAuthType : unsigned char
00054 {
00055     eSocksAuthTypeGssApi = 0x01,
00056     eSocksAuthTypeUnPw   = 0x02,
00057     eSocksAuthTypeNoAuth = 0x04,
00058 };
00059
00060
00064 enum SocksClientCmd : unsigned char
00065 {
00066     eSocksClientCmdConnect   = 1,
00067     eSocksClientCmdBind      = 2,
00068     eSocksClientCmdUdpAssoc  = 3
00069 };
00070
00074 enum SocksAdrType : unsigned char
00075 {
00076     eSocksAdrTypeNone    = 0x00,
00077     eSocksAdrTypeIpv4    = 0x01,
00078     eSocksAdrTypeDomain  = 0x03,
00079     eSocksAdrTypeIpv6    = 0x04
00080 };
00081
00082
00083 // Currently not used in code, but defined for future use if needed. Correspond to values returned in
00084 // SOCKS responses per RFC 1928.
00085 enum SocksReply : unsigned char
00086 {
```

```
00087     eSocksReplySuc          = 0,
00088     eSocksReplyGeneral      = 1,
00089     eSocksReplyRuleset      = 2,
00090     eSocksReplyNetwork      = 3,
00091     eSocksReplyHost         = 4,
00092     eSocksReplyConRef       = 5,
00093     eSocksReplyTtlExp       = 6,
00094     eSocksReplyCmdNotSup    = 7,
00095     eSocksReplyAdrNotSup    = 8,
00096 };
00097
00098 class SocksProxy
00099 {
00100     public:
00101         SocksProxy(){};
00102         SocksProxy( const NBString& adr, uint16_t port, SocksAdrType adrType, SocksAuthType authType,
      SocksClientCmd cmd, const char* username, const char* password, bool enable );
00103         SocksProxy( const NBString& adr, uint16_t port, SocksAdrType adrType, SocksAuthType authType,
      SocksClientCmd cmd, bool enable );
00104
00105         void SetUsername( const NBString& username ){ m_username = username; }
00106         void SetPassword( const NBString& password ){ m_password = password; }
00107         void SetPort( uint16_t port ){ m_port = port; }
00108         void SetAddressType( SocksAdrType adrType ){ m_adrType = adrType; }
00109         void SetAddress( const NBString& adr ){ m_adr = adr; }
00110         void SetAuthType( SocksAuthType authType ){ m_authType = authType; }
00111         void SetClientCmd( SocksClientCmd cmd ){ m_cmd = cmd; }
00112         void SetEnabled(bool enable){ m_enable = enable; }
00113
00114         NBString& GetUsername(){ return m_username; }
00115         NBString& GetPassword(){ return m_password; }
00116         uint16_t GetPort(){ return m_port; }
00117         SocksAdrType GetAddressType(){ return m_adrType; }
00118         NBString& GetAddress(){ return m_adr; }
00119         SocksAuthType GetAuthType(){ return m_authType; }
00120         SocksClientCmd GetClientCmd(){ return m_cmd; }
00121         bool IsEnabled(){ return m_enable; }
00122
00126         void ClearProxyValues();
00127
00139         int SocksConnect( PSOCKET socket, uint16_t localPort, uint32_t timeout, const IPADDR &ifip,
      int ifn );
00140
00150         int SendAuthAndProcessReply( int tcpFd, const IPADDR& adr, int port );
00151
00152     private:
00153
00160         int SocksProcessReply( int tcpFd );
00161
00168         int SendAuthMethod( int tcpFd );
00169
00176         int AuthWithUsrNmPw( int tcpFd );
00177
00178         NBString m_adr;
00179         NBString m_username;
00180         NBString m_password;
00181         uint16_t m_port             = 1080;
00182         SocksAdrType m_adrType      = eSocksAdrTypeNone;
00183         SocksAuthType m_authType    = eSocksAuthTypeNoAuth;
00184         SocksClientCmd m_cmd        = eSocksClientCmdConnect;
00185         bool m_enable               = false;
00186 };
00187
00188 extern SocksProxy* gSocksProxy;
00189
00190 /*
00191  * @brief Try to initiate a SOCKS5 connection with an IP address.
00192  *
00193  * @param auth The authorization type to use. No Authorization and plain text Username and Password
      are supported. The use of GSSAPI
00194  * needs to be implemented by the developer by overriding AuthWithGssApi().
00195  * @param cmd Which command to isssue. Currently only the Connect command is supported. See RFC 1928
      for details on the other commands.
00196  * @param adrType Which address type to use. This function supports IPv4 and IPv6.
00197  * @param adr The host address to connect to.
00198  * @param timeout The timeout value for the connection.
00199  * @param username The username for the connection, if using the username and password authorization
      type.
00200  * @param password The password for the connection, if using the username and password authorization
      type.
00201  * @param socksPort The port to connect to. The default SOCKS port is 1080.
00202  *
00203  * @return int The FD of the connection if successful, or a TCP or SOCKS error code if unsuccessful.
00204  */
00205 //int SocksConnect( PSOCKET socket, IPADDR adr, uint32_t timeout );
00206
00207 /*
```

```
00208  * @brief Try to initiate a SOCKS5 connection with a domain name or IP address.
00209  *
00210  * @param auth The authorization type to use. No Authorization and plain text Username and Password
       are supported. The use of GSSAPI
00211  * needs to be implemented by the developer by overriding AuthWithGssApi().
00212  * @param cmd Which command to isssue. Currently only the Connect command is supported. See RFC 1928
       for details on the other commands.
00213  * @param adrType Which address type to use. This function supports IPv4, IPv6, and domain names.
00214  * @param adr The host address or domain name to connect to.
00215  * @param timeout The timeout value for the connection.
00216  * @param username The username for the connection, if using the username and password authorization
       type.
00217  * @param password The password for the connection, if using the username and password authorization
       type.
00218  * @param socksPort The port to connect to. The default SOCKS port is 1080.
00219  *
00220  * @return int The FD of the connection if successful, or a TCP or SOCKS error code if unsuccessful.
00221  */
00222 //int SocksConnect( PSOCKET socket, SocksAuthType auth, SocksClientCmd cmd, SocksAdrType adrType,
       const NBString& adr, uint32_t timeout, const NBString& username, const NBString& password, uint16_t
       socksPort = 1080  );
00223
00224
00231 bool AuthWithGssApi();
00232
00239 void SetSocksProxySettings( SocksProxy* socksProxy );
00240
00246 SocksProxy* GetSocksProxySettings();
00247
00248 #endif /* _NB_SOCKS_H_ */
00249
```

24.487 stackFns.h

```
00001 #ifndef __STACKFNS_H
00002 #define __STACKFNS_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006 #include <basictypes.h>
00007
00008 template<typename rT, typename... Ts>
00009 using ArbFn_t = rT (*)(Ts...);
00010
00011 template<typename rT, typename... Ts>
00012 rT RunFnOnStack(ArbFn_t<rT, Ts...> fn, uint32_t *src, uint32_t *end, Ts... args)
00013 {
00014     uint32_t StackFunc[(((uint32_t)end) - ((uint32_t)src)) / 4 + 1];
00015     ArbFn_t<rT, Ts...> pFn = (ArbFn_t<rT, Ts...>)(((uint32_t)StackFunc) | 1);
00016     uint32_t *dst = StackFunc;
00017     while (src < end)
00018     {
00019         *dst++ = *src++;
00020     }
00021     asm volatile("dsb");
00022     return pFn(args...);
00023 }
00024 #endif /* ----- #ifndef __STACKFNS_H  ----- */
```

24.488 startnet.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NB_START_NET_H
00006 #define NB_START_NET_H
00007
00008 /* All of the definitions necessary to start the stack and HTTP */
00009 // NB Constants
00010 #include <constants.h>
00011 // NB Libs
00012 #include <basictypes.h>
00013 #include <http.h>
00014 #include <iosys.h>
00015 #include <ip.h>
00016 #include <nbrtos.h>
00017 #include <nbrtoscpu.h>
00018 #include <nettypes.h>
00019 #include <system.h>
00020 #include <utils.h>
00021 #endif
```

24.489 stopwatch.h File Reference

NetBurner Stopwatch Timer API.
`#include <constants.h>`

Classes

- class StopWatch

  Stopwatch for timing events.

24.489.1 Detailed Description

NetBurner Stopwatch Timer API.

24.490 stopwatch.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00035 #ifndef _NB_STOPWATCH_H
00036 #define _NB_STOPWATCH_H
00037
00038 #include <constants.h>
00039
00045 class StopWatch
00046 {
00047     unsigned long long ref_start;
00048     unsigned long long elapsed;
00049     bool bRunning;
00050
00051   public:
00061     StopWatch(int timer_number = FIRST_UNUSED_TIMER);
00062
00068     void Start();
00069
00075     void Clear();
00076
00082     unsigned long long Stop();
00083
00089     unsigned long long GetTime();
00090
00096     double CountResolution();
00097
00103     double Convert(unsigned long long);
00104 };
00105
00106 #endif
00107
```

24.491 StreamUpdate.h File Reference

Read an Application File from an Input Stream.
`#include <basictypes.h>`

Macros

- #define **STREAM_UP_FAIL** (0)

  Action failed.
- #define **STREAM_UP_OK** (1)

  Action succeeded.

Functions

- int ReadBinaryApplicationCodeFromStream (int fd)

  Read a binary application image from an input stream and program flash memory.
- int ReadS19ApplicationCodeFromStream (int fd)

  Read an ASCII application image in s-record .s19 format from an input stream and program flash memory.

24.491.1 Detailed Description

Read an Application File from an Input Stream.
Functions to read an application form a source with a file descriptor. Useful for programming/updating an application image from sources such as flash memory cards, posts from other devices, and files received from FTP.

24.492 StreamUpdate.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00015 /*
00016  * The code module also contains functions to read and write the User Parameter
00017  * section of onboard flash memory as a stream. They are here for compatibility
00018  * with proir tools releases, but the recommended 3.x tools method is to use
00019  * the application storage of the Configuration System, since it can be accessed
00020  * through the Configuration Server and JSON objects.
00021  */
00022
00033 #ifndef _NB_STREAM_UP_H
00034 #define _NB_STREAM_UP_H
00035
00036 #include <basictypes.h>
00037
00041 #define STREAM_UP_FAIL (0)
00042 #define STREAM_UP_OK (1)
00063 int ReadBinaryApplicationCodeFromStream(int fd);
00064
00084 int ReadS19ApplicationCodeFromStream(int fd);
00085
00086 // Send an application image as to an output stream in s-record S19 format
00087 int SendApplicationCodeAsS19(int fd);
00088
00089 // Exposed to allow custom updates to be used
00090 int ProcessS3(const char *cp, uint32_t base_Addr, puint8_t CopyTo, uint32_t &cur_pos, uint32_t
      maxlen);
00091
00092 //================================== USER FLASH FUNCTIONS ========================================
00093
00094 /*
00095  * Send User Parameter Flash data as a binary output stream to the specified file
00096  * descriptor.
00097  *
00098  * Parameters:
00099  *   int fd     The socket file descriptor to send the data to
00100  *
00101  * Returns:
00102  *   STREAM_UP_OK   The system was able to send the data
00103  *   STREAM_UP_FAIL The system failed to send the data
00104  *
00105  * See Also:
00106  *   int SendUserFlashToStreamAsS19(int fd);
00107  *   int ReadBinaryUserFlashFromStream(int fd);
00108  *   int ReadS19UserFlashFromStream(int fd);
00109  */
00110 int SendUserFlashToStreamAsBinary(int fd);
00111
00112 /*
00113  * Send User Parameter Flash data as an ASCII s-record S19 output stream to the
00114  * specified file descriptor.
00115  *
00116  * Parameters:
00117  *   int fd  The socket file descriptor to send the data to
00118  *
00119  * Returns:
00120  *    STREAM_UP_OK      The system was able to send the data
00121  *    STREAM_UP_FAIL    The system failed to send the data
00122  *
00123  * See Also:
```

```
00124  *   int SendUserFlashToStreamAsBinary(int fd);
00125  *   int ReadBinaryUserFlashFromStream(int fd);
00126  *   int ReadS19UserFlashFromStream(int fd);
00127  */
00128 int SendUserFlashToStreamAsS19(int fd);
00129
00130 /*
00131  * Read data from an ASCII s-record S19 input stream and store it in
00132  * User Parameter Flash.
00133  *
00134  * Parameters:
00135  *   int fd The socket file descriptor to read data from
00136  *
00137  * Returns:
00138  *  STREAM_UP_OK    The system was able to read the data and update flash
00139  *  STREAM_UP_FAIL  The system failed to read or update
00140  *
00141  * See Also:
00142  *   int SendUserFlashToStreamAsS19(int fd);
00143  *   int SendUserFlashToStreamAsBinary(int fd);
00144  *   int ReadBinaryUserFlashFromStream(int fd);
00145  */
00146 int ReadS19UserFlashFromStream(int fd);
00147
00148 /*
00149  * Read data from an binary input stream and store it in User Parameter Flash.
00150  *
00151  * Parameters:
00152  *  int fd  The socket file descriptor to read data from
00153  *
00154  * Returns:
00155  *   STREAM_UP_OK   The system was able to read the data and update flash
00156  *   STREAM_UP_FAIL The system failed to read or update
00157  *
00158  * See Also:
00159  *   int SendUserFlashToStreamAsS19(int fd);
00160  *   int SendUserFlashToStreamAsBinary(int fd);
00161  *   int ReadS19UserFlashFromStream(int fd);
00162  */
00163 int ReadBinaryUserFlashFromStream(int fd);
00164
00165 //================================== UTILITY STRUCURES AND FUNCTIONS
      ========================================
00166
00167 // Structure to hold the data from a update before actually programming it in flash
00168 struct TwoPartUpdateStruct
00169 {
00170     uint8_t S0Record[24];
00171     puint8_t pBlob;
00172     uint32_t address;
00173     uint32_t pgm_size;
00174     int Result;
00175
00176     TwoPartUpdateStruct()
00177     {
00178         pBlob = 0;
00179         Result = STREAM_UP_FAIL;
00180     };
00181 };
00182
00183 /*
00184  * Populates the structure with an allocated binary buffer to program into flash.
00185  * Returns STREAM_UP_OK, and sets structure result to STREAM_UP_OK if successful.
00186  */
00187 int ReadTwoPartAppUdate(int fd, TwoPartUpdateStruct &us);
00188
00189 /*
00190  * Take a successfully read stream update and programs the flash memory.
00191  * Returns STREAM_UP_FAIL if update fails.
00192  * Forces a device reboot on success.
00193  * Frees any dynamically allocated memory.
00194  */
00195 int DoTwoPartAppUpdate(TwoPartUpdateStruct &us);
00196
00197 /*
00198  * Called to abort an update and free any dynamically allocated memory
00199  */
00200 void AbortTwoPartAppUpdate(TwoPartUpdateStruct &us);
00201
00202 #endif
00203
```

24.493 syslog.h

```
00001 /*NB_REVISION*/
```

```
00002
00003 /*NB_COPYRIGHT*/
00004
00005 /*-------------------------------------------------------------------
00006  * The Syslog utility enables you to send logging information
00007  * to a destination host computer using UDP.
00008  * ------------------------------------------------------------------*/
00009 #ifndef SYSLOG_H
00010 #define SYSLOG_H
00011
00012 #include <nettypes.h>
00013
00014 /* The application assigns this address to the destination host where
00015  * the syslog information should be sent. If not address is specified,
00016  * the syslog information is sent to the UDP broadcast address
00017  * of 255.255.255.255
00018  */
00019 extern IPADDR SysLogAddress;
00020
00021 /* Call this function when you want to send data to the syslog host.
00022  * The format is the same as printf(). Note that since it does
00023  * floating point, it will increase the application size.
00024  */
00025 int SysLog(const char *format, ...);
00026
00027 /* Call this function when you want to send data to the syslog host
00028  * via a specific interface. Primarily useful when logging in network drivers.
00029  * The format is the same as printf(). Note that since it does
00030  * floating point, it will increase the application size.
00031  */
00032 int SysLogVia(int interfaceNum, const char *format, ...);
00033
00034 #endif   // SYSLOG_H
```

24.494 system.h File Reference

NetBurner System Functions.

`#include <nettypes.h>`

Functions

- int SaveUserParameters (void *pCopyFrom, int len)

  Save data to the on-chip flash memory User Parameter area.
- void * GetUserParameters (void)

  Returns a void pointer to the user parameter area.
- const char * GetReleaseTag ()

  Returns the NNDK release tag information.

24.494.1 Detailed Description

NetBurner System Functions.

24.495 system.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00014 // The system maintains a configuration record for initialization
00015
00016 #ifndef _SYSTEM_STORAGE_H
00017 #define _SYSTEM_STORAGE_H
00018 #include <nettypes.h>
00019
00020 // Exceptions
00021 #define EXCPT_REBOOT (0)        /* Reboot after an exception */
00022 #define EXCPT_HALT (1)          /* Halt after an exception */
00023 #define EXCPT_QUIET (2)         /* Reboot quietly after an exception */
00024 #define EXCPT_RESET (3)         /* Reboot quietly after an exception */
00025 #define EXCPT_REBOOT_TO_MON (4) /* Reboot to (Alt)monitor after an exception */
```

```
00026
00027 // MAC-48 Media Access Control address in bytes (nettypes.h)
00028 #ifndef MACADDRESS_OCTETS_48
00029 #define MACADDRESS_OCTETS_48 (6)
00030 #endif
00031
00032 /*
00033    IEEE 802.11 identifier and key parameters
00034
00035       m_FileName member is used to save service set identity (SSID) and
00036       either Wired Equivalent Privacy, Wi-Fi Protected Access (WPA) or WPA2
00037       Pre Shared Key (PSK) on platforms that support the interface.
00038
00039       SSID is restricted to printable ASCII characters, 1 to 32 in length.
00040
00041       WEP Key is 10 or 26 hexadecimal ASCII characters.
00042       WPA(2) PSK is printable ASCII 8 to 63 characters in length.
00043
00044       The total length of SSID and the key must not exceed 68 characters.
00045
00046       It is stored as a concatenated null terminate string for example:
00047          "SSID:NetBurner,WPA:XJrLy4GbZe1/3wORx5ja4bBC74284OqbhWwVa3+-"
00048
00049 */
00050 #define SYSTEM_CONFIG_RECORD_SSID_SIZE_MIN (1)
00051 #define SYSTEM_CONFIG_RECORD_SSID_SIZE_MAX (32)
00052 #define SYSTEM_CONFIG_RECORD_SSID_TAG ("SSID:")
00053 #define SYSTEM_CONFIG_RECORD_SSID_TAG_LENGTH (5)
00054
00055 #define SYSTEM_CONFIC_RECORD_SEPERATOR (",")
00056 #define SYSTEM_CONFIC_RECORD_SEPERATOR_LENGTH (1)
00057
00058 #define SYSTEM_CONFIG_RECORD_WEP_KEY_SIZE_64 (10)
00059 #define SYSTEM_CONFIG_RECORD_WEP_KEY_SIZE_104 (26)
00060 #define SYSTEM_CONFIG_RECORD_WPA_PSK_SIZE_MIN (8)
00061 #define SYSTEM_CONFIG_RECORD_WPA_PSK_SIZE_MAX (63)
00062 #define SYSTEM_CONFIG_RECORD_WEP_KEY_TAG ("WEP:")
00063 #define SYSTEM_CONFIG_RECORD_WPA_PSK_TAG ("WPA:")
00064 #define SYSTEM_CONFIG_RECORD_WPA2_PSK_TAG ("WP2:")
00065 #define SYSTEM_CONFIG_RECORD_KEY_TAG_LENGTH (4)
00066
00067 #define SYSTEM_CONFIG_RECORD_M_FILENAME_SIZE (80)
00068
00069 #define SYSTEM_CONFIG_RECORD_SSID_KEY_LENGTH_MAX \
00070     (SYSTEM_CONFIG_RECORD_M_FILENAME_SIZE -     \
00071      (SYSTEM_CONFIG_RECORD_SSID_TAG_LENGTH + SYSTEM_CONFIC_RECORD_SEPERATOR_LENGTH +
      SYSTEM_CONFIG_RECORD_KEY_TAG_LENGTH + 1))
00072
00073 // These functions replace the system config record with the one that is passed
00074 // in.  It calculates a new checksum, and it ignores the MAC address.
00075
00076 class UDPPacket;   // Forward declaration
00077 extern int (*pLegacyUdp)(UDPPacket &p, int ifn);
00078
00096 int SaveUserParameters(void *pCopyFrom, int len);
00097
00113 void *GetUserParameters(void);
00114
00120 const char *GetReleaseTag();
00121
00122 // Get starting memory pointer into config record.
00123 uint8_t *GetConfigRecordStart();
00124
00125 // Get ending config record storage
00126 uint8_t *GetConfigRecordEnd();
00127
00128 // Erase the config record and fixup any legacy records as required....
00129 void EraseWholeConfigRecord();
00130
00131 uint32_t GetDefaultBaud();
00132 uint32_t GetDefaultSerialPort();
00133 bool ShowBootMessages();
00134
00135 #endif /* _SYSTEM_STORAGE_H */
00136
```

24.496 taskmon.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_TM_H
00006 #define _NB_TM_H
00007 void EnableTaskMonitor();
```

```
00008 /* The folling captures the buffer that would be sent out on stask states.
00009 It returns a null terminated string.
00010
00011 IT is not reentrant, so one should ensure that task scan is not active or it
00012 could have occasional odd results
00013 */
00014 const char *GetNBRTOS_TaskStateBuffer();
00015 #endif
```

24.497 tcp.h File Reference

NetBurner TCP API.

```
#include <predef.h>
#include <ip.h>
#include <nettypes.h>
#include <tcp_private.h>
```

Macros

- #define **TCP_ERR_NORMAL** (0)

  No errors.
- #define **TCP_ERR_TIMEOUT** (-1)

  Socket timed out.
- #define **TCP_ERR_NOCON** (-2)

  No connection exists.
- #define **TCP_ERR_CLOSING** (-3)

  Socket is in the process of closing.
- #define **TCP_ERR_NOSUCH_SOCKET** (-4)

  No such socket exists.
- #define **TCP_ERR_NONE_AVAIL** (-5)

  No new sockets are available.
- #define **TCP_ERR_CON_RESET** (-6)

  Connection has been reset.
- #define **TCP_ERR_CON_ABORT** (-7)

  Connection has been aborted.
- #define **SO_DEBUG** 1

  Reserved.
- #define **SO_NONAGLE** 2

  Disable the Nagle algorithm.
- #define **SO_NOPUSH** 4

  Disable TCP PUSH feature.
- #define **TCP_STATE_CLOSED** (0)

  Socket Closed.
- #define **TCP_STATE_LISTEN** (1)

  Listen Socket.
- #define **TCP_STATE_SYN_SENT** (2)

  Packet with synchronization flag sent.
- #define **TCP_STATE_SYN_RCVD** (3)

  Packet with synchronisation flag received.
- #define **TCP_STATE_WAIT_FOR_ACCEPT** (4)

  Socket waiting for accept.
- #define **TCP_STATE_ESTABLISHED** (5)

  TCP session established.
- #define **TCP_STATE_CLOSE_WAIT** (6)

FIN received from client and socet is in the process of closing.

- #define **TCP_STATE_LAST_ACK** (7)

  Server is in the process of sending its own FIN signal.

- #define **TCP_STATE_FIN_WAIT_1** (8)

  Connection is active but not currently being used.

- #define **TCP_STATE_FIN_WAIT_2** (9)

  Client has received first ACK of the first FIN signal from server.

- #define **TCP_STATE_CLOSING** (10)

  Session is closing, but data may still be processed.

- #define **TCP_STATE_TIME_WAIT** (11)

  Client recognizes connection still open, but not currently being used.

Typedefs

- typedef void() tcp_notify_handler(int tcpFd)

  TCP notification callback type.

Functions

- int accept (int listening_socket, IPADDR *address, uint16_t *port, uint16_t timeout)

  Accept an incoming connection from a listening socket.

- int connect (const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout)

  Make an outgoing TCP connection to a remote host.

- int connectvia (const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, const IPADDR &localIp↩ Address)

  Make an outgoing TCP connection to a remote host using the specified local interface IP address.

- int connectvia (const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, int ifnum)

  Make an outgoing TCP connection to a remote host using the specified local interface number.

- int connectwlocal (const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort, uint32_t timeout, const IPADDR &localIpAddress=IPADDR::NullIP(), int intf=-1)

  Make an outgoing TCP connection to a remote host using the specified local interface number or IP address.

- int listen (const IPADDR &addr, uint16_t port, uint8_t maxpend=5)

  Listen for incoming connections on the specified network interface IP address.

- int listenvia (const IPADDR &addr, uint16_t port, int ifn, uint8_t maxpend=5)

  Listen for incoming connections on the specified network interface IP address.

- int listenvia (const IPADDR &addr, uint16_t port, const IPADDR &localIpAddress, uint8_t maxpend=5)

  Listen for incoming connections on the specified network interface IP address.

- int NoBlockConnect (const IPADDR &ipAddress, uint16_t remotePort)

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.

- int NoBlockConnectVia (const IPADDR &ipAddress, uint16_t remotePort, const IPADDR &interfaceIp↩ Address=IPADDR::NullIP())

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.

- int NoBlockConnectVia (const IPADDR &ipAddress, uint16_t remotePort, int ifnum)

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.

- int NoBlockConnectwlocal (const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort, IPADDR interfaceIpAddress=IPADDR::NullIP(), int ifn=-1)

  Create a file descriptor for a TCP connection and return immediately. This function does not wait for a connection to be established. Before using the file descriptor, the application must verify the connection was successful with TcpGetSocketState(fd). The state will be TCP_STATE_ESTABLISHED when the connection has been successfully established. Note: The connection status can also be obtained with the wireavail() function.
- IPADDR GetSocketRemoteAddr (int fd)

  Returns the IP address of the remote host associated with the specified file descriptor.
- IPADDR GetSocketLocalAddr (int fd)

  Returns the IP address of the local interface associated with the connection.
- uint16_t GetSocketRemotePort (int fd)

  Returns the port number of the remote host associated with the connection.
- uint16_t GetSocketLocalPort (int fd)

  Returns the local port number associated with the connection.
- uint32_t TcpGetLastRxTime (int fd)

  Returns the value of system Time Ticks when the last packet was received. Used for the TCP Keep Alive feature.
- void TcpSendKeepAlive (int fd)

  Send a TCP keep alive packet to a remote host.
- uint32_t TcpGetLastRxInterval (int fd)

  Returns the number of system Time Ticks since the last packet was received. This is the difference between the current system time and lastRxTime of the socket.
- int GetTcpRtxCount (int fd)

  Returns the number of re-transmits that have occurred on the specified connection.
- uint8_t SetOutOfOrderBuffers (int fd, uint8_t max)

  Set the maximum number of out-of-order TCP buffers for the specified TCP socket.
- int setsockoption (int fd, int option)

  Set TCP socket options.
- int clrsockoption (int fd, int option)

  Clear TCP socket options.
- int getsockoption (int fd)

  Returns the options for the specified TCP socket.
- int SetSocketUnackBuffers (int fd, uint8_t val)

  Set the maximum number of outbound TCP buffers in the transmit un-acknowledged list for the specified TCP socket.
- int SetSocketRxBuffers (int fd, int n)

  Set the number of TCP receive buffers for the specified TCP socket.
- int SetSocketTxBuffers (int fd, int n)

  Set the number of TCP transmit buffers for the specified TCP socket.
- int abortsocket (int fd)

  Execute an abort on the specified TCP socket.
- int SockReadWithTimeout (int fd, char *buf, int nbytes, uint32_t timeout)

  Attempt to read from a TCP socket until the timeout expires.
- char SocketPeek (int fd)

  Returns the next char that would be read, 0 if no data.
- int TcpGetSocketInterface (int fd)

  Return the network interface associated with a TCP socket.
- uint8_t TcpGetSocketState (int fd)

  Return the current state of a TCP socket.
- uint16_t TcpGetRxBufferSpaceUsed (int fd)

  Returns the number of bytes used in a socket's RX buffer.
- uint16_t TcpGetTxBufferAvailSpace (int fd)

  Returns the number of bytes available in a socket's TX buffer.
- uint16_t TcpGetTxDataWaiting (int fd)

Returns the number of bytes waiting to be sent in a socket's TX Buffer.

- BOOL TcpAllDataAcked (int socket)

  Check the data acknowledged state of a socket.

- BOOL WaitForSocketFlush (int fd, uint32_t ticks)

  Wait for a socket flush operation to complete. A socket is flushed if all sent data has been acknowledged.

- void RegisterTCPReadNotify (int tcpFd, tcp_notify_handler *notifyHandler)

  Register a TCP socket and callback function for read notifications.

- void RegisterTCPWriteNotify (int tcpFd, tcp_notify_handler *notifyHandler)

  Register a TCP socket and callback function for write notifications.

24.497.1 Detailed Description

NetBurner TCP API.

24.498 tcp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00014 #ifndef _NB_TCP_H
00015 #define _NB_TCP_H
00016
00017 #include <predef.h>
00018 #include <ip.h>
00019 #include <nettypes.h>
00020 #include <tcp_private.h>
00021
00022 #ifndef IPV6
00023 #ifndef IPV4ONLY
00024 #error Must select an IP version in tcp.h
00025 #endif
00026 #endif
00027
00028 /* TCP socket status and results */
00037 #define TCP_ERR_NORMAL (0)
00038 #define TCP_ERR_TIMEOUT (-1)
00039 #define TCP_ERR_NOCON (-2)
00040 #define TCP_ERR_CLOSING (-3)
00041 #define TCP_ERR_NOSUCH_SOCKET (-4)
00042 #define TCP_ERR_NONE_AVAIL (-5)
00043 #define TCP_ERR_CON_RESET (-6)
00044 #define TCP_ERR_CON_ABORT (-7)
00067 int accept(int listening_socket, IPADDR *address, uint16_t *port, uint16_t timeout);
00068
00069 #ifdef IPV6
00070 int accept4(int listening_socket, IPADDR4 *address, uint16_t *port, uint16_t timeout);
00071 #define accept6 accept
00072 #else
00073 #define accept4 accept
00074 #endif
00075
00089 inline int connect(const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout)
00090 {
00091     return CoreConnect(ipAddress, 0, remotePort, timeout, IPADDR::NullIP(), -1);
00092 };
00093
00112 inline int connectvia(const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, const IPADDR
      &localIpAddress)
00113 {
00114     return CoreConnect(ipAddress, 0, remotePort, timeout, localIpAddress, -1);
00115 };
00116
00136 inline int connectvia(const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, int ifnum)
00137 {
00138     return CoreConnect(ipAddress, 0, remotePort, timeout, IPADDR::NullIP(), ifnum);
00139 };
00140
00166 inline int connectwlocal(const IPADDR &ipAddress,
00167                          uint16_t localPort,
00168                          uint16_t remotePort,
00169                          uint32_t timeout,
00170                          const IPADDR &localIpAddress = IPADDR::NullIP(),
00171                          int intf = -1)
00172 {
```

```
00173     return CoreConnect(ipAddress, localPort, remotePort, timeout, localIpAddress, intf);
00174 };
00175
00176 #ifdef IPV6
00177 inline int connect6(const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout)
00178 {
00179     return CoreConnect(ipAddress, 0, remotePort, timeout, IPADDR::NullIP(), -1);
00180 };
00181 inline int connectvia6(const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, const IPADDR
      &localIpAddress)
00182 {
00183     return CoreConnect(ipAddress, 0, remotePort, timeout, localIpAddress, -1);
00184 };
00185 inline int connectvia6(const IPADDR &ipAddress, uint16_t remotePort, uint32_t timeout, int ifnum)
00186 {
00187     return CoreConnect(ipAddress, 0, remotePort, timeout, IPADDR::NullIP(), ifnum);
00188 };
00189 inline int connectwlocal6(const IPADDR &ipAddress,
00190                           uint16_t localPort,
00191                           uint16_t remotePort,
00192                           uint32_t timeout,
00193                           const IPADDR &localIpAddress = IPADDR::NullIP(),
00194                           int ifn = -1)
00195 {
00196     return CoreConnect(ipAddress, localPort, remotePort, timeout, localIpAddress, ifn);
00197 };
00198
00199 inline int connect4(IPADDR4 ipAddress, uint16_t remotePort, uint32_t timeout)
00200 {
00201     return CoreConnect4(ipAddress, 0, remotePort, timeout, IPADDR4::NullIP(), -1);
00202 };
00203 inline int connectvia4(IPADDR4 ipAddress, uint16_t remotePort, uint32_t timeout, IPADDR4
      localIpAddress)
00204 {
00205     return CoreConnect4(ipAddress, 0, remotePort, timeout, localIpAddress, -1);
00206 };
00207 inline int connectvia4(IPADDR4 ipAddress, uint16_t remotePort, uint32_t timeout, int ifnum)
00208 {
00209     return CoreConnect4(ipAddress, 0, remotePort, timeout, IPADDR4::NullIP(), ifnum);
00210 };
00211 inline int connectwlocal4(IPADDR4 ipAddress,
00212                           uint16_t localPort,
00213                           uint16_t remotePort,
00214                           uint32_t timeout,
00215                           IPADDR4 localIpAddress = IPADDR4::NullIP(),
00216                           int ifn = -1)
00217 {
00218     return CoreConnect4(ipAddress, localPort, remotePort, timeout, localIpAddress, ifn);
00219 };
00220 #else
00221 #define connect4 connect
00222 #define connectvia4 connectvia
00223 #define connectwlocal4 connectwlocal
00224 #endif
00225
00247 inline int listen(const IPADDR &addr, uint16_t port, uint8_t maxpend = 5)
00248 {
00249     return CoreListen(addr, port, -1, IPADDR::NullIP(), maxpend);
00250 };
00251
00270 inline int listenvia(const IPADDR &addr, uint16_t port, int ifn, uint8_t maxpend = 5)
00271 {
00272     return CoreListen(addr, port, ifn, IPADDR::NullIP(), maxpend);
00273 };
00274
00294 inline int listenvia(const IPADDR &addr, uint16_t port, const IPADDR &localIpAddress, uint8_t maxpend
      = 5)
00295 {
00296     return CoreListen(addr, port, -1, localIpAddress, maxpend);
00297 };
00298
00299 #ifdef IPV6
00300
00301 inline int listen6(const IPADDR &addr, uint16_t port, uint8_t maxpend = 5)
00302 {
00303     return CoreListen(addr, port, -1, IPADDR::NullIP(), maxpend);
00304 };
00305
00306 inline int listenvia6(const IPADDR &addr, uint16_t port, int ifn, uint8_t maxpend = 5)
00307 {
00308     return CoreListen(addr, port, ifn, IPADDR::NullIP(), maxpend);
00309 };
00310
00311 inline int listenvia6(const IPADDR &addr, uint16_t port, const IPADDR &localIpAddress, uint8_t maxpend
      = 5)
00312 {
00313     return CoreListen(addr, port, -1, localIpAddress, maxpend);
```

```
00314 };
00315
00316 inline int listen4(IPADDR4 addr, uint16_t port, uint8_t maxpend = 5)
00317 {
00318     return CoreListen4(addr, port, -1, IPADDR4::NullIP(), maxpend);
00319 };
00320
00321 inline int listenvia4(IPADDR4 addr, uint16_t port, int ifn, uint8_t maxpend = 5)
00322 {
00323     return CoreListen(addr, port, ifn, IPADDR4::NullIP(), maxpend);
00324 };
00325
00326 inline int listenvia4(IPADDR4 addr, uint16_t port, IPADDR4 localIpAddress, uint8_t maxpend = 5)
00327 {
00328     return CoreListen4(addr, port, -1, localIpAddress, maxpend);
00329 };
00330
00331 #else
00332 #define listen4 listen
00333 #define listenvia4 listenvia
00334
00335 #endif
00336
00356 inline int NoBlockConnect(const IPADDR &ipAddress, uint16_t remotePort)
00357 {
00358     return NoBlockCoreConnect(ipAddress, 0, remotePort, IPADDR::NullIP(), -1);
00359 };
00360
00385 inline int NoBlockConnectVia(const IPADDR &ipAddress, uint16_t remotePort, const IPADDR
    &interfaceIpAddress = IPADDR::NullIP())
00386 {
00387     return NoBlockCoreConnect(ipAddress, 0, remotePort, interfaceIpAddress, -1);
00388 };
00389
00414 inline int NoBlockConnectVia(const IPADDR &ipAddress, uint16_t remotePort, int ifnum)
00415 {
00416     return NoBlockCoreConnect(ipAddress, 0, remotePort, IPADDR::NullIP(), ifnum);
00417 };
00418
00452 inline int NoBlockConnectwlocal(const IPADDR &ipAddress,
00453                                 uint16_t localPort,
00454                                 uint16_t remotePort,
00455                                 IPADDR interfaceIpAddress = IPADDR::NullIP(),
00456                                 int ifn = -1)
00457 {
00458     return NoBlockCoreConnect(ipAddress, localPort, remotePort, interfaceIpAddress, ifn);
00459 };
00460
00461 #ifdef IPV6
00462 inline int NoBlockConnect6(const IPADDR &ipAddress, uint16_t remotePort)
00463 {
00464     return NoBlockCoreConnect(ipAddress, 0, remotePort, IPADDR::NullIP(), -1);
00465 };
00466
00467 inline int NoBlockConnectVia6(const IPADDR &ipAddress, uint16_t remotePort, const IPADDR
    &interfaceIpAddress = IPADDR::NullIP())
00468 {
00469     return NoBlockCoreConnect(ipAddress, 0, remotePort, interfaceIpAddress, -1);
00470 };
00471
00472 inline int NoBlockConnectVia6(const IPADDR &ipAddress, uint16_t remotePort, int ifnum)
00473 {
00474     return NoBlockCoreConnect(ipAddress, 0, remotePort, IPADDR::NullIP(), ifnum);
00475 };
00476
00477 inline int NoBlockConnectwlocal6(const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort,
    IPADDR interfaceIpAddress, int ifn = -1)
00478 {
00479     return NoBlockCoreConnect(ipAddress, localPort, remotePort, interfaceIpAddress, ifn);
00480 };
00481
00482 inline int NoBlockConnect4(const IPADDR4 &ipAddress, uint16_t remotePort, uint32_t timeout)
00483 {
00484     return NoBlockCoreConnect4(ipAddress, 0, remotePort, IPADDR4::NullIP(), -1);
00485 };
00486
00487 inline int NoBlockConnectVia4(const IPADDR4 &ipAddress, uint16_t remotePort, const IPADDR4
    &interfaceIpAddress = IPADDR4::NullIP())
00488 {
00489     return NoBlockCoreConnect4(ipAddress, 0, remotePort, interfaceIpAddress, -1);
00490 };
00491
00492 inline int NoBlockConnectVia4(const IPADDR4 &ipAddress, uint16_t remotePort, int ifn)
00493 {
00494     return NoBlockCoreConnect4(ipAddress, 0, remotePort, IPADDR4::NullIP(), ifn);
00495 };
00496
```

```
00497 inline int NoBlockConnectwlocal4(const IPADDR4 &ipAddress,
00498                                  uint16_t localPort,
00499                                  uint16_t remotePort,
00500                                  const IPADDR4 &interfaceIpAddress,
00501                                  int ifn = -1)
00502 {
00503     return NoBlockCoreConnect4(ipAddress, localPort, remotePort, interfaceIpAddress, ifn);
00504 };
00505
00506 #else
00507 #define NoBlockConnect4 NoBlockConnect
00508 #define NoBlockConnectVia4 NoBlockConnectVia
00509 #define NoBlockConnectwlocal4 NoBlockConnectwlocal
00510
00511 #endif
00512
00513 int getsocketerror(int fd);
00514
00515 /* Socket status functions */
00516 IPADDR4 GetSocketRemoteAddr4(int fd);
00517 IPADDR4 GetSocketLocalAddr4(int fd);
00518
00519 #ifdef IPV6
00520 bool bRemoteAddrIsIpV6();
00521 bool bLocalAddrIsIpV6();
00522
00523 IPADDR GetSocketRemoteAddr6(int fd);
00524
00525 IPADDR GetSocketLocalAddr6(int fd);
00526
00536 inline IPADDR GetSocketRemoteAddr(int fd)
00537 {
00538     return GetSocketRemoteAddr6(fd);
00539 };
00540
00550 inline IPADDR GetSocketLocalAddr(int fd)
00551 {
00552     return GetSocketLocalAddr6(fd);
00553 };
00554 #else
00555 inline IPADDR4 GetSocketRemoteAddr(int fd)
00556 {
00557     return GetSocketRemoteAddr4(fd);
00558 };
00559 inline IPADDR4 GetSocketLocalAddr(int fd)
00560 {
00561     return GetSocketLocalAddr4(fd);
00562 };
00563 #endif
00564
00574 uint16_t GetSocketRemotePort(int fd);
00575
00585 uint16_t GetSocketLocalPort(int fd);
00586
00587 /*----- Keepalive functions -----*/
00588
00609 uint32_t TcpGetLastRxTime(int fd);
00610
00622 void TcpSendKeepAlive(int fd);
00623
00624 /*
00625  *
00626  * Interval in ticks since last received packet
00627  *
00628  *    Parameters:
00629  *       socket
00630  *
00631  *    Return:
00632  *       Interval is ticks from now until the last received packet
00633  *
00634  *    Notes:
00635  *       Will handle one rollover approx. 6 years but not more than one which
00636  *       given the timeliness of the use of this function should not matter.
00637  *
00638  */
00639
00650 uint32_t TcpGetLastRxInterval(int fd);
00651
00659 int GetTcpRtxCount(int fd); /* return the number of retransmit counts*/
00660
00661 uint8_t SetOutOfOrderbuffers(int fd, uint8_t max);
00662
00680 inline uint8_t SetOutOfOrderBuffers(int fd, uint8_t max)
00681 {
00682     return SetOutOfOrderbuffers(fd, max);
00683 }
00684
```

```
00685 #if defined MULTIHOME || defined AUTOIP
00686
00687 int connectvia4(IPADDR4 addr, uint16_t localport, uint16_t remoteport, uint32_t timeout, IPADDR4 ipa);
00688 int NoBlockconnectvia4(IPADDR4 addr, uint16_t localport, uint16_t remoteport, IPADDR4 ipa, int intfnum
      = -1);
00689 int connectvia4(IPADDR4 addr, uint16_t localport, uint16_t remoteport, uint32_t timeout, int intfnum);
00690
00691 #ifdef IPV6
00692 int connectvia6(const IPADDR &addr, uint16_t localport, uint16_t remoteport, uint32_t timeout, const
      IPADDR &ipa);
00693 inline int connectvia(const IPADDR &addr, uint16_t localport, uint16_t remoteport, uint32_t timeout,
      const IPADDR &ipa)
00694 {
00695     return connectvia6(addr, localport, remoteport, timeout, ipa);
00696 };
00697
00698 int connectvia6(const IPADDR &addr, uint16_t localport, uint16_t remoteport, uint32_t timeout, int
      intfnum);
00699 inline int connectvia(const IPADDR &addr, uint16_t localport, uint16_t remoteport, uint32_t timeout,
      int intfnum)
00700 {
00701     return connectvia6(addr, localport, remoteport, timeout, intfnum);
00702 };
00703
00704 int NoBlockconnectvia6(IPADDR4 addr, uint16_t localport, uint16_t remoteport, const IPADDR &ipa);
00705 inline int NoBlockconnectvia(IPADDR4 addr, uint16_t localport, uint16_t remoteport, const IPADDR &ipa)
00706 {
00707     return NoBlockconnectvia6(addr, localport, remoteport, ipa);
00708 };
00709
00710 int NoBlockconnectvia6(IPADDR4 addr, uint16_t localport, uint16_t remoteport, int intfnum);
00711 inline int NoBlockconnectvia(IPADDR4 addr, uint16_t localport, uint16_t remoteport, int intfnum)
00712 {
00713     return NoBlockconnectvia6(addr, localport, remoteport, intfnum);
00714 };
00715 #else
00716
00717 inline int connectvia(IPADDR4 addr, uint16_t localport, uint16_t remoteport, uint32_t timeout, IPADDR4
      ipa)
00718 {
00719     return connectvia4(addr, localport, remoteport, timeout, ipa);
00720 };
00721
00722 inline int NoBlockconnectvia(IPADDR4 addr, uint16_t localport, uint16_t remoteport, IPADDR4 ipa)
00723 {
00724     return NoBlockconnectvia4(addr, localport, remoteport, ipa);
00725 };
00726
00727 inline int connectvia(IPADDR4 addr, uint16_t localport, uint16_t remoteport, uint32_t timeout, int
      intfnum)
00728 {
00729     return connectvia4(addr, localport, remoteport, timeout, intfnum);
00730 };
00731
00732 inline int NoBlockconnectvia(IPADDR4 addr, uint16_t localport, uint16_t remoteport, int intfnum)
00733 {
00734     return NoBlockconnectvia4(addr, localport, remoteport, intfnum);
00735 };
00736 #endif
00737 #endif
00738
00739 /* Get and set the socket options */
00740
00745 #define SO_DEBUG 1
00746 #define SO_NONAGLE 2
00747 #define SO_NOPUSH 4
00760 int setsockoption(int fd, int option);
00761
00772 int clrsockoption(int fd, int option);
00773
00783 int getsockoption(int fd);
00784
00785 int setsocketackbuffers(int fd, uint8_t val);
00786
00807 inline int SetSocketUnackBuffers(int fd, uint8_t val)
00808 {
00809     return setsocketackbuffers(fd, val);
00810 }
00811
00832 int SetSocketRxBuffers(int fd, int n);
00833
00849 int SetSocketTxBuffers(int fd, int n);
00850
00858 int abortsocket(int fd);
00859
00880 int SockReadWithTimeout(int fd, char *buf, int nbytes, uint32_t timeout);
00881
```

```
00889 char SocketPeek(int fd);
00890
00891 void DumpTcpDebug();
00892 void EnableTcpDebug(uint16_t dbFlags);
00893
00894 #ifdef SPEED_TCP
00895 typedef void(tcp_data_handler)(int fd, puint8_t data, uint16_t len);
00896 void RegisterDataProcessor(int fd, tcp_data_handler *pdh);
00897 #endif
00898
00899
00907 int TcpGetSocketInterface(int fd);
00908
00909 /* TCP socket states RFC 793 (Do not change) */
00910
00915 #define TCP_STATE_CLOSED (0)
00916 #define TCP_STATE_LISTEN (1)
00917 #define TCP_STATE_SYN_SENT (2)
00918 #define TCP_STATE_SYN_RCVD (3)
00919 #define TCP_STATE_WAIT_FOR_ACCEPT (4)
00920 #define TCP_STATE_ESTABLISHED (5)
00921 #define TCP_STATE_CLOSE_WAIT (6)
00922 #define TCP_STATE_LAST_ACK (7)
00923 #define TCP_STATE_FIN_WAIT_1 (8)
00924 #define TCP_STATE_FIN_WAIT_2 (9)
00925 #define TCP_STATE_CLOSING (10)
00926 #define TCP_STATE_TIME_WAIT (11)
00936 uint8_t TcpGetSocketState(int fd);
00937
00948 uint16_t TcpGetRxBufferSpaceUsed(int fd);
00949
00959 uint16_t TcpGetTxBufferAvailSpace(int fd);
00960
00971 uint16_t TcpGetTxDataWaiting(int fd);
00972
00983 BOOL TcpAllDataAcked(int socket);
00984
00997 BOOL WaitForSocketFlush(int fd, uint32_t ticks);
00998
00999 /*-------------------------------------------------------------------------
01000
01001       Expanded file descriptor callback
01002
01003          Parameters:
01004             tcpFd             - Actual connection file descriptor
01005             notifyHandler     - Notification callback
01006
01007          Return:
01008             None
01009
01010          Notes:
01011             Data arrival or an connection error state calls this routine.
01012
01013 *-------------------------------------------------------------------------*/
01037 typedef void(tcp_notify_handler)(int tcpFd);
01038
01049 void RegisterTCPReadNotify(int tcpFd, tcp_notify_handler *notifyHandler);
01050
01062 void RegisterTCPWriteNotify(int tcpFd, tcp_notify_handler *notifyHandler);
01065 /* associate a void * with tcp */
01066 void TCPAssociateExtraData(int tcpFd, void *pData);
01067 void *TCPGetExtraData(int tcpFd);
01068
01069 #ifdef TCP_THROUGHPUT_INFO
01082 uint64_t TCPGetPayloadBytesSent(int tcpFd);
01083
01096 uint64_t TCPGetPayloadBytesReceived(int tcpFd);
01097
01110 uint64_t TCPGetTotalBytesSent(int tcpFd);
01111
01124 uint64_t TCPGetTotalBytesReceived(int tcpFd);
01125 #endif
01126
01127 #endif /* #ifndef _NB_TCP_H */
01128
```

24.499 tcp_private.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_PVT_TCP_H
00006 #define _NB_PVT_TCP_H
00007
```

```
00008 /*************************************************************************/
00009 /*   Internal TCP Definitions                                            */
00010 /*************************************************************************/
00011 typedef struct
00012 {
00013     beuint16_t srcPort;
00014     beuint16_t dstPort;
00015     beuint32_t SeqNumber;
00016     beuint32_t AckNumber;
00017     uint8_t header_len;
00018     uint8_t flags;
00019     beuint16_t window;
00020     uint16_t csum;   // Do not make big endian. csum is calculated as if native
00021     beuint16_t urgent;
00022     uint8_t DATA[];   // The data field is actually as long as the packet....
00023 } TCPPKT;
00024
00025 typedef TCPPKT *PTCPPKT;
00026
00027 /*************************************************************************/
00028 /*   Warning  WARNING  WARNING  If  you  use  these  functions on an  */
00029 /*   uninitialized  buffer you will get bogus values for the pointer  */
00030 /*   As  the header length field in the IP packet is not yet set up!  */
00031 /*   Use the GetInitxxx functions instead                              */
00032 /*************************************************************************/
00033
00034 inline PTCPPKT GetTcpPkt(PIPPKT pIp)
00035 {
00036     if (pIp == NULL) { return NULL; }
00037     if (pIp->bVerHdrLen == 0x45) { return (PTCPPKT)pIp->DATA; }
00038     return (PTCPPKT)(pIp->DATA + (((pIp->bVerHdrLen & 0XF) * 4) - 20));
00039 }
00040
00041 inline PTCPPKT GetTcpPkt(PoolPtr p)
00042 {
00043     return GetTcpPkt(GetIpPkt(p));
00044 };
00045 inline PTCPPKT GetTcpPkt(PEFRAME pFrame)
00046 {
00047     return GetTcpPkt(GetIpPkt(pFrame));
00048 };
00049
00050 inline PTCPPKT GetInitTcpPkt4(PIPPKT pIp)
00051 {
00052     pIp->bVerHdrLen = 0x45;
00053     return (PTCPPKT)pIp->DATA;
00054 }
00055
00056 inline PTCPPKT GetInitTcpPkt4(PoolPtr p)
00057 {
00058     return GetInitTcpPkt4(GetIpPkt(p));
00059 };
00060
00061 // Used by the IP.cpp process packet function
00062 void process_tcp4(PoolPtr pp, PIPPKT pIP, uint16_t csum);
00063
00064 void DumpTcpDebug();
00065 void EnableTcpDebug(uint16_t);
00066 void DumpTcpPacket(PIPPKT pIP);
00067
00068 int CoreConnect(const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort, uint32_t timeout,
      const IPADDR &ifip, int ifn);
00069 int NoBlockCoreConnect(const IPADDR &ipAddress, uint16_t localPort, uint16_t remotePort, const IPADDR
      &ifip, int ifn);
00070
00071 #ifdef IPV6
00072 inline int CoreConnect4(IPADDR4 ipAddress, uint16_t localPort, uint16_t remotePort, uint32_t timeout,
      IPADDR4 ifip, int ifn)
00073 {
00074     IPADDR ipl = ipAddress;
00075     IPADDR ipf;
00076     if (ifip.IsNull()) { ipf = IPADDR::NullIP(); }
00077     else
00078         ipf = ifip;
00079     return CoreConnect(ipl, localPort, remotePort, timeout, ipf, ifn);
00080 }
00081
00082 inline int NoBlockCoreConnect4(IPADDR4 ipAddress, uint16_t localPort, uint16_t remotePort, IPADDR4
      ifip, int ifn)
00083 {
00084     IPADDR ipl = ipAddress;
00085     IPADDR ipf;
00086     if (ifip.IsNull()) { ipf = IPADDR::NullIP(); }
00087     else
00088         ipf = ifip;
00089     return NoBlockCoreConnect(ipl, localPort, remotePort, ipf, ifn);
00090 }
```

```
00091 #endif
00092
00093 int CoreListen(const IPADDR &addr, uint16_t port, int ifn, const IPADDR &localIpAddress, uint8_t
      maxpend = 5);
00094
00095 #ifdef IPV6
00096 inline int CoreListen4(IPADDR4 ipAddress, uint16_t port, int ifn, const IPADDR4 localIpAddress,
      uint8_t maxpend)
00097 {
00098     IPADDR ipl = ipAddress;
00099     IPADDR ipf;
00100     if (localIpAddress.IsNull()) { ipf = IPADDR4::NullIP(); }
00101     else
00102         ipf = localIpAddress;
00103     return CoreListen(ipl, port, ifn, ipf, maxpend);
00104 }
00105 #endif
00106
00107 void InitTCP();
00108
00109 #endif
```

24.500 tftp.h File Reference

NetBurner TFTP API.

`#include <nettypes.h>`

Macros

- #define **TFTP_OK** (0)

  Success.
- #define **TFTP_TIMEOUT** (1)

  TFTP Timeout.
- #define **TFTP_ERROR** (2)

  TFTP Error.

Functions

- int GetTFTP (PCSTR fileName, PCSTR mode, puint8_t buffer, int &len, uint32_t timeout, IPADDR4 server, uint16_t port=IANA_TFTP_PORT)

  Get a file from a TFTP server.
- int SendTFTP (PCSTR fileName, PCSTR mode, puint8_t buffer, int len, uint32_t timeout, uint32_t packet↩ Timeout, IPADDR4 server, uint16_t port=IANA_TFTP_PORT)

  Send a file to a TFTP server.

24.500.1 Detailed Description

NetBurner TFTP API.

24.501 tftp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00014 #ifndef _NB_TFTP_H
00015 #define _NB_TFTP_H
00016 #include <nettypes.h>
00017 #define IANA_TFTP_PORT (69)
00018
00022 #define TFTP_OK (0)
00023 #define TFTP_TIMEOUT (1)
00024 #define TFTP_ERROR (2)
00044 int GetTFTP(PCSTR fileName, PCSTR mode, puint8_t buffer, int &len, uint32_t timeout, IPADDR4 server,
      uint16_t port = IANA_TFTP_PORT);
```

```
00045
00065 int SendTFTP(PCSTR fileName,
00066              PCSTR mode,
00067              puint8_t buffer,
00068              int len,
00069              uint32_t timeout,
00070              uint32_t packetTimeout,
00071              IPADDR4 server,
00072              uint16_t port = IANA_TFTP_PORT);
00073
00074 #endif
00075
```

24.502 timezones.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef NB_TIMEZONE_H
00005 #define NB_TIMEZONE_H
00006
00007 struct TimeZoneRecord
00008 {
00009     const char *Posix;
00010     const char *Description;
00011     const char *Name;
00012     const char *Short_name;
00013     int bUsCanada;
00014 };
00015
00016 extern const TimeZoneRecord *TZRecords;
00017
00018 #endif
```

24.503 udp.h File Reference

NetBurner User Datagram Protocol Header File.

```
#include <predef.h>
#include <ip.h>
```

Classes

- class UDPPacket

  UDP Packet Class.

Macros

- #define **UDP_ERR_NOSUCH_SOCKET** (-1)

  Socket does not exist.
- #define **UDP_ERR_NOTOPEN_TO_WRITE** (-2)

  Socket not open for write.
- #define **UDP_ERR_NOTOPEN_TO_READ** (-3)

  Socket not open for read.

Functions

- bool RegisterUDPFifo (uint16_t listenPort, OS_FIFO *pFifo)

  Register a FIFO to receive incoming UDP packets.
- bool RegisterUDPFifoVia (uint16_t listenPort, OS_FIFO *pFifo, int interface)

  Register a FIFO to receive incoming UDP packets on the specified network interface.
- uint16_t RegisterEphemeralFifo (OS_FIFO *pfifo, int ifn=-1)

  Register a UDP FIFO with a random port number to receive incoming UDP packets.
- bool RegisterUDPFifoWithNotify (uint16_t listenPort, OS_FIFO *pFifo, udp_data_notify *pNotifyFunction)

Register a FIFO to receive incoming UDP packets and a callback function to receive a notification when a packet is received.

- bool RegisterUDPFifoWithNotifyVia (uint16_t listenPort, OS_FIFO *pFifo, udp_data_notify *pNotifyFunction, int interface)

  Register a FIFO to receive incoming UDP packets and a callback function to receive a notification when a packet is received. Same as RegisterUDPFifoWithNotify() with the additional parameter to specify the local network interface.
- void UnregisterUDPFifo (uint16_t listenPort, bool drain=false)

  Unregister a UDP FIFO.
- int CreateRxUdpSocket (uint16_t listening_port)

  Open a UDP socket for receiving incoming UDP packets.
- int CreateTxUdpSocket (const IPADDR &send_to_addr, uint16_t remote_port, uint16_t local_port)

  Open a UDP socket for transmitting UDP packets.
- int CreateRxTxUdpSocket (const IPADDR &send_to_addr, uint16_t send_to_remote_port, uint16_t local_port)

  Open a UDP socket that can transmit and receive UDP packets.
- int sendto (int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t remote_port)

  Send a UDP packet.
- int sendtovia (int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t remote_port, int intfnum)

  Send a UDP packet on the specified interface.
- int recvfrom (int sock, puint8_t buffer, int len, IPADDR *pAddr, uint16_t *pLocal_port, uint16_t *pRemote_port)

  Receive a UDP packet.
- int SendFragmentedUdpPacket (const IPADDR &to, uint16_t source_port, uint16_t dest_port, puint8_t data, int length)

  Send a large fragmented UDP packet.
- int CreateRxUdpSocketVia (uint16_t listening_port, int interfacenum)

  Open a UDP socket for receiving incoming UDP packets on the specified network interface, by interface number.
- int CreateRxUdpSocketVia (uint16_t listening_port, const IPADDR interfaceIp)

  Open a UDP socket for receiving incoming UDP packets on the network interface with the specified IP address. If more than one interface has the same IP address, the lower interface number is used.
- int CreateRxTxUdpSocketVia (const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t local_port, int interfacenum)

  Open a UDP socket that can transmit and receive UDP packets on the local network interface specified by an interface number.
- int CreateRxTxUdpSocketVia (const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t local_port, IPADDR interfaceIp)

  Open a UDP socket that can transmit and receive UDP packets on the local network interface specified by an IP address.
- int CreateTxUdpSocketVia (const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port, int interfacenum)

  Open a UDP socket for transmitting UDP packets on the local network interface specified by the interface number.
- int CreateTxUdpSocketVia (const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port, IPADDR interfaceIp)

  Open a UDP socket for transmitting UDP packets on the local network interface specified by the interface IP address.

24.503.1 Detailed Description

NetBurner User Datagram Protocol Header File.

24.504 udp.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00019 #ifndef _NB_UDP_H_
00020 #define _NB_UDP_H_
00021 #include <predef.h>
00022 #include <ip.h>
00023
00024 #ifdef IPV6
00025 struct IPv6FrameProcessingStruct;   // Forward
00026 #endif
00027
00028 #ifdef IPV6
00029 struct IP6FRAME;
00030 #endif
00031
00032 struct UdpSocketDataSet
00033 {
00034     OS_FIFO the_fifo;
00035     int fd = 0;
00036     IPADDR address;
00037     int ifn;
00038     uint16_t rxport = 0;
00039     uint16_t lport = 0;
00040     uint16_t rport = 0;
00041     uint16_t io_counter=0;
00042 };
00043
00078 class UDPPacket
00079 {
00080    protected:
00081     PoolPtr m_p;
00082     PUDPPKT m_pPkt;
00083
00084 #ifdef IPV6
00085     IP6FRAME *m_pIP6f;
00086 #endif
00087
00088 #ifdef UDP_FRAGMENTS
00089     BOOL m_bIsFragBuffer;
00090 #endif
00091
00092     PIPPKT GetIpPacket();
00093     int BuildHeaders4(IPADDR4 destIP, IPADDR4 *srcIP, uint8_t ttl);
00094     friend PoolPtr CreateDnsResponse4(IPADDR4 serverIp, IPADDR4 clientIp, uint16_t serverPort,
      uint16_t clientPort, const char *name, IPADDR hostIp, uint32_t ttl);
00095
00096     UDPPacket(UDPPacket *pkt,bool clone);
00097
00098
00099    public:
00116     UDPPacket(OS_FIFO *pFifo, TickTimeout timeout);
00117
00118     UDPPacket(OS_FIFO *pFifo, uint32_t timeout);
00119
00133     UDPPacket(int sock);
00134
00135 #ifdef IPV6
00136     UDPPacket(IPv6FrameProcessingStruct &p6);
00137     void Fix_6_Sb_4();
00138 #endif
00139
00150     UDPPacket(PoolPtr p);
00151
00161     UDPPacket(UDPPacket &pkt);
00162
00163 #ifdef IPV6
00164     UDPPacket(bool bIsIpv6 = false);
00165 #else
00166     UDPPacket();
00167 #endif
00168
00174     ~UDPPacket();
00175
00176     // Returns true if it was able to receive another packet into this UDP object.
00177     inline bool Receive(OS_FIFO *pFifo, uint32_t wait_ticks)
00178     {
00179         if (m_p != 0)
00180         {
00181 #ifdef UDP_FRAGMENTS
00182             FragFreeBuffer(m_p);
00183 #else
```

```
00184             FreeBuffer(m_p);
00185 #endif
00186
00187             m_p = NULL;
00188         }
00189
00190         m_p = OSPoolFifoPend(pFifo, wait_ticks);
00191
00192         if (m_p)
00193         {
00194 #ifdef UDP_FRAGMENTS   // Do a fragment test
00195             puint8_t pb = (puint8_t)m_p;
00196
00197             if ((((puint8_t)m_p) >= pFragmentBuffers) && (((puint8_t)m_p) < pMaxFragmentBuffer))
00198             {
00199                 m_bIsFragBuffer = TRUE;
00200                 m_pPkt = (PUDPPKT)(((PIPPKT)(pb + 4))->DATA);
00201             }
00202             else
00203             {
00204                 m_bIsFragBuffer = FALSE;
00205                 m_pPkt = GetUdpPkt(m_p);
00206                 if (m_p)
00207                 {
00208 #ifdef UDP_FRAGMENTS   // Do a fragment test
00209                     puint8_t pb = (puint8_t)m_p;
00210
00211                     if ((((puint8_t)m_p) >= pFragmentBuffers) && (((puint8_t)m_p) <
    pMaxFragmentBuffer))
00212                     {
00213                         m_bIsFragBuffer = TRUE;
00214                         m_pPkt = (PUDPPKT)(((PIPPKT)(pb + 4))->DATA);
00215                     }
00216                     else
00217                     {
00218                         m_bIsFragBuffer = FALSE;
00219                         m_pPkt = GetUdpPkt(m_p);
00220                     }
00221 #else
00222                     SetUpUdpPkt(m_p);
00223 #endif
00224                 }
00225                 else
00226                 {
00227                     m_pPkt = NULL;
00228                 }
00229             }
00230 #else
00231             SetUpUdpPkt(m_p);
00232 #endif
00233         }
00234         else
00235         {
00236             m_pPkt = NULL;
00237         }
00238
00239         if (m_p == NULL) return false;
00240         return true;
00241     };
00242
00250     void SetSourcePort(uint16_t port);
00251
00259     uint16_t GetSourcePort(void) const;
00260
00261     IPADDR4 GetSourceAddress4(void);
00262     IPADDR4 GetDestinationAddress4();
00263
00269     MACADR GetMacSource();
00270
00271 #ifdef IPV6
00272     IPADDR GetSourceAddress6(void);
00273     IPADDR GetDestinationAddress6();
00274
00275     void SetUpUdpPkt(PoolPtr pp);
00276     PUDPPKT InitUdpPkt(bool IpV6 = false);
00277
00283     inline IPADDR GetSourceAddress(void) { return GetSourceAddress6(); };
00284
00290     inline IPADDR GetDestinationAddress(void) { return GetDestinationAddress6(); }
00291
00297     inline bool bIsIPV6() { return (m_pIP6f != NULL); };
00298 #else
00299     inline IPADDR4 GetSourceAddress(void) { return GetSourceAddress4(); };
00300     inline IPADDR4 GetDestinationAddress() { return GetDestinationAddress4(); }
00301
00302     /* Get UDP Packet pointer from network buffer pool buffer */
00303     inline PUDPPKT GetUdpPkt(PoolPtr p) { return ::GetUdpPkt(GetIpPkt(p)); };
```

```
00304
00305     inline void SetUpUdpPkt(PoolPtr pp) { m_pPkt = GetUdpPkt(pp); };
00306
00307     inline PUDPPKT GetInitUdpPkt(PIPPKT pIp)
00308     {
00309         pIp->bVerHdrLen = 0x45;
00310         return (PUDPPKT)pIp->DATA;
00311     }
00312
00313     inline PUDPPKT GetInitUdpPkt(PoolPtr p) { return GetInitUdpPkt(GetIpPkt(p)); };
00314 #endif
00315
00323     void SetDestinationPort(uint16_t port);
00324
00332     uint16_t GetDestinationPort(void) const;
00333
00339     uint16_t GetPacketId(void);
00340
00346     puint8_t GetDataBuffer(bool bReAllocateIfNeeded = false);
00347
00355     void SetDataSize(uint16_t numBytes);
00356
00364     uint16_t GetDataSize(void) const;
00365
00374     void AddData(puint8_t pData, uint16_t len);
00375
00383     void AddData(PCSTR pData);
00384
00392     void AddDataWord(uint16_t w);
00393
00401     void AddDataByte(uint8_t b);
00402
00411     BOOL Validate(void);   // Check the Checksum
00412
00417     void ResetData(void);   // Zero the data buffer and return.
00418
00419     void SendAndKeep4(IPADDR4 destIP, uint8_t ttl = 0);
00420     void Send4(IPADDR4 destIP, uint8_t ttl = 0);
00421     void SendAndKeepViaInterfaceNum4(IPADDR4 to, int interface, uint8_t ttl = 0);
00422     void SendViaInterfaceNum4(IPADDR4 destIP, int interface, uint8_t ttl = 0);
00423
00424 #ifdef IPV6
00425     void SendAndKeep6(const IPADDR &to, uint8_t ttl = 0);
00426     void Send6(const IPADDR &to, uint8_t ttl = 0);
00427     void SendAndKeepViaInterfaceNum6(const IPADDR &to, int interface, uint8_t ttl = 0);
00428     void SendViaInterfaceNum6(const IPADDR &to, int interface, uint8_t ttl = 0);
00429
00438     inline void SendAndKeep(const IPADDR &to, uint8_t ttl = 0) { SendAndKeep6(to, ttl); };
00439
00449     inline void Send(const IPADDR &to, uint8_t ttl = 0) { Send6(to, ttl); };
00450
00461     inline void SendAndKeepViaInterfaceNum(const IPADDR &to, int interface, uint8_t ttl = 0)
00462     {
00463         SendAndKeepViaInterfaceNum6(to, interface, ttl);
00464     }
00465
00475     inline void SendViaInterfaceNum(const IPADDR &to, int interface, uint8_t ttl = 0) {
      SendViaInterfaceNum6(to, interface, ttl); };
00476
00477 #else
00478     inline void SendAndKeep(IPADDR4 to, uint8_t ttl = 0) { SendAndKeep4(to, ttl); };
00479     inline void Send(IPADDR4 to, uint8_t ttl = 0) { Send4(to, ttl); };
00480     inline void SendAndKeepViaInterfaceNum(IPADDR4 to, int interface, uint8_t ttl = 0) {
      SendAndKeepViaInterfaceNum4(to, interface, ttl); };
00481     inline void SendViaInterfaceNum(IPADDR4 to, int interface, uint8_t ttl = 0) {
      SendViaInterfaceNum4(to, interface, ttl); };
00482 #endif
00483
00484 #ifdef UDP_FRAGMENTS
00485     void FragFreeBuffer(PoolPtr mp);
00486     void ReleaseBuffer(void);
00487     PoolPtr GetPoolPtr(void);
00488     void FixTransmitBuffers();
00489 #else
00490     void ReleaseBuffer(void)
00491     {
00492         m_p = NULL;
00493         m_pPkt = NULL;
00494
00495 #ifdef IPV6
00496         m_pIP6f = NULL;
00497 #endif
00498     };
00499
00500     PoolPtr GetPoolPtr(void) { return m_p; };
00501 #endif
00502
```

```
00503 #if defined MULTIHOME || defined AUTOIP
00504     void SendViaIfAddr4(IPADDR4 to, IPADDR4 *from_ip, uint8_t ttl = 0);
00505     void SendViaIfAddrAndNum4(IPADDR4 to, IPADDR4 *from_ip, int interface, uint8_t ttl = 0);
00506     void SendAndKeepViaIfAddr4(IPADDR4 to, IPADDR4 *from_ip, uint8_t ttl = 0);
00507
00508 #ifdef IPV6
00509     void SendAndKeepViaIfAddr6(const IPADDR &to, const IPADDR &from_ip, uint8_t ttl = 0);
00510
00522     inline void SendAndKeepViaIfAddr(const IPADDR &to, const IPADDR &from_ip, uint8_t ttl = 0) {
    SendAndKeepViaIfAddr6(to, from_ip, ttl); }
00523     // Send destroys the data in the packet.
00524     void SendViaIfAddr6(const IPADDR &to, const IPADDR &from_ip, uint8_t ttl = 0);
00525
00537     inline void SendViaIfAddr(const IPADDR &to, const IPADDR &from_ip, uint8_t ttl = 0) {
    SendViaIfAddr(to, from_ip, ttl); };
00538 #else
00539     // Send and keep makes a copy of this UDP packet and sends it.
00540     inline void SendAndKeepViaIfAddr(IPADDR4 to, IPADDR4 from_ip, uint8_t ttl = 0) {
    SendAndKeepViaIfAddr4(to, &from_ip, ttl); }
00541     // Send destroys the data in the packet.
00542     inline void SendViaIfAddr(IPADDR4 to, IPADDR4 from_ip, uint8_t ttl = 0) { SendViaIfAddr4(to,
    &from_ip, ttl); };
00543 #endif   //
00544 #endif   // Multi home
00545
00546     bool process_all_udp();
00547
00548     // Uncomment when C++11 is enabled.
00549     // UDPPacket & operator = (const UDPPacket & pf) =delete;
00550
00551     // Destructive Copy constructor destroys other UdpPacket
00552     inline void CopyFrom(UDPPacket cpyfrm)
00553     {
00554         if (m_p != 0)
00555         {
00556 #ifdef UDP_FRAGMENTS
00557             FragFreeBuffer(m_p);
00558 #else
00559             FreeBuffer(m_p);
00560 #endif
00561
00562             m_p = NULL;
00563         }
00564         m_p = cpyfrm.m_p;
00565         m_pPkt = cpyfrm.m_pPkt;
00566         cpyfrm.m_p = NULL;
00567         cpyfrm.m_pPkt = NULL;
00568
00569 #ifdef IPV6
00570         m_pIP6f = cpyfrm.m_pIP6f;
00571         cpyfrm.m_pIP6f = NULL;
00572 #endif
00573
00574 #ifdef UDP_FRAGMENTS
00575         m_bIsFragBuffer = cpyFrm.m_bIsFragBuffer;
00576 #endif
00577     };
00578
00579     void Swap(UDPPacket &rhs);
00580
00581 };   // end of UDPPacket Class
00582
00583 #ifdef UDP_FRAGMENTS
00584 extern uint8_t pFragmentBuffers[UDP_FRAGMENTS * (65536 + 4)];
00585 extern puint8_t pMaxFragmentBuffer;
00586 #endif
00587
00588 typedef void(udp_data_notify)(OS_FIFO *pfifo, uint16_t port);
00589
00603 bool RegisterUDPFifo(uint16_t listenPort, OS_FIFO *pFifo);
00604
00619 bool RegisterUDPFifoVia(uint16_t listenPort, OS_FIFO *pFifo, int interface);
00620
00636 uint16_t RegisterEphemeralFifo(OS_FIFO *pfifo, int ifn = -1);
00637
00654 bool RegisterUDPFifoWithNotify(uint16_t listenPort, OS_FIFO *pFifo, udp_data_notify *pNotifyFunction);
00655
00674 bool RegisterUDPFifoWithNotifyVia(uint16_t listenPort, OS_FIFO *pFifo, udp_data_notify
    *pNotifyFunction, int interface);
00675
00684 void UnregisterUDPFifo(uint16_t listenPort, bool drain = false);
00685
00686 // INTERNAL FUNCTIONS. Used by the ip.cpp process packet functions
00687
00688 void process_udp4(PoolPtr pp, PIPPKT pIP, uint16_t csum);
00689
00690 #ifdef IPV6
```

```
00691 struct IPv6FrameProcessingStruct;   // Forward
00692 bool process_udp6(IPv6FrameProcessingStruct &p6proc);
00693 #endif
00694
00698 #define UDP_ERR_NOSUCH_SOCKET (-1)
00699 #define UDP_ERR_NOTOPEN_TO_WRITE (-2)
00700 #define UDP_ERR_NOTOPEN_TO_READ (-3)       // end of groupUDP
00704
00705 //----- Wrappers to create a UDP socket interface -----
00706
00728 int CreateRxUdpSocket(uint16_t listening_port);
00729
00730 int CreateTxUdpSocket4(const IPADDR4 send_to_addr, uint16_t remote_port, uint16_t local_port);
00731 int CreateRxTxUdpSocket4(const IPADDR4 send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port);
00732
00733 #ifdef IPV6
00734
00735 int CreateTxUdpSocket6(const IPADDR &send_to_addr, uint16_t remote_port, uint16_t local_port);
00736
00750 inline int CreateTxUdpSocket(const IPADDR &send_to_addr, uint16_t remote_port, uint16_t local_port)
00751 {
00752     return CreateTxUdpSocket6(send_to_addr, remote_port, local_port);
00753 };
00754 #else
00755 inline int CreateTxUdpSocket(const IPADDR4 send_to_addr, uint16_t remote_port, uint16_t local_port)
00756 {
00757     return CreateTxUdpSocket4(send_to_addr, remote_port, local_port);
00758 };
00759 #endif
00760
00761 #ifdef IPV6
00762
00763 int CreateRxTxUdpSocket6(const IPADDR &send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port);
00764
00776 inline int CreateRxTxUdpSocket(const IPADDR &send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port)
00777 {
00778     return CreateRxTxUdpSocket6(send_to_addr, send_to_remote_port, local_port);
00779 };
00780 #else
00781 inline int CreateRxTxUdpSocket(const IPADDR4 send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port)
00782 {
00783     return CreateRxTxUdpSocket4(send_to_addr, send_to_remote_port, local_port);
00784 };
00785 #endif
00786
00787 int sendto4(int sock, puint8_t what_to_send, int len_to_send, IPADDR4 to_addr, uint16_t remote_port);
00788
00789 #ifdef IPV6
00790
00791 int sendto6(int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t
      remote_port);
00792
00806 inline int sendto(int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t
      remote_port)
00807 {
00808     return sendto6(sock, what_to_send, len_to_send, to_addr, remote_port);
00809 };
00810 #else
00811 inline int sendto(int sock, puint8_t what_to_send, int len_to_send, IPADDR4 to_addr, uint16_t
      remote_port)
00812 {
00813     return sendto4(sock, what_to_send, len_to_send, to_addr, remote_port);
00814 };
00815 #endif
00816
00817 #if defined MULTIHOME || defined AUTOIP
00818
00819 int sendtovia4(int sock, puint8_t what_to_send, int len_to_send, IPADDR4 to_addr, uint16_t
      remote_port, int intfnum);
00820
00821 #ifdef IPV6
00822 int sendtovia6(int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t
      remote_port, int intfnum);
00823 [[deprecated]] inline int sendto6via(int sock,
00824                                      puint8_t what_to_send,
00825                                      int len_to_send,
00826                                      const IPADDR &to_addr,
00827                                      uint16_t remote_port,
00828                                      int intfnum)
00829 {
00830     return sendtovia6(sock, what_to_send, len_to_send, to_addr, remote_port, intfnum);
00831 }
00832
```

```
00847 inline int sendtovia(int sock, puint8_t what_to_send, int len_to_send, const IPADDR &to_addr, uint16_t
      remote_port, int intfnum)
00848 {
00849     return sendtovia6(sock, what_to_send, len_to_send, to_addr, remote_port, intfnum);
00850 };
00851 #else
00852 inline int sendtovia(int sock, puint8_t what_to_send, int len_to_send, IPADDR4 to_addr, uint16_t
      remote_port, int intfnum)
00853 {
00854     return sendtovia4(sock, what_to_send, len_to_send, to_addr, remote_port, intfnum);
00855 };
00856 #endif
00857 #endif
00858
00859 int recvfrom4(int sock, puint8_t buffer, int len, IPADDR4 *pAddr, uint16_t *pLocal_port, uint16_t
      *pRemote_port);
00860
00861 #ifdef IPV6
00862
00863 int recvfrom6(int sock, puint8_t buffer, int len, IPADDR *pAddr, uint16_t *pLocal_port, uint16_t
      *pRemote_port);
00864
00865 // If address is IPV4 you get a IPV4 encoded in IPV6 ie ::FFFF.x.x.x.x
00880 inline int recvfrom(int sock, puint8_t buffer, int len, IPADDR *pAddr, uint16_t *pLocal_port, uint16_t
      *pRemote_port)
00881 {
00882     return recvfrom6(sock, buffer, len, pAddr, pLocal_port, pRemote_port);
00883 };
00884 #else
00885 inline int recvfrom(int sock, puint8_t buffer, int len, IPADDR4 *pAddr, uint16_t *pLocal_port,
      uint16_t *pRemote_port)
00886 {
00887     return recvfrom4(sock, buffer, len, pAddr, pLocal_port, pRemote_port);
00888 };
00889 #endif
00890
00891 /* Send a large fragmented UDP packet to is the destination address source_port is the source port
      dest_port
00892  * is the destination port data is the data to send length is the length to send.
00893  *
00894  * returns 0 on failure, length on success
00895  */
00896 int SendFragmentedUdpPacket4(IPADDR4 to, uint16_t source_port, uint16_t dest_port, puint8_t data, int
      length);
00897
00898 #ifdef IPV6
00899 int SendFragmentedUdpPacket6(const IPADDR &to, uint16_t source_port, uint16_t dest_port, puint8_t
      data, int length);
00900
00913 inline int SendFragmentedUdpPacket(const IPADDR &to, uint16_t source_port, uint16_t dest_port,
      puint8_t data, int length)
00914 {
00915     return SendFragmentedUdpPacket6(to, source_port, dest_port, data, length);
00916 };
00917 #else
00918 inline int SendFragmentedUdpPacket(IPADDR4 to, uint16_t source_port, uint16_t dest_port, puint8_t
      data, int length)
00919 {
00920     return SendFragmentedUdpPacket4(to, source_port, dest_port, data, length);
00921 };
00922 #endif
00923
00924 #ifndef IPV6
00925 #ifndef IPV4ONLY
00926 #error Must select an IP version in udp.h
00927 #endif
00928 #endif
00929
00942 int CreateRxUdpSocketVia(uint16_t listening_port, int interfacenum);
00943
00958 int CreateRxUdpSocketVia(uint16_t listening_port, const IPADDR interfaceIp);
00959
00960 int CreateTxUdpSocketVia4(const IPADDR4 send_to_addr, uint16_t remote_port, uint16_t local_port, int
      interfacenum);
00961 int CreateTxUdpSocketVia4(const IPADDR4 send_to_addr, uint16_t remote_port, uint16_t local_port, const
      IPADDR4 interfaceIp);
00962 int CreateRxTxUdpSocketVia4(const IPADDR4 send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port, int interfacenum);
00963 int CreateRxTxUdpSocketVia4(const IPADDR4 send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port, const IPADDR4 interfaceIp);
00964
00965 [[deprecated]] inline int CreateTxUdpSocket4Via(const IPADDR4 send_to_addr, uint16_t remote_port,
      uint16_t local_port, int interfacenum)
00966 {
00967     return CreateTxUdpSocketVia4(send_to_addr, remote_port, local_port, interfacenum);
00968 }
00969 [[deprecated]] inline int CreateTxUdpSocket4Via(const IPADDR4 send_to_addr,
```

```
00970                                                 uint16_t remote_port,
00971                                                 uint16_t local_port,
00972                                                 const IPADDR4 interfaceIp)
00973 {
00974     return CreateTxUdpSocketVia4(send_to_addr, remote_port, local_port, interfaceIp);
00975 }
00976 [[deprecated]] inline int CreateRxTxUdpSocket4Via(const IPADDR4 send_to_addr,
00977                                                    uint16_t send_to_remote_port,
00978                                                    uint16_t local_port,
00979                                                    int interfacenum)
00980 {
00981     return CreateRxTxUdpSocketVia4(send_to_addr, send_to_remote_port, local_port, interfacenum);
00982 }
00983 [[deprecated]] inline int CreateRxTxUdpSocket4Via(const IPADDR4 send_to_addr,
00984                                                    uint16_t send_to_remote_port,
00985                                                    uint16_t local_port,
00986                                                    const IPADDR4 interfaceIp)
00987 {
00988     return CreateRxTxUdpSocketVia4(send_to_addr, send_to_remote_port, local_port, interfaceIp);
00989 }
00990
00991 #ifdef IPV6
00992 int CreateRxTxUdpSocketVia6(const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t
    local_port, int interfacenum);
00993 int CreateRxTxUdpSocketVia6(const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t
    local_port, const IPADDR interfaceIp);
00994 int CreateTxUdpSocketVia6(const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port, int
    interfacenum);
00995 int CreateTxUdpSocketVia6(const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port, const
    IPADDR interfaceIp);
00996
00997 [[deprecated]] inline int CreateRxTxUdpSocket6Via(const IPADDR send_to_addr,
00998                                                    uint16_t send_to_remote_port,
00999                                                    uint16_t local_port,
01000                                                    int interfacenum)
01001 {
01002     return CreateRxTxUdpSocketVia6(send_to_addr, send_to_remote_port, local_port, interfacenum);
01003 }
01004 [[deprecated]] inline int CreateRxTxUdpSocket6Via(const IPADDR send_to_addr,
01005                                                    uint16_t send_to_remote_port,
01006                                                    uint16_t local_port,
01007                                                    const IPADDR interfaceIp)
01008 {
01009     return CreateRxTxUdpSocketVia6(send_to_addr, send_to_remote_port, local_port, interfaceIp);
01010 }
01011 [[deprecated]] inline int CreateTxUdpSocket6Via(const IPADDR send_to_addr, uint16_t remote_port,
    uint16_t local_port, int interfacenum)
01012 {
01013     return CreateTxUdpSocketVia6(send_to_addr, remote_port, local_port, interfacenum);
01014 }
01015 [[deprecated]] inline int CreateTxUdpSocket6Via(const IPADDR send_to_addr,
01016                                                 uint16_t remote_port,
01017                                                 uint16_t local_port,
01018                                                 const IPADDR interfaceIp)
01019 {
01020     return CreateTxUdpSocketVia6(send_to_addr, remote_port, local_port, interfaceIp);
01021 }
01022
01036 inline int CreateRxTxUdpSocketVia(const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t
    local_port, int interfacenum)
01037 {
01038     return CreateRxTxUdpSocketVia6(send_to_addr, send_to_remote_port, local_port, interfacenum);
01039 }
01040
01053 inline int CreateRxTxUdpSocketVia(const IPADDR send_to_addr, uint16_t send_to_remote_port, uint16_t
    local_port, IPADDR interfaceIp)
01054 {
01055     return CreateRxTxUdpSocketVia6(send_to_addr, send_to_remote_port, local_port, interfaceIp);
01056 }
01057
01072 inline int CreateTxUdpSocketVia(const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port,
    int interfacenum)
01073 {
01074     return CreateTxUdpSocketVia6(send_to_addr, remote_port, local_port, interfacenum);
01075 }
01076
01092 inline int CreateTxUdpSocketVia(const IPADDR send_to_addr, uint16_t remote_port, uint16_t local_port,
    IPADDR interfaceIp)
01093 {
01094     return CreateTxUdpSocketVia6(send_to_addr, remote_port, local_port, interfaceIp);
01095 }
01096 #else
01097 inline int CreateRxTxUdpSocketVia(const IPADDR4 send_to_addr, uint16_t send_to_remote_port, uint16_t
    local_port, int interfacenum)
01098 {
01099     return CreateRxTxUdpSocketVia4(send_to_addr, send_to_remote_port, local_port, interfacenum);
01100 }
```

```
01101
01102 inline int CreateRxTxUdpSocketVia(const IPADDR4 send_to_addr, uint16_t send_to_remote_port, uint16_t
      local_port, IPADDR4 interfaceIp)
01103 {
01104     return CreateRxTxUdpSocketVia4(send_to_addr, send_to_remote_port, local_port, interfaceIp);
01105 }
01106
01107 inline int CreateTxUdpSocketVia(const IPADDR4 send_to_addr, uint16_t remote_port, uint16_t local_port,
      int interfacenum)
01108 {
01109     return CreateTxUdpSocketVia4(send_to_addr, remote_port, local_port, interfacenum);
01110 }
01111
01112 inline int CreateTxUdpSocketVia(const IPADDR4 send_to_addr, uint16_t remote_port, uint16_t local_port,
      IPADDR4 interfaceIp)
01113 {
01114     return CreateTxUdpSocketVia4(send_to_addr, remote_port, local_port, interfaceIp);
01115 }
01116
01117 #endif
01118
01119 #endif /* #ifndef _NB_UDP_H_ */
01120
```

24.505 utils.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_UTILS_H
00006 #define _NB_UTILS_H
00007
00008 // NB Definitions
00009 #include <predef.h>
00010
00011 // NB Libs
00012 #include <stddef.h>
00013 #ifndef NB_NET_TYPES_H
00014 #include "nettypes.h"
00015 #endif   // NB_NET_TYPES_H
00016
00017 #ifdef IPV6
00018 #include <ipv6/ipv6_addr.h>
00019 IPADDR AsciiToIp6(const char *p);   // Convert an ASCII IP string to an IP address
00020 #endif
00021
00022 //
00023 // Flags to control the debug print functions
00024 //
00025 #define DB_TCPIP (1)
00026 #define DB_HTTP (2)
00027 #define DB_ETHER (4)
00028 #define DB_RTOS (8)
00029 #define DB_BUFFER (16)
00030 #define DB_PPP (32)
00031 #define DB_AU (64)
00032 #define DB_MAIL (128)
00033 #define DB_IP (256)
00034 #define DB_TCPDATA (512)
00035 #define DB_SSL (1024)
00036 #define DB_SNMP (2048)
00037 #define DB_IPV6_ND (4096)
00038 #define DB_IPV6_ICMP (8192)
00039 #define DB_IPV6_ROUTE (16384)
00040 #define DB_IPV6_FRAG (32768)
00041 #define DB_IPV6_ERR (65536)
00042
00044 // DIAGNOSTIC FUNCTIONS
00045 //
00046 void ShowCounters();                          // Dump all system counters to stdio
00047 void sShowCounters(char *buffer, int slen);   // Dump all system counters to a large memory buffer
00048 #ifdef FILE
00049 void fShowCounters(FILE *fp);   // Dump all system counters to a file descriptor
00050 #endif                          // FILE
00051
00053 // UTILITY I/O FUNCTIONS
00054 //
00055 void FdShowRingData(int fd, const uint8_t *ringBuf, uint32_t bufLen, // Dump part of a ring buffer to
      file descriptor
00056                 uint32_t start, uint32_t end, const char *indent=NULL);
00057 void ShowRingData(const uint8_t *ringBuf, uint32_t bufLen, // Dump part of a ring buffer to stdio
00058                 uint32_t start, uint32_t end, const char *indent=NULL);
00059 void FdShowData(int fd, const uint8_t *fromptr, uint16_t len);   // Dump a block of data to file
      descriptor and show in ASCII and hex
```

```
00060 void ShowData(const uint8_t *fromptr, uint16_t len);   // Dump a block of data to stdio and show in
      ASCII and hex
00061 void ShowMac(const MACADR *ma);                        // Dump a MAC address to stdio
00062 void fdShowMac(int fd, const MACADR *ma);              // Dump a MAC address to stdio
00063
00064 inline void ShowMac(const MACADR &ma)
00065 {
00066     ShowMac(&ma);
00067 };   // Dump a MAC address to stdio
00068
00069 void snShowMac(char *buf, size_t maxlen, const MACADR *ma);   // Dump a MAC address to char buf
00070 inline void snShowMac(char *buf, size_t maxlen, const MACADR &ma)
00071 {
00072     snShowMac(buf, maxlen, &ma);
00073 };
00074
00075 void MacToID(MACADR *ma, char *IDBuf);   // Write 6 character ID string based on unique portion of MAC
00076
00077 void outbyte(char c);                // Write out a single, unbuffered byte to stdio
00078 void print(const char *);            // Write out a zero-terminated, unbuffered string
00079 void putnum(int i);                  // Write out a hexadecimal, unbuffered number to stdio
00080 void putbytenum(unsigned char c);    // Write out a hexadecimal, unbuffered byte to stdio
00081 IPADDR4 AsciiToIp4(const char *p);   // Convert an ASCII IP string to an IP address
00082 MACADR AsciiToMac(const char *p);
00083 BOOL ValidIPv4(const char *p);
00084
00085 #ifndef IPV6
00086 #define AsciiToIp AsciiToIp4
00087 #else
00088 #define AsciiToIp AsciiToIp6
00089 #endif
00090
00091 void ShowIP4(const IPADDR4 ia);                              // Dump an IP address in ASCII IP string
      format to stdio
00092 int snShowIP4(char *buf, size_t maxlen, const IPADDR4 ia);   // Dump an IP address in ASCII IP string
      format to char buf
00093
00094 #ifdef IPV6
00095 // Dump an IP address in ASCII IP string format to stdio
00096 inline void ShowIP6(const IPADDR &ia)
00097 {
00098     ia.print();
00099 };
00100 inline int snShowIP6(char *buf, size_t maxlen, const IPADDR &ia)
00101 {
00102     return ia.sprintf(buf, maxlen);
00103 };
00104 #define ShowIP ShowIP6
00105 #define snShowIP snShowIP6
00106 #else
00107 #define ShowIP ShowIP4
00108 #define snShowIP snShowIP4
00109 #endif
00110
00111 // char *itoa( int value, char *buffer, int radix ); // Converts an integer to ASCII (adds the stdlib
      itoa)
00112
00114 // HIGHER RESOLUTION COUNTER READ FUNCTION
00115 //
00116 extern uint32_t GetPreciseTime(void);   // Gets the time tick since system start at a higher
00117                                         // resolution, depending on the platform:  0.868-us for
00118                                         // MOD5234/70, and 1.929-us for MOD5282
00119
00125 unsigned long long Get_msec();
00126
00127 extern "C"
00128 {
00129     int kill(int pid, int sig);
00130     void _exit(int i);
00131     int _fini(void);
00132 }
00133
00134 #ifdef _STDIO_H_
00135 #ifndef FILE
00136 void fShowCounters(FILE *fp);
00137 #endif   // FILE
00138 void fShowMac(FILE *fp, const MACADR *ma);
00139
00140 void fShowIP4(FILE *fp, const IPADDR4 ia);
00141
00142 #ifdef IPV6
00143 void fShowIP6(FILE *fp, const IPADDR &ia);
00144 #define fShowIP fShowIP6
00145 #else
00146 #define fShowIP fShowIP4
00147 #endif
00148
```

```
00149 #endif   // _STDIO_H_
00150
00151 /*
00152   Convert Binary to/from Hexadecimal ASCII
00153
00154   Parameters:
00155      fromBufferPtr      - Source data
00156      fromByteCount      - Source byte count
00157      toBufferPtr        - Converted data
00158      toByteCount        - Converted byte count
00159
00160   Return:
00161      Pointer to next free byte in converted buffer
00162
00163   Notes:
00164      Hexadecimal ASCII buffer must be at least twice the size of the binary
00165      buffer size.
00166
00167 */
00168 unsigned char *convertBinaryToHexAscii(unsigned char *fromBufferPtr,
00169                                        unsigned int fromByteCount,
00170                                        unsigned char *toBufferPtr,
00171                                        unsigned int toByteCount);
00172
00173 unsigned char *convertHexAsciiToBinary(unsigned char *fromBufferPtr,
00174                                        unsigned int fromByteCount,
00175                                        unsigned char *toBufferPtr,
00176                                        unsigned int toByteCount);
00177
00178 /* Search for a C string in an arbitrary memory blob that may contain NULL
00179  * chars
00180  *
00181  * Paramemeters:
00182  *      seacrh          - pointer to the buffer to search
00183  *      target          - the C string being searched for
00184  *      len             - length of the search buffer
00185  *
00186  *  Return:
00187  *      pointer to the start of the target string in the search buffer
00188  *          or
00189  *      NULL if target not found in search buffer
00190  */
00191 const char *bufnstr(const char *search, const char *target, size_t len);
00192
00194 // DEBUG FUNCTIONS
00195 //
00196 #if ((defined _DEBUG) || (defined _DEBUG_PRINT))
00197 extern unsigned int DB_FLAGS;
00198 void SetLogLevel();
00199 #ifdef COLDFIRE
00200 #define ASSERT(x)                                                       \
00201     if (!(x))                                                           \
00202     {                                                                   \
00203         iprintf("ASSERT failed at %d of %s\n", __LINE__, __FILE__); \
00204         asm("Halt");                                                    \
00205     }
00206 #elif defined CORTEX_M7
00207 #define ASSERT(x)                                                       \
00208     if (!(x))                                                           \
00209     {                                                                   \
00210         iprintf("ASSERT failed at %d of %s\n", __LINE__, __FILE__); \
00211         while (1)                                                       \
00212             asm("wfi");                                                 \
00213     }
00214 #else
00215 #define ASSERT(x)                                                       \
00216     if (!(x))                                                           \
00217     {                                                                   \
00218         iprintf("ASSERT failed at %d of %s\n", __LINE__, __FILE__); \
00219         while (1)                                                       \
00220             ;                                                           \
00221     }
00222 #endif
00223 #define DBPRINT(x, y) \
00224     if (DB_FLAGS & x) printf(y);
00225 #define DBNUM(x, y) \
00226     if (DB_FLAGS & x) putnum(y);
00227 #define DBBYTE(x, y) \
00228     if (DB_FLAGS & x) putbytenum(y);
00229 #define DBPRINTF(x, ...) \
00230     if (DB_FLAGS & x) printf(__VA_ARGS__);
00231 #define DBIPRINTF(x, ...) \
00232     if (DB_FLAGS & x) iprintf(__VA_ARGS__);
00233 #else
00234 #define ASSERT(x) ((void)0)
00235 #define DBPRINT(x, y) ((void)0)
00236 #define DBNUM(x, y) ((void)0)
```

```
00237 #define DBBYTE(x, y) ((void)0)
00238 #define DBPRINTF(x, ...) ((void)0)
00239 #define DBIPRINTF(x, ...) ((void)0)
00240 #endif   // _DEBUG
00241
00242 #if ((defined _DEBUG) || (defined _DEBUG_PRINT))
00243 #define PPPDBPRINT(x) DBPRINT(DB_PPP, (x))
00244 #define PPPDBPRINTF(...) DBPRINTF(DB_PPP, __VA_ARGS__)
00245 #define PPPDBIPRINTF(...) DBIPRINTF(DB_PPP, __VA_ARGS__)
00246 #define PPPDBNUM(x) DBNUM(DB_PPP, (x))
00247 #define PPPDBBYTE(x) DBBYTE(DB_PPP, (x))
00248 #else
00249 /*
00250    #define PPPDBPRINT(x) printf(x);
00251         #define PPPDBPRINTF(...)  printf(__VA_ARGS__)
00252         #define PPPDBIPRINTF(...) printf(__VA_ARGS__)
00253         #define PPPDBNUM(x) printf("%08X",x)
00254         #define PPPDBBYTE(x) printf("%02X",x)
00255 */
00256 #define PPPDBPRINT(x)
00257 #define PPPDBPRINTF(...)
00258 #define PPPDBIPRINTF(...)
00259 #define PPPDBNUM(x)
00260 #define PPPDBBYTE(x)
00261
00262 #endif
00263
00264 void LocalOutByte(char c);
00265 extern "C"
00266 {
00267     void Local_OutString(const char *cp);
00268 }
00270 // USER_ENTER_CRITICAL ASSERTION
00271 //
00272 #if ((defined _DEBUG) || (defined USER_CRITICAL_SANITY))
00273 #ifdef COLDFIRE
00274 #define USER_CRITICAL_ASSERT()                                                  \
00275     if (critical_count != 0)                                                    \
00276     {                                                                           \
00277         Local_OutString("Illegal pend after USER_ENTER_CRITICAL(). Halting.\r\n"); \
00278         while (1)                                                               \
00279             asm("Halt");                                                        \
00280     }
00281 #elif (defined CORTEX_M7)
00282 #define USER_CRITICAL_ASSERT()                                                  \
00283     if (critical_count != 0)                                                    \
00284     {                                                                           \
00285         Local_OutString("Illegal pend after USER_ENTER_CRITICAL(). Halting.\r\n"); \
00286         while (1)                                                               \
00287             asm("wfi");                                                         \
00288     }
00289 #else
00290 #define USER_CRITICAL_ASSERT()                                                  \
00291     if (critical_count != 0)                                                    \
00292     {                                                                           \
00293         Local_OutString("Illegal pend after USER_ENTER_CRITICAL(). Halting.\r\n"); \
00294         while (1)                                                               \
00295             ;                                                                   \
00296     }
00297 #endif
00298 #else   // !((defined _DEBUG) || (defined USER_CRITICAL_SANITY))
00299 #define USER_CRITICAL_ASSERT() ((void)0)
00300 #endif
00301
00303 // DEVELOPMENT BOARD I/O FUNCTIONS
00304 //
00305 BOOL OnModDev70(void);            // Return TRUE if on MOD-DEV-70, otherwise FALSE
00306 void putleds(unsigned char c);    // Set the LEDs on the MOD-DEV-70/100
00307 unsigned char getleds();          // Get the LEDs state from CPLD, MOD-DEV-100 only
00308 void putdisp(unsigned short w);   // Set the 7-segment LEDs on the MOD-DEV-100, BCD input, CPLD
      controls segmnet switching
00309 unsigned short getdisp();         // Get the 7-segment LEDs on the MOD-DEV-100, BCD
00310 unsigned char getdipsw();         // Read the DIP switch values on the MOD-DEV-70/100
00311
00312 //--- The following functions are only valid on v1.12+ revisions of the dev board
00313 //--- CPLD programming files are available to upgrade older board revs
00314 // Set the 7-segment LEDs on the MOD-DEV-100
00315 void putSegments(unsigned char DisplayMask,
00316                  unsigned char DisplayData);   // Display Mask - 4bit value indicates which display
      shows value,  DisplayData - Segment bit
00317                                               // 7 is decimal point, bits 0-6 = Segments A-G
00318 void putDecimal(unsigned char DisplayMask);    // Set the 7-segment decimal point on the MOD-DEV-100,
      Display Mask - 4bit value indicates
00319                                               // which display has decimal point
00320 unsigned char getCPLDver();                   // returns the version # of the CPLD
00321 unsigned char getCPLDrev();                   // returns the revision # of the CPLD
00322
```

```
00323 //  Predefined segments for the putSegments display function
00324 #define SevenSeg_0 (0x3F)
00325 #define SevenSeg_1 (0x06)
00326 #define SevenSeg_2 (0x5B)
00327 #define SevenSeg_3 (0x4F)
00328 #define SevenSeg_4 (0x66)
00329 #define SevenSeg_5 (0x6D)
00330 #define SevenSeg_6 (0x7C)
00331 #define SevenSeg_7 (0x07)
00332 #define SevenSeg_8 (0x7F)
00333 #define SevenSeg_9 (0x67)
00334 #define SevenSeg_A (0x77)
00335 #define SevenSeg_B (0x7F)
00336 #define SevenSeg_C (0x39)
00337 #define SevenSeg_D (0x3F)
00338 #define SevenSeg_E (0x79)
00339 #define SevenSeg_F (0x71)
00340 #define SevenSeg_H (0x76)
00341 #define SevenSeg_I (0x06)
00342 #define SevenSeg_J (0x1E)
00343 #define SevenSeg_L (0x38)
00344 #define SevenSeg_O (0x3F)
00345 #define SevenSeg_P (0x73)
00346 #define SevenSeg_S (0x6D)
00347 #define SevenSeg_U (0x3E)
00348 #define SevenSeg_b (0x7C)
00349 #define SevenSeg_c (0x58)
00350 #define SevenSeg_d (0x5E)
00351 #define SevenSeg_h (0x74)
00352 #define SevenSeg_l (0x06)
00353 #define SevenSeg_n (0x54)
00354 #define SevenSeg_o (0x53)
00355 #define SevenSeg_r (0x50)
00356 #define SevenSeg_u (0x1C)
00357 #define SevenSeg_Dash (0x40)
00358 #define SevenSeg_Decimal (0x80)
00359 #define SevenSeg_OFF (0x00)
00360
00361 #endif /* #ifndef _NB_UTILS_H */
```

24.506 vjhc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef NB_VJHC_H
00005 #define NB_VJHC_H
00006
00007 #include <ip.h>
00008 const int VJHC_MAX_HDR = 128;       /* max TCP+IP hdr length (by protocol def) */
00009 const uint8_t VJHC_MAX_STATES = 16; /* must be >2 and <255 */
00010 /*
00011  * "state" data for each active tcp conversation on the wire.  This is
00012  * basically a copy of the entire IP/TCP header from the last packet together
00013  * with a small identifier the transmit & receive ends of the line use to
00014  * locate saved header.
00015  */
00016
00017 typedef struct
00018 {
00019     beuint16_t th_sport;
00020     beuint16_t th_dport;
00021     beuint32_t th_seq;
00022     beuint32_t th_ack;
00023     uint8_t th_len;
00024     uint8_t th_flags;
00025     beuint16_t th_win;
00026     beuint16_t th_sum;
00027     beuint16_t th_urp;
00028 } __attribute__((packed)) vjtcphdr;
00029
00030 inline vjtcphdr *GetVJTcpHdrFromIp(IPPKT *pIp)
00031 {
00032     uint32_t hlen = (pIp->bVerHdrLen & 0x0F);
00033     return (vjtcphdr *)(((puint32_t)(pIp)) + hlen);
00034 }
00035
00036 struct __attribute__((packed)) cstate
00037 {
00038     beuint16_t cs_hlen; /* size of hdr (receive only) */
00039     uint8_t cs_id;      /* connection # associated with this state */
00040     uint8_t cs_filler;
00041
00042     union __attribute__((packed))
00043     {
```

```
00044         IPPKT csu_ip; /* ip/tcp hdr from most recent packet */
00045         char extra[VJHC_MAX_HDR];
00046     };
00047
00048     void Clear();
00049     bool Match(IPPKT *pIp);
00050     vjtcphdr *GetHdr();
00051 };
00052
00053 #define cs_hdr (char *)&csu_ip
00054 #define cs_ip csu_ip
00055
00056 class slcompress
00057 {
00058    private:
00059     int last_tsn;      /* most recently used tstate */
00060     uint8_t last_recv; /* last rcvd conn. id */
00061     uint8_t last_xmit; /* last sent conn. id */
00062     beuint16_t flags;
00063     struct cstate tstate[VJHC_MAX_STATES]; /* xmit connection states */
00064     struct cstate rstate[VJHC_MAX_STATES]; /* receive connection states */
00065    public:
00066     volatile int NumStatesRX;
00067     volatile int NumStatesTX;
00068
00069     uint8_t compress_tcp(uint8_t *&pData, int &data_len, int compress_cid);
00070     PoolPtr uncompress_tcp(PoolPtr p, uint8_t *bufp, int len, uint32_t type);
00071     void init();
00072 };
00073
00074 #endif
```

24.507 http_funcs.h File Reference

JSON HTTP functions.

```
#include <nbrtos.h>
#include <nettypes.h>
#include <dns.h>
#include <webclient/web_buffers.h>
```

Classes

- class ParsedURI

  Parsed Uniform Resource Identifier Class (URI)

Functions

- void SetHttpDiag (bool b)

  Enable/disable Web Client HTTP diagnostics to the console port.
- int DoMultipartStartPost (ParsedURI &TheUri, const char *separator, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND, uint32_t contentLength=0)

  Start a multipart HTTP post using a pre-parsed URI object.
- int DoMultipartStartPost (const char *pUrl, const char *separator, uint16_t TIMEOUT_WAIT=10 *TICKS_↩ PER_SECOND, uint32_t contentLength=0)

  Start a multipart HTTP post using a pointer to a URL.
- void DoMultipartItem (int tcpfd, const char *Disposition, const char *separator, const unsigned char *data, int len)

  Send a multipart item.
- void DoMultipartBoundary (int tcpfd, const char *Disposition, const char *separator)

  Send a multipart boundary.
- bool DoMultipartFinished (int tcpfd, const char *separator, buffer_object &result_buffer, uint16_t TIMEOUT↩ _WAIT=10 *TICKS_PER_SECOND)

  Send a multipart item.
- bool DoUrlEncodedFormPost (ParsedURI &TheUri, char *headers, char *form_data, buffer_object &result↩ _buffer, uint16_t TIMEOUT_WAIT)

Post a JSON file using a HTTP Form Post and a a ParsedURI object.

- bool DoUrlEncodedFormPost (const char *pUrl, char *headers, char *form_data, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT)

  Post a JSON file using a HTTP POST and a URL string and pointer to JSON data.
- bool DoJsonPost (const char *pUrl, const char *Json_Data_To_Post, buffer_object &result_buffer, const char *AdditionalHeaders=NULL, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using a HTTP POST and a URL string and pointer to JSON data.
- bool DoJsonPost (ParsedURI &TheUri, const char *Json_Data_To_Post, buffer_object &result_buffer, const char *AdditionalHeaders=NULL, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using a HTTP POST and a ParsedURI object.
- bool DoJsonPost (const char *pUrl, ParsedJsonDataSet &jsonout, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using a HTTP POST and a URL string.
- bool DoJsonPost (ParsedURI &TheUri, ParsedJsonDataSet &jsonout, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using HTTP and a ParsedURI object.
- bool DoJsonPostHttpFile (const char *pUrl, const char *FragmentName, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using HTTP and a URL string.
- bool DoJsonPostHttpFile (ParsedURI &TheUri, const char *FragmentName, buffer_object &result_buffer, const char *AdditionalHeaders, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Post a JSON file using HTTP and a ParsedURI object.
- bool DoGet (ParsedURI &TheUri, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)
- bool DoGetEx (ParsedURI &TheUri, const char *headers, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)
- bool DoGet (const char *pUrl, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- bool DoGetEx (const char *pUrl, const char *headers, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- int DoGet (ParsedURI &TheUri, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a reference to a parsed Uniform Resource Identifier (URI)
- int DoGetEx (ParsedURI &TheUri, const char *headers, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a reference to a parsed Uniform Resource Identifier (URI)
- int DoGet (const char *pUrl, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- int DoGetEx (const char *pUrl, const char *headers, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a HTTP/HTTPS GET request using a pointer to a URL string.
- int DoGetUpdate (ParsedURI &TheUri, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a firmware update from the specified URI.
- int DoGetUpdate (const char *pUrl, uint16_t TIMEOUT_WAIT=10 *TICKS_PER_SECOND)

  Execute a firmware update from the specified URI.
- int PopulateAuthHeader (const char *user, const char *password, char *buffer, int maxlen)

  Fill in a username and password into a buffer for use as an extra header.

24.507.1 Detailed Description

JSON HTTP functions.
These Web Client functions can be used with TCP, or with TLS WEB_CLIENT_SSL_SUPPORT is enabled.

24.508 http_funcs.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00011 #ifndef _HTTP_FUNCS_H
00012 #define _HTTP_FUNCS_H
00013
00014 #include <nbrtos.h>
00015 #include <nettypes.h>
00016 #include <dns.h>
00017 #include <webclient/web_buffers.h>
00018
00035 class ParsedURI
00036 {
00037 public:
00038     enum eProtocol_t {
00039         eProto_None,
00040         eProto_HTTP,
00041         eProto_HTTPS,
00042         eProto_SSH,
00043         eProto_SMTP,
00044         eProto_MQTT,
00045         eProto_MQTTS,
00046         eProto_WS,
00047         eProto_WSS
00048     };
00049 private:
00050
00051     char URIBuf[256];
00052     const char *CachedProtoStr;
00053     char *CachedHost;
00054     char *CachedPath;
00055     IPADDR CachedHostIp;
00056     tick_t ValidUntil;
00057     uint16_t CachedPort;
00058     eProtocol_t CachedProto;
00059     uint8_t len;
00060
00061     #ifdef WEB_CLIENT_SSL_SUPPORT
00062         bool secure;
00063     #endif
00064
00065     int  advanceCompareState(int state);
00066     bool equals(const char *pUrl);
00067     bool equals(const ParsedURI &rhs);
00068     static ParsedURI *getParsedURI(OS_FIFO *pFifo);
00069    public:
00079         void NewUri(const char *uri, uint16_t timeout = 20 * TICKS_PER_SECOND, bool skipLookup =
      false);
00080
00081
00085         ParsedURI();
00086
00094         ParsedURI(const char *uri, uint16_t timeout = 20 * TICKS_PER_SECOND, bool skipLookup = false)
00095         {
00096             NewUri(uri, timeout);
00097         };
00098
00099
00106         bool valid() const
00107         {
00108             return ((ValidUntil == 0xFFFFFFFF)|| (ValidUntil &&(((uint32_t)((int32_t)ValidUntil -
      ((int32_t)TimeTick))) >= 0)))&& URIBuf[0];
00109         };
00110
00111
00117         const char *GetPath()
00118         {
00119                 if(CachedPath) return CachedPath;
00120                 return "";
00121         };
00122
00123
00129         const char *GetHost()
00130         {
```

```
00131                   return CachedHost;
00132           };
00133
00134
00140           IPADDR GetAddr()
00141           {
00142               if (valid())
00143                   return CachedHostIp;
00144               else
00145                   return IPADDR4::NullIP();
00146           };
00147
00148
00154           uint16_t GetPort()
00155           {
00156                   return CachedPort;
00157           };
00158
00159           eProtocol_t GetProto()
00160           {
00161                   return CachedProto;
00162           }
00163
00164           #ifdef WEB_CLIENT_SSL_SUPPORT
00171               bool IsSecure()
00172               {
00173                   return secure;
00174               }
00175           #endif
00176           void dump();
00177
00181           void Invalidate() { ValidUntil = 0;  CachedHostIp.SetNull(); };
00182           void Resolve(bool block = true, uint16_t timeout = 20 * TICKS_PER_SECOND);
00183
00184           //Get an FD from UDP rx and send out requests...
00185           int start_fd_dns();
00186           bool process_fd_dns(int fd);
00187
00188           ParsedURI & operator=(const char *rhs) { NewUri(rhs); return *this; }
00189           ParsedURI & operator=(const ParsedURI &rhs);
00190           bool operator==(const ParsedURI &rhs) { return equals(rhs); }
00191           bool operator!=(const ParsedURI &rhs) { return !equals(rhs); }
00192 };
00193
00194
00200 void SetHttpDiag(bool b);
00201
00202
00203 #ifndef WebErrorReporterFunc
00204 typedef void(WebErrorReporterFunc)(int ErrorState);
00205 #endif
00206
00207 extern WebErrorReporterFunc *pWebErrorReporter;
00208
00209
00210 class ParsedJsonDataSet;   // Forward declaration
00211
00212
00223 int DoMultipartStartPost(ParsedURI &TheUri,
00224                          const char *separator,
00225                          uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND,
00226                          uint32_t contentLength = 0);
00227
00238 int DoMultipartStartPost(const char *pUrl,
00239                          const char *separator,
00240                          uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND,
00241                          uint32_t contentLength = 0);
00242
00252 void DoMultipartItem(int tcpfd, const char *Disposition, const char *separator,const unsigned char *
      data, int len);
00253
00254
00262 void DoMultipartBoundary(int tcpfd, const char *Disposition, const char *separator);
00263
00264
00275 bool DoMultipartFinished(int tcpfd, const char *separator, buffer_object &result_buffer, uint16_t
      TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00276
00288 bool DoUrlEncodedFormPost(  ParsedURI & TheUri,
00289                             char * headers,
00290                             char * form_data,
00291                             buffer_object & result_buffer,
00292                             uint16_t TIMEOUT_WAIT);
00293
00305 bool DoUrlEncodedFormPost(  const char * pUrl,
00306                             char * headers,
00307                             char * form_data,
```

```
00308                           buffer_object & result_buffer,
00309                           uint16_t TIMEOUT_WAIT);
00310
00322 bool DoJsonPost(const char *pUrl,
00323                 const char *Json_Data_To_Post,
00324                 buffer_object &result_buffer,
00325                 const char *AdditionalHeaders = NULL,
00326                 uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00327
00339 bool DoJsonPost(ParsedURI &TheUri,
00340                 const char *Json_Data_To_Post,
00341                 buffer_object &result_buffer,
00342                 const char *AdditionalHeaders = NULL,
00343                 uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00344
00356 bool DoJsonPost(const char *pUrl,
00357                 ParsedJsonDataSet &jsonout,
00358                 buffer_object &result_buffer,
00359                 const char *AdditionalHeaders,
00360                 uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00361
00373 bool DoJsonPost(ParsedURI &TheUri,
00374                 ParsedJsonDataSet &jsonout,
00375                 buffer_object &result_buffer,
00376                 const char *AdditionalHeaders,
00377                 uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00378
00379 // Do a JSON post using a JSON file template with function file variables call backs etc... like the
    webserver
00391 bool DoJsonPostHttpFile(const char *pUrl,
00392                         const char *FragmentName,
00393                         buffer_object &result_buffer,
00394                         const char *AdditionalHeaders,
00395                         uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00396
00408 bool DoJsonPostHttpFile(ParsedURI &TheUri,
00409                         const char *FragmentName,
00410                         buffer_object &result_buffer,
00411                         const char *AdditionalHeaders,
00412                         uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00413
00423 bool DoGet(ParsedURI &TheUri, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT = 10 *
    TICKS_PER_SECOND);
00424
00435 bool DoGetEx(ParsedURI &TheUri,const char * headers, buffer_object &result_buffer, uint16_t
    TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00436
00437
00449 bool DoGet(const char *pUrl, buffer_object &result_buffer, uint16_t TIMEOUT_WAIT = 10 *
    TICKS_PER_SECOND);
00450
00463 bool DoGetEx(const char *pUrl,const char * headers,buffer_object &result_buffer, uint16_t TIMEOUT_WAIT
    = 10 * TICKS_PER_SECOND);
00464
00465
00466
00480 int DoGet(ParsedURI &TheUri, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT = 10 *
    TICKS_PER_SECOND);
00481
00496 int DoGetEx(ParsedURI &TheUri, const char* headers,unsigned char *result, int maxl, uint16_t
    TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00497
00498
00511 int DoGet(const char *pUrl, unsigned char *result, int maxl, uint16_t TIMEOUT_WAIT = 10 *
    TICKS_PER_SECOND);
00512
00526 int DoGetEx(const char *pUrl,const char * headers,unsigned char *result, int maxl, uint16_t
    TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00527
00528
00529
00530 //Turn on update diagnostics
00531 void SetHttpUpDiag(bool b);
00532
00533
00542 int DoGetUpdate(ParsedURI &TheUri, uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00543
00552 int DoGetUpdate(const char *pUrl, uint16_t TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00553
00554
00555
00566 int PopulateAuthHeader(const char * user, const char * password, char * buffer, int maxlen);
00567
00568 /* Parse the uri and open tcpp to the correct host/port nothing is sent*/
00569 int StartConnection(ParsedURI &TheUri,TickTimeout & TT);
00570 inline int StartConnection(ParsedURI &TheUri,uint32_t timeout=TICKS_PER_SECOND*10)
00571 {
```

```
00572 TickTimeout tt(timeout);
00573 return StartConnection(TheUri,tt);
00574 }
00575
00576
00577
00578
00579 /* Returns an FD otherwise starts a transaction and sendsheaders...
00580 including the host: heder type.
00581 headers should be formatted without leading and trailing \r\n
00582 */
00583 int StartTransaction(ParsedURI &TheUri, const char *headers = 0, const char *action = "GET", uint32_t
      TIMEOUT_WAIT = 10 * TICKS_PER_SECOND);
00584 int StartTransaction(const char * pUrl, const char * headers=0,const char * action="GET", uint32_t
      TIMEOUT_WAIT= 10 * TICKS_PER_SECOND);
00585
00586
00587
00593 #endif
```

24.509 web_buffers.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _WEB_BUFFERS_H
00006 #define _WEB_BUFFERS_H
00007
00008 class buffer_object
00009 {
00010    public:
00011     // A write of ZERO bytes indicates end of data stream
00012     virtual int WriteData(const unsigned char *pCopyFrom, int num_bytes) = 0;
00013     virtual int ReadFrom(int fd) = 0;
00014     virtual void ProcessHeader(const char * hdr) {};
00015 };
00016
00017 class SimpleBufferObject : public buffer_object
00018 {
00019     unsigned char *cp_to;
00020     int space_left;
00021     int used;
00022    public:
00023
00024     virtual int WriteData(const unsigned char *pCopyFrom, int num_bytes);
00025     virtual int ReadFrom(int fd);
00026
00027     SimpleBufferObject(unsigned char *p, int len)
00028     {
00029         cp_to = p;
00030         space_left = len;
00031         used = 0;
00032     };
00033     int AmountUsed() { return used; };
00034 };
00035
00036 #endif
```

24.510 web_client.h File Reference

Web Client.

```
#include <nbstring.h>
#include <webclient/http_funcs.h>
```

Macros

- #define **WEB_CLIENT_ERROR_NO_ETHERNET** (1)

  No Ethernet connection.

- #define **WEB_CLIENT_ERROR_NO_ADDRESS** (2)

  Interface has no IP address.

- #define **WEB_CLIENT_ERROR_NO_GATEWAY** (3)

  Interface has no gateway IP address.

- #define **WEB_CLIENT_ERROR_GATEWAY_WRONG** (4)

  Interface has incorrect gateway IP address.
- #define **WEB_CLIENT_ERROR_NO_DNS_ADDR** (5)

  Interface has a DNS errror.
- #define **WEB_CLIENT_ERROR_NO_DNS_RESOLVE** (6)

  DNS was not able to resolve the URL name.
- #define **WEB_CLIENT_ERROR_NO_NTP** (7)

  NTP failed.
- #define **WEB_CLIENT_ERROR_NO_SERVER_RESPONSE** (8)

  No response from server.
- #define **WEB_CLIENT_ERROR_NO_SERVER_CONNECT** (9)

  Unable to connect to server.
- #define **WEB_CLIENT_ERROR_NO_ERROR** (10)

  No error.

Functions

- bool StartWebClient (int prio, const char *url1, const char *url2=NULL, bool bDoNtp=false)

  Start the web client using a URL string.
- bool StartWebClient (int prio, const NBString &url1, const NBString &url2, bool bDoNtp=false)

  Start the web client using a NBString.
- bool StartWebClient (int prio, const NBString &url1, bool bDoNtp=false)

  Start the web client using a NBString.

24.510.1 Detailed Description

Web Client.

24.511 web_client.h

Go to the documentation of this file.

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00009 #ifndef _WEB_CLIENT_H
00010 #define _WEB_CLIENT_H
00011
00017 #include <nbstring.h>
00018 #include <webclient/http_funcs.h>
00019
00024 #define WEB_CLIENT_ERROR_NO_ETHERNET (1)
00025 #define WEB_CLIENT_ERROR_NO_ADDRESS (2)
00026 #define WEB_CLIENT_ERROR_NO_GATEWAY (3)
00027 #define WEB_CLIENT_ERROR_GATEWAY_WRONG (4)
00028 #define WEB_CLIENT_ERROR_NO_DNS_ADDR (5)
00029 #define WEB_CLIENT_ERROR_NO_DNS_RESOLVE (6)
00030 #define WEB_CLIENT_ERROR_NO_NTP (7)
00031 #define WEB_CLIENT_ERROR_NO_SERVER_RESPONSE (8)
00032 #define WEB_CLIENT_ERROR_NO_SERVER_CONNECT (9)
00033 #define WEB_CLIENT_ERROR_NO_ERROR (10)
00036 #define WEB_CLIENT_ERROR_LAST_STATE (10)
00037
00038 extern const char *web_error_state_text[WEB_CLIENT_ERROR_LAST_STATE + 1];
00039
00040 #ifndef WebErrorReporterFunc
00041 typedef void(WebErrorReporterFunc)(int ErrorState);
00042 #endif
00043
00044 extern WebErrorReporterFunc *pWebErrorReporter;
00045
00046 bool DoActualClientRequest(ParsedURI &TheUri, uint16_t &next_time_delay);
00047
00058 bool StartWebClient(int prio, const char *url1, const char *url2 = NULL, bool bDoNtp = false);
00059
00070 bool StartWebClient(int prio, const NBString &url1, const NBString &url2, bool bDoNtp = false);
00071
```

```
00081 bool StartWebClient(int prio, const NBString &url1, bool bDoNtp = false);
00082
00083 // Semaphore to wake web client early
00084 extern OS_SEM WebClientSem;
00085
00088 #endif
```

24.512 websockets.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __WEBSOCKETS_H
00006 #define __WEBSOCKETS_H
00007
00008 // NB Definitions
00009 #include <predef.h>
00010
00011 // NB Libs
00012 #include <buffers.h>
00013 #include <http.h>
00014 #include <iosys.h>
00015 #include <nbrtos.h>
00016 #include <nettypes.h>
00017
00018 #define WS_BUFFER_SEGMENTS 4
00019 #define WS_MAX_SOCKS 4
00020 #define WS_BUFFER_MIN 10
00021 #define WS_ACCUM_DLY ((TICKS_PER_SECOND / 4) + 1)
00022 #define WS_FLUSH_TIMEOUT 0
00023
00024 #define WS_FIN_BIT 0x80
00025
00026 #define WS_OP_CONT 0x0
00027 #define WS_OP_TEXT 0x1
00028 #define WS_OP_BIN 0x2
00029 #define WS_OP_CLOSE 0x8
00030 #define WS_OP_PING 0x9
00031 #define WS_OP_PONG 0xA
00032
00033 #define WS_SO_TEXT 0x8
00034
00035 namespace NB
00036 {
00037 class WebSocket
00038 {
00039     struct WS_Client_Short
00040     {
00041         uint8_t opAndFin;
00042         uint8_t maskAndLen;
00043         uint32_t mask;
00044         uint8_t pData[];
00045     } __attribute__((packed));
00046     struct WS_Client_Med
00047     {
00048         uint8_t opAndFin;
00049         uint8_t maskAndLen;
00050         beuint16_t length;
00051         uint32_t mask;
00052         uint8_t pData[];
00053     } __attribute__((packed));
00054     struct WS_Client_Long
00055     {
00056         uint8_t opAndFin;
00057         uint8_t maskAndLen;
00058         beuint32_t lengthHi;
00059         beuint32_t lengthLo;
00060         uint32_t mask;
00061         uint8_t pData[];
00062     } __attribute__((packed));
00063     struct WS_Server_Short
00064     {
00065         uint8_t opAndFin;
00066         uint8_t maskAndLen;
00067         uint8_t pData[];
00068     } __attribute__((packed));
00069     struct WS_Server_Med
00070     {
00071         uint8_t opAndFin;
00072         uint8_t maskAndLen;
00073         beuint16_t length;
00074         uint8_t pData[];
00075     } __attribute__((packed));
00076     struct WS_Server_Long
```

```
00077     {
00078         uint8_t opAndFin;
00079         uint8_t maskAndLen;
00080         beuint32_t lengthHi;
00081         beuint32_t lengthLo;
00082         uint8_t pData[];
00083     } __attribute__((packed));
00084
00085     enum WS_State
00086     {
00087         WS_INIT,
00088         WS_CONNECTING,
00089         WS_ESTABLISHED,
00090         WS_CLOSE_PENDING,
00091         WS_CLOSE_SENT,
00092         WS_CLOSE_RECV,
00093         WS_TCP_CLOSED,
00094         WS_CLOSED,
00095         WS_ABORT,
00096         WS_FAIL
00097     };
00098
00099     static const char *WS_StateStr(WS_State state);
00100     enum WS_StatusCode
00101     {
00102         WS_STAT_NORM_CLOSE = 1000,
00103         WS_STAT_GOING_AWAY = 1001,
00104         WS_STAT_PROT_ERROR = 1002,
00105         WS_STAT_UNACCEPT_TYPE = 1003,
00106         WS_STAT_NONE = 1005,            // Must not be sent
00107         WS_STAT_NONE_ABNORMAL = 1006,   // Must not be sent
00108         WS_STAT_NOT_CONSISTENT = 1007,
00109         WS_STAT_POLICY_VIOLATION = 1008,
00110         WS_STAT_MSG_TOO_BIG = 1009,
00111         WS_STAT_EXPECTED_EXTENSION = 1010,
00112         WS_STAT_UNEXPECTED_COND = 1011,
00113         WS_STAT_TLS_FAILURE = 1015,   // Must not be sent
00114     };
00115
00116     fifo_buffer_storage rxBuffer;
00117     fifo_buffer_storage ctlBuffer;
00118     fifo_buffer_storage txBuffer;
00119
00120   public:
00121     int m_fd_tcp;
00122     int m_fd_ws;
00123     OS_CRIT socket_crit;
00124
00125   private:
00126     WS_State m_state;
00127     int m_options;
00128     uint32_t m_SendTime;
00129     uint32_t m_AccumDly;
00130     bool m_serverNotClient;
00131
00132     volatile bool m_CtlBlock;
00133     uint32_t m_currentIndex;
00134     uint32_t m_remLen;
00135
00136     uint8_t m_frameHeadBuf[14];
00137     uint8_t m_frameHeadNext;
00138     bool m_gotLen;
00139     uint32_t m_currentMask;
00140     uint32_t m_remainingLen;
00141
00142     uint8_t m_txFrameHeadBuf[14];
00143     uint8_t m_txFrameHeadNext;
00144     uint8_t m_txFrameHeadLen;
00145     uint32_t m_txCurrMask;
00146     uint32_t m_txRemLen;
00147     bool m_txFlushInProgress;
00148     bool m_txCtlSendInProgress;
00149     OS_SEM *m_ctlWaitSem;
00150
00151     uint32_t m_pingTick;
00152     int      m_pingLen;
00153     OS_SEM  *m_pingWaitSem;
00154
00155     uint32_t m_txBufferMin;
00156
00157     void ServerRxFromTCP();
00158     bool ReadRemainingHeader();
00159     void ProcessCtlFrame();
00160     bool ValidatePong(uint32_t seed, uint32_t len);
00161     void Close(bool closedAfterSend, uint16_t code = 0, const char *reason = NULL, int len = 0);
00162     void RemoteClose();
00163     void CleanUp();
```

```
00164     void SendCtlFrame();
00165     void Init(int fd_tcp, int fd_ws, bool serverNotClient);
00166     void Flush_Header(uint8_t *buf, uint32_t mask, uint32_t len);
00167
00168     static uint32_t s_flushTick;
00169     static fd_set s_pendingCtlSocks;
00170     static OS_CRIT s_ws_crit_obj;
00171     static bool s_bFinishedInit;
00172
00173     static void StaticInit();
00174     static int GetNewSocket(int fd_tcp, bool serverNotClient);
00175
00176     static WebSocket *GetRecordFromTCP(int fd);
00177
00178     static int CoreConnect(IPADDR ip, const char *host, const char *resource, const char *protocol,
      int portnum, bool useSSL = false);
00179
00180   public:
00181     int WriteData(const char *buf, int nbytes);
00182     int ReadData(char *buf, int nbytes);
00183     void Flush();
00184     void Pong(uint32_t len);
00185     inline WS_State GetState() { return m_state; }
00186     int Ping(uint32_t len, uint32_t *sentTick);
00187     int WaitForPingReply(uint32_t timeout);
00188     int GetPingReplyTick(uint32_t *replyTick);
00189
00190     int setoption(int option);
00191     int clroption(int option);
00192     int getoption();
00193
00194     inline int GetWriteSpace() { return txBuffer.SpaceAvail(); }
00195
00196     static int s_openSocketCount;   // really should be private, but can't figure how
00197     static WebSocket WSSockStructs[WS_MAX_SOCKS];
00198
00199     static int ws_readwto(int fd, char *buf, int nbytes, int timeout);
00200     static int ws_read(int fd, char *buf, int nbytes);
00201     static int ws_write(int fd, const char *buf, int nbytes);
00202     static int ws_externalclose(int fd);
00203     static void ws_flush(int fd);
00204     static int ws_setoption(int fd, int option);
00205     static int ws_clroption(int fd, int option);
00206     static int ws_getoption(int fd);
00207
00208     static WebSocket *GetWebSocketRecord(int fd);
00209     static int promote_tcp_ws(int tcp_fd);
00210     static void ws_read_notify(int fd);    // tcp data rx callback
00211     static void ws_write_notify(int fd);   // tcp data tx space callback
00212     static void GetFlushTime();
00213
00214     static void RunSkippedCallBack();
00215
00216     static int Connect(const char * host, const char * resource, const char * protocol, int portnum =
      80, bool useSSL = false);
00217     static inline int Connect(const char * host, const char * resource, int portnum = 80, bool useSSL
      = false)
00218     { return Connect(host, resource, NULL, portnum, useSSL); }
00219     static int Connect(IPADDR host, const char * resource, const char * protocol, int portnum = 80,
      bool useSSL = false);
00220     static inline int Connect(IPADDR host, const char * resource, int portnum = 80, bool useSSL =
      false)
00221     { return Connect(host, resource, NULL, portnum, useSSL); }
00222
00223     void DumpSock();
00224     static void DumpSockets();
00225     static int GetFreeWebSocketCount();
00226 };
00227
00228 }   // namespace NB
00229
00230 int WSPing(int fd, uint32_t len, uint32_t *sentTick);
00231 int WSGetPingReplyTick(int fd, uint32_t *replyTick);
00232 int WSWaitForPingReply(int fd, uint32_t timeout);
00233 int WSUpgrade(HTTP_Request *req, int sock);
00234 #endif /* ----- #ifndef __WEBSOCKETS_H  ----- */
```

24.513 bsp.h File Reference

Low level hardware functions for the MOD5441x platform.

Functions

- void SpreadSpectrumOscillator (bool enable)

  Enable spread spectrum oscialltor mode on the CPU clock.

24.513.1 Detailed Description

Low level hardware functions for the MOD5441x platform.

24.513.2 Function Documentation

24.513.2.1 SpreadSpectrumOscillator()

```
void SpreadSpectrumOscillator (
            bool enable )
```

Enable spread spectrum oscialltor mode on the CPU clock.
The 25MHz CPU oscillator defaults to standard mode, but can be put in a spread spectrum mode that will help with radiated emissions. This will reduce EMI, but will cause the clock to have some jitter.
When enabled, the oscillator center frequency is reduced by 2% to avoid overclocking the CPU. The frequency will then vary by +/- 2%. This means the average clock rate will be reduced by 1% when this feature is enabled. To compensate, the system CPU_CLOCK variable and system time tick rate aer modified to reflect the new bus clock speed. Finally, the console/debug serial port will be closed and reopened to recalculate the baud rate.

Warning

Any periperhals or clock dependent sections of your application must take this adjustment into consideration. We recommend calling this function in the very beginning of the application to avoid baud rate miscalculations.

Parameters

enable	Set to true to enable spread spectrum mode

24.514 MOD5441X/include/bsp.h

Go to the documentation of this file.

```
00001
00005 #ifndef __NB_BSP_H
00006 #define __NB_BSP_H
00007
00008 // The ratio of the average clock rate in spread spectrum mode compared to the original clock rate
00009 #define SPREAD_SPECTRUM_OSC_CLK_RATIO (0.98925)
00010 #define CPU_CLOCK_SS_DISABLED (250000000)
00011 #define CPU_CLOCK_SS_ENABLED (247312500)
00012
00037 void SpreadSpectrumOscillator(bool enable);
00038
00039 #endif
```

24.515 bsp.h File Reference

Hardware functions that are unique to a specific platform.

```
#include <cpu_pins.h>
#include <stdio.h>
#include <component/pmc.h>
```

Classes

- class UniqueIdentifier

  Get the 128-bit Unique Identifier.

Functions

- void EnableExtBusBuff (bool enable)

  Enable External Bus Interface Buffers.
- uint32_t SetPLL (uint8_t multiplier)

  Sets PLL to adjust system clock speed, primarily used for power reduction.
- uint32_t SetMCKDivider (uint8_t divider)

  Sets Master Clock prescaler (PRES) in the MCKR register to adjust system clock speed. Primarily used for power reduction.
- void DrivePCK (uint8_t prescaler=1, uint8_t clkSource=PMC_PCK_CSS_MCK)

  Enables external CPU clock signal on J1[31].
- void DisablePCK ()

  Disables external CPU clock signal on J1[31].

24.515.1 Detailed Description

Hardware functions that are unique to a specific platform.

24.515.2 Function Documentation

24.515.2.1 DisablePCK()

```
void DisablePCK ( )
```

Disables external CPU clock signal on J1[31].

24.515.2.2 DrivePCK()

```
void DrivePCK (
            uint8_t prescaler = 1,
            uint8_t clkSource = PMC_PCK_CSS_MCK )
```

Enables external CPU clock signal on J1[31].

Parameters

prescaler	The prescaler to apply to the clock source PCK generation
clkSource	The clock source used to generate the PCK clock. See pmc.h for options

24.515.2.3 EnableExtBusBuff()

```
void EnableExtBusBuff (
            bool enable )
```

Enable External Bus Interface Buffers.
When enabled the following singnals will always be driven as outputs on the P1 header: A0/NSB0, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A13, A14, A16/BA0, NSB1 The data buffer is driven when the TIP signal is active, which is also dependant on this function to be enabled. When enabled the following singnals will be driven in the direction controlled by the NRD EBI signal: D0-D15

When disabled, all buffered signals on the P1 header will be HiZ. This will reduce EMI, power consumption and allow for NRD to be used as a GPIO

Parameters

enable	true to enable external bus buffers

24.515.2.4 SetMCKDivider()

```
uint32_t SetMCKDivider (
            uint8_t divider )
```

Sets Master Clock prescaler (PRES) in the MCKR register to adjust system clock speed. Primarily used for power reduction.
Please refer to the ATSAME70 processor reference manual Clock Generator chapter for complete limitations of setting the PLL to avoid potentially damaging the processor
The calculation used to determine CPU speed is: CPU_CLOCK = (OSC_CLOCK $*$ PLL multiplier) / divider (PRES) The default MODM7AE70 module has a default PLL speed (OSC_CLOCK $*$ PLL multiplier) of 300MHz

- Configurations handled by this function

  1. Adjusts prescaler for Master Clk in the MCKR register
  2. If new CPU Clock is 150MHz or LOWER then the Peripheral to CPU Clock ratio is set to 1:1
  3. Sets new CPU_CLOCK and PERIPH_CLOCK variables
     These are used in many baud rate calculations in NetBurner API, for example serial ports
  4. Reinitializes the system tick so TimeTicks are accurate
  5. Adjusts Flash memory timings
  6. Adjust SDRAM refresh timings
  7. Closes and reopens default serial port to adjust baud

Parameters

divider	Divider for CPU_CLOCK calculation. Sets PRES variable in MCKR register. Only 1,2,3,4,8,16,32,64 are valid, other values have no affect.

Returns

The current CPU clock frequency

24.515.2.5 SetPLL()

```
uint32_t SetPLL (
            uint8_t multiplier )
```

Sets PLL to adjust system clock speed, primarily used for power reduction.
Please refer to the ATSAME70 processor reference manual Clock Generator chapter for complete limitations of setting the PLL to avoid potentially damaging the processor
The calculation used to determine CPU speed is: CPU_CLOCK = OSC_CLOCK $*$ mult The default MODM7AE70 module has a 12000000hz OSC_CLOCK The function will do nothing if new frequency is $>$300MHz or $<$160Mhz due to electrical specifications The only exception to this rule is setting the PLL multiplier to 1 or 0 which will disable it

- Configurations handled by this function

  1. Adjusts PLL configurations for Main Clk

2. Sets CPU_CLOCK and PERIPH_CLOCK variables
   These are used in many baud rate calculations in NetBurner API, for example serial ports
3. Reinitializes the system tick so TimeTicks are accurate
4. Adjusts Flash memory timings
5. Adjust SDRAM refresh timings
6. Closes and reopens default serial port to adjust baud

Parameters

multiplier	Multiplier for CPU_CLOCK calculation. Only 0,1,14-25 are valid. Set to 1 or 0 to disable PLL

Returns

The current CPU clock frequency

24.516 MODM7AE70/include/bsp.h

Go to the documentation of this file.

```
00001
00005 #ifndef __NB_BSP_H
00006 #define __NB_BSP_H
00007
00008 #include <cpu_pins.h>
00009 #include <stdio.h>
00010 #include <component/pmc.h>
00011 extern PinIO E70_LED;
00012
00021 class UniqueIdentifier
00022 {
00023     uint8_t val[16];
00024
00025   public:
00032     const uint8_t *GetUniqueIdentifier();
00033
00037     UniqueIdentifier() { GetUniqueIdentifier(); }
00038
00044     inline const uint8_t *GetBuffer() { return val; }
00045
00049     inline void Print()
00050     {
00051         iprintf("0x");
00052         for (uint8_t index = 0; index < 16; index++)
00053         {
00054             iprintf("%02x", val[index]);
00055         }
00056         iprintf("\r\n");
00057     }
00058 };
00059
00077 void EnableExtBusBuff(bool enable);
00078
00079 class ExtBusEnableCtx
00080 {
00081   public:
00082     static volatile uint32_t depth;
00083     ExtBusEnableCtx();
00084     ~ExtBusEnableCtx();
00085     void dump();
00086 };
00087
00115 uint32_t SetPLL(uint8_t multiplier);
00116
00144 uint32_t SetMCKDivider(uint8_t divider);
00145
00153 void DrivePCK(uint8_t prescaler = 1, uint8_t clkSource = PMC_PCK_CSS_MCK);
00154
00159 void DisablePCK();
00160
00161 #endif /* ----- #ifndef __CPU_HAL_H  ----- */
```

24.517 bsp.h File Reference

Hardware functions that are unique to a specific platform.

```
#include <cpu_pins.h>
#include <stdio.h>
#include <component/pmc.h>
```

Classes

- class UniqueIdentifier

  Get the 128-bit Unique Identifier.

Functions

- void EnableExtBusBuff (bool enable)

  Enable External Bus Interface Buffers.
- uint32_t SetPLL (uint8_t multiplier)

  Sets PLL to adjust system clock speed, primarily used for power reduction.
- uint32_t SetMCKDivider (uint8_t divider)

  Sets Master Clock prescaler (PRES) in the MCKR register to adjust system clock speed. Primarily used for power reduction.
- void DrivePCK (uint8_t prescaler=1, uint8_t clkSource=PMC_PCK_CSS_MCK)

  Enables external CPU clock signal on J1[31].
- void DisablePCK ()

  Disables external CPU clock signal on J1[31].

24.517.1 Detailed Description

Hardware functions that are unique to a specific platform.

24.517.2 Function Documentation

24.517.2.1 DisablePCK()

```
void DisablePCK ( )
```

Disables external CPU clock signal on J1[31].

24.517.2.2 DrivePCK()

```
void DrivePCK (
            uint8_t prescaler = 1,
            uint8_t clkSource = PMC_PCK_CSS_MCK )
```

Enables external CPU clock signal on J1[31].

Parameters

prescaler	The prescaler to apply to the clock source PCK generation
clkSource	The clock source used to generate the PCK clock. See pmc.h for options

24.517.2.3 EnableExtBusBuff()

```
void EnableExtBusBuff (
            bool enable )
```

Enable External Bus Interface Buffers.
When enabled the following singnals will always be driven as outputs on the P1 header: A0/NSB0, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A13, A14, A16/BA0, NSB1 The data buffer is driven when the TIP signal is active, which is also dependant on this function to be enabled. When enabled the following singnals will be driven in the direction controlled by the NRD EBI signal: D0-D15
When disabled, all buffered signals on the P1 header will be HiZ. This will reduce EMI, power consumption and allow for NRD to be used as a GPIO

Parameters

enable	true to enable external bus buffers

24.517.2.4 SetMCKDivider()

```
uint32_t SetMCKDivider (
            uint8_t divider )
```

Sets Master Clock prescaler (PRES) in the MCKR register to adjust system clock speed. Primarily used for power reduction.
Please refer to the ATSAME70 processor reference manual Clock Generator chapter for complete limitations of setting the PLL to avoid potentially damaging the processor
The calculation used to determine CPU speed is: CPU_CLOCK = (OSC_CLOCK ∗ PLL multiplier) / divider (PRES)
The default SBE70LC module has a default PLL speed (OSC_CLOCK ∗ PLL multiplier) of 300MHz

- Configurations handled by this function

  1. Adjusts prescaler for Master Clk in the MCKR register
  2. If new CPU Clock is 150MHz or LOWER then the Peripheral to CPU Clock ratio is set to 1:1
  3. Sets new CPU_CLOCK and PERIPH_CLOCK variables
     These are used in many baud rate calculations in NetBurner API, for example serial ports
  4. Reinitializes the system tick so TimeTicks are accurate
  5. Adjusts Flash memory timings
  6. Adjust SDRAM refresh timings
  7. Closes and reopens default serial port to adjust baud

Parameters

divider	Divider for CPU_CLOCK calculation. Sets PRES variable in MCKR register. Only 1,2,3,4,8,16,32,64 are valid, other values have no affect.

Returns

The current CPU clock frequency

24.517.2.5 SetPLL()

```
uint32_t SetPLL (
            uint8_t multiplier )
```

Sets PLL to adjust system clock speed, primarily used for power reduction.

Please refer to the ATSAME70 processor reference manual Clock Generator chapter for complete limitations of setting the PLL to avoid potentially damaging the processor
The calculation used to determine CPU speed is: CPU_CLOCK = OSC_CLOCK $*$ mult The default SBE70LCV module has a 12000000hz OSC_CLOCK The function will do nothing if new frequency is $>$300MHz or $<$160Mhz due to electrical specifications The only exception to this rule is setting the PLL multiplier to 1 or 0 which will disable it

- Configurations handled by this function

  1. Adjusts PLL configurations for Main Clk
  2. Sets CPU_CLOCK and PERIPH_CLOCK variables
     These are used in many baud rate calculations in NetBurner API, for example serial ports
  3. Reinitializes the system tick so TimeTicks are accurate
  4. Adjusts Flash memory timings
  5. Adjust SDRAM refresh timings
  6. Closes and reopens default serial port to adjust baud

Parameters

multiplier	Multiplier for CPU_CLOCK calculation. Only 0,1,14-25 are valid. Set to 1 or 0 to disable PLL

Returns

The current CPU clock frequency

24.518 SBE70LC/include/bsp.h

Go to the documentation of this file.

```
00001
00005 #ifndef __NB_BSP_H
00006 #define __NB_BSP_H
00007
00008 #include <cpu_pins.h>
00009 #include <stdio.h>
00010 #include <component/pmc.h>
00011 extern PinIO E70_LED;
00012
00021 class UniqueIdentifier
00022 {
00023     uint8_t val[16];
00024
00025    public:
00032     const uint8_t *GetUniqueIdentifier();
00033
00037     UniqueIdentifier() { GetUniqueIdentifier(); }
00038
00044     inline const uint8_t *GetBuffer() { return val; }
00045
00049     inline void Print()
00050     {
00051         iprintf("0x");
00052         for (uint8_t index = 0; index < 16; index++)
00053         {
00054             iprintf("%02x", val[index]);
00055         }
00056         iprintf("\r\n");
00057     }
00058 };
00059
00077 void EnableExtBusBuff(bool enable);
00078
00079 class ExtBusEnableCtx
00080 {
00081    public:
00082     static volatile uint32_t depth;
00083     ExtBusEnableCtx();
00084     ~ExtBusEnableCtx();
00085     void dump();
00086 };
```

```
00087
00115 uint32_t SetPLL(uint8_t multiplier);
00116
00144 uint32_t SetMCKDivider(uint8_t divider);
00145
00153 void DrivePCK(uint8_t prescaler = 1, uint8_t clkSource = PMC_PCK_CSS_MCK);
00154
00159 void DisablePCK();
00160
00161 #endif /* ----- #ifndef __CPU_HAL_H  ----- */
```

24.519 bsp_devboard.h File Reference

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

`#include <pins.h>`

24.519.1 Detailed Description

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

24.520 MOD5441X/include/bsp_devboard.h

Go to the documentation of this file.

```
00001 #ifndef __NB_BSP_H
00002 #define __NB_BSP_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00012 #include <pins.h>
00013
00014 #define PinGpioOutputFn PIN_GPIO
00015 #define PinGpioInputFn PIN_GPIO
00016
00017 #define LED_COUNT 8
00018
00019 #define PIN_LED1 15
00020 #define PIN_LED2 16
00021 #define PIN_LED3 31
00022 #define PIN_LED4 23
00023 #define PIN_LED5 37
00024 #define PIN_LED6 19
00025 #define PIN_LED7 20
00026 #define PIN_LED8 24
00027
00028 #define LED1 J2[PIN_LED1]
00029 #define LED2 J2[PIN_LED2]
00030 #define LED3 J2[PIN_LED3]
00031 #define LED4 J2[PIN_LED4]
00032 #define LED5 J2[PIN_LED5]
00033 #define LED6 J2[PIN_LED6]
00034 #define LED7 J2[PIN_LED7]
00035 #define LED8 J2[PIN_LED8]
00036
00037 class LEDArray
00038 {
00039    public:
00040     PinIO operator[](int n)
00041     {
00042         switch (n)
00043         {
00044             case 1: return J2[PIN_LED1];
00045             case 2: return J2[PIN_LED2];
00046             case 3: return J2[PIN_LED3];
00047             case 4: return J2[PIN_LED4];
00048             case 5: return J2[PIN_LED5];
00049             case 6: return J2[PIN_LED6];
00050             case 7: return J2[PIN_LED7];
00051             case 8: return J2[PIN_LED8];
00052             default: return J2[PIN_LED1];
00053         }
00054     }
00055 };
00056
00057 static LEDArray LEDs;
00058
00059 #endif
```

24.521 bsp_devboard.h File Reference

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.
`#include <pins.h>`

24.521.1 Detailed Description

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

24.522 MODM7AE70/include/bsp_devboard.h

Go to the documentation of this file.

```
00001 #ifndef __NB_BSP_H
00002 #define __NB_BSP_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00012 #include <pins.h>
00013
00014 #define PinGpioOutputFn PinIO::PIN_FN_OUT
00015 #define PinGpioInputFn PinIO::PIN_FN_OUT
00016
00017 #define LED_COUNT 8
00018
00019 #define PIN_LED1 15
00020 #define PIN_LED2 16
00021 #define PIN_LED3 18
00022 #define PIN_LED4 23
00023 #define PIN_LED5 17
00024 #define PIN_LED6 19
00025 #define PIN_LED7 20
00026 #define PIN_LED8 24
00027
00028 #define LED1 P2[PIN_LED1]
00029 #define LED2 P2[PIN_LED2]
00030 #define LED3 P2[PIN_LED3]
00031 #define LED4 P2[PIN_LED4]
00032 #define LED5 P2[PIN_LED5]
00033 #define LED6 P2[PIN_LED6]
00034 #define LED7 P2[PIN_LED7]
00035 #define LED8 P2[PIN_LED8]
00036
00037 class LEDArray
00038 {
00039    public:
00040     PinIO operator[](int n)
00041     {
00042         switch (n)
00043         {
00044             case 1: return P2[PIN_LED1];
00045             case 2: return P2[PIN_LED2];
00046             case 3: return P2[PIN_LED3];
00047             case 4: return P2[PIN_LED4];
00048             case 5: return P2[PIN_LED5];
00049             case 6: return P2[PIN_LED6];
00050             case 7: return P2[PIN_LED7];
00051             case 8: return P2[PIN_LED8];
00052             default: return P2[PIN_LED1];
00053         }
00054     }
00055 };
00056
00057 static LEDArray LEDs;
00058
00059 #endif
```

24.523 bsp_devboard.h File Reference

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.
`#include <pins.h>`

24.523.1 Detailed Description

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

24.524 NANO54415/include/bsp_devboard.h

Go to the documentation of this file.

```
00001 #ifndef __NB_BSP_H
00002 #define __NB_BSP_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00012 #include <pins.h>
00013
00014 #define PinGpioOutputFn PIN_GPIO
00015 #define PinGpioInputFn PIN_GPIO
00016
00017 #define LED_COUNT 4
00018
00019 #define PIN_LED0 19
00020 #define PIN_LED1 21
00021 #define PIN_LED2 23
00022 #define PIN_LED3 25
00023
00024 #define LED0 Pins[PIN_LED0]
00025 #define LED1 Pins[PIN_LED1]
00026 #define LED2 Pins[PIN_LED2]
00027 #define LED3 Pins[PIN_LED3]
00028
00029 class LEDArray
00030 {
00031    public:
00032     PinIO operator[](int n)
00033     {
00034         switch (n)
00035         {
00036             case 0: return Pins[PIN_LED0];
00037             case 1: return Pins[PIN_LED1];
00038             case 2: return Pins[PIN_LED2];
00039             case 3: return Pins[PIN_LED3];
00040             default: return Pins[PIN_LED0];
00041         }
00042     }
00043 };
00044
00045 static LEDArray LEDs;
00046
00047 #endif
```

24.525 bsp_devboard.h File Reference

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

`#include <pins.h>`

24.525.1 Detailed Description

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

24.526 SB800EX/include/bsp_devboard.h

Go to the documentation of this file.

```
00001 #ifndef __NB_BSP_H
00002 #define __NB_BSP_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00012 #include <pins.h>
00013
00014 #define PinGpioOutputFn PIN_GPIO
00015 #define PinGpioInputFn PIN_GPIO
00016
```

```
00017 // Note:
00018 //  The SB800EX has two bidirectional/dual color indicator LEDs.
00019 //  For the sake of usage simplicity, this header treats each direction of the
00020 //  LED as a separate LED.
00021 //  The mapping from Software to Hardware is:
00022 //  LED1 => LED1, Red
00023 //  LED2 => LED2, Red
00024 //  LED3 => LED1, Green
00025 //  LED4 => LED2, Green
00026 #define LED_COUNT 4
00027
00028 #define PIN_LED1 7
00029 #define PIN_LED2 5
00030 #define PIN_LED3 4
00031 #define PIN_LED4 0
00032
00033 #define LED1 BiDirLED(PortI[PIN_LED1], PortI[PIN_LED2])
00034 #define LED2 BiDirLED(PortI[PIN_LED3], PortH[PIN_LED4])
00035 #define LED3 BiDirLED(PortI[PIN_LED2], PortI[PIN_LED1])
00036 #define LED4 BiDirLED(PortH[PIN_LED4], PortI[PIN_LED3])
00037
00038 class BiDirLED
00039 {
00040     CPU_PINS::PinIO A;
00041     CPU_PINS::PinIO K;
00042
00043   public:
00044     BiDirLED(const CPU_PINS::PinIO &anode, const CPU_PINS::PinIO &cathode) : A(anode), K(cathode) {}
00045     BiDirLED() : A(), K() {}
00046
00047     void set(BOOL val = TRUE)
00048     {
00049         A.set(val);
00050         K.clr();
00051     }   // Set output high
00052     BOOL toggle()
00053     {
00054         K.clr();
00055         return A.toggle();
00056     }   // Toggle the pin state
00057     void clr()
00058     {
00059         A.clr();
00060         K.clr();
00061     }                                   // Set output low
00062     BOOL read() { return A.read(); }    // Read pin hi/low state
00063     void hiz() { read(); }              // Set output to tristate
00064     void drive() { A.drive(); }         // Turn output on (opposite of tristate)
00065
00066     void function(int ft)
00067     {
00068         A.function(ft);
00069         K.function(ft);
00070     }                                               // Set pin to special function
00071     int getFunction() { return A.getFunction(); }   // Get the special function the pin is set to
00072
00073     BiDirLED &operator=(BOOL b)
00074     {
00075         A.set(b);
00076         K.clr();
00077         return *this;
00078     }
00079     BiDirLED &operator=(int i)
00080     {
00081         if (i > 0)
00082         {
00083             A.set(1);
00084             K.set(0);
00085         }
00086         else if (i < 0)
00087         {
00088             A.set(0);
00089             K.set(1);
00090         }
00091         else
00092         {
00093             A.set(0);
00094             K.set(0);
00095         }
00096         return *this;
00097     };
00098
00099     operator int() { return read(); };          // Read and return int value
00100     operator BOOL() { return read(); };         // Read and return BOOL value
00101     operator bool() { return (read() != 0); };  // Read and return boolean value
00102 };
00103
```

```
00104 class LEDArray
00105 {
00106    public:
00107     BiDirLED operator[](int n)
00108     {
00109         switch (n)
00110         {
00111             case 1: return LED1;
00112             case 2: return LED2;
00113             case 3: return LED3;
00114             case 4: return LED4;
00115             default: return LED1;
00116         }
00117     }
00118 };
00119
00120 static LEDArray LEDs;
00121
00122 #endif
```

24.527 bsp_devboard.h File Reference

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

`#include <pins.h>`

24.527.1 Detailed Description

Hardware definitions related to the standard NNDK carrier board that are unique to a specific platform.

24.528 SOMRT1061/include/bsp_devboard.h

Go to the documentation of this file.

```
00001 #ifndef __NB_BSP_H
00002 #define __NB_BSP_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007
00013 #include <pins.h>
00014
00015 #define PinGpioOutputFn PinIO::PIN_FN_OUT
00016 #define PinGpioInputFn PinIO::PIN_FN_OUT
00017
00018 #define LED_COUNT 4
00019
00020 #define PIN_LED0 86
00021 #define PIN_LED1 85
00022 #define PIN_LED2 75
00023 #define PIN_LED3 74
00024
00025 #define LED0 Pins[PIN_LED0]
00026 #define LED1 Pins[PIN_LED1]
00027 #define LED2 Pins[PIN_LED2]
00028 #define LED3 Pins[PIN_LED3]
00029
00030 class LEDArray {
00031     public:
00032     PinIO operator[] (int n){
00033         switch (n) {
00034             case 0: return Pins[PIN_LED0];
00035             case 1: return Pins[PIN_LED1];
00036             case 2: return Pins[PIN_LED2];
00037             case 3: return Pins[PIN_LED3];
00038             default:return Pins[PIN_LED0];
00039         }
00040     }
00041 };
00042
00043 static LEDArray LEDs;
00044
00045 #endif
```

24.529 MOD5441X/include/pinconstant.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef MOD54415_PIN_CONSTANT
00006 #define MOD54415_PIN_CONSTANT
00007
00008 //#define ENABLE_BE2_BE3      ( 1 )   // Uncomment only for MOD54415 H/W Revision 1.7 and earlier
00009
00010 #define PIN_GPIO (0)   // Universal GPIO Definition
00011
00012 // Connector: J1 / Pin:  5 / CPU Pin: E1
00013 #define PINJ1_5_FB_CS1 (3)   // Primary Function: General Purpose Chip Select 1
00014 #define PINJ1_5_NFC_CE (1)   // Alternate Function: NAND Flash Controller - Chip Enable
00015 #define PINJ1_5_GPIO (0)     // Port B - 4
00016
00017 // Connector: J1 / Pin:  6 / CPU Pin: B1
00018 #define PINJ1_6_FB_CS4 (3)   // Primary Function: General Purpose Chip Select 4
00019 #define PINJ1_6_DREQ1 (2)    // Alternate Function: External DMA Request 1
00020 #define PINJ1_6_GPIO (0)     // Port B - 5
00021
00022 // Connector: J1 / Pin:  7 / CPU Pin: F2
00023 #define PINJ1_7_FB_CS5 (3)   // Primary Function: General Purpose Chip Select 5
00024 #define PINJ1_7_DACK1 (2)    // Alternate Function: External DMA Acknowledge 1
00025 #define PINJ1_7_GPIO (0)     // Port B - 6
00026
00027 #ifdef ENABLE_BE2_BE3
00028
00029 // Connector: J1 / Pin:  9 / CPU Pin: E2
00030 #define PINJ1_9_FB_BE2 (3)   // Primary Function: Byte Enable 2
00031 #define PINJ1_9_FB_CS2 (2)   // Alternate Function 1: General Purpose Chip Select 2
00032 #define PINJ1_9_FB_A0 (1)    // Alternate Function 2: Address 0 / NAND Flash Controller - Command
      Latch Enable
00033 #define PINJ1_9_GPIO (0)     // Port A - 2
00034
00035 // Connector: J1 / Pin: 10 / CPU Pin: C1
00036 #define PINJ1_10_FB_BE3 (3)   // Primary Function: Byte Enable 3
00037 #define PINJ1_10_FB_CS3 (2)   // Alternate Function 1: General Purpose Chip Select 3
00038 #define PINJ1_10_FB_A1 (1)    // Alternate Function 2: Address 1 / NAND Flash Controller - Address
      Latch Enable
00039 #define PINJ1_10_GPIO (0)     // Port A - 3
00040
00041 #else
00042
00043 // Connector: J1 / Pin: 9 / CPU Pin: D1
00044 #define PINJ1_9_FB_BE1 (3)      // Primary Function: Byte Enable 1
00045 #define PINJ1_9_FB_TSIZ1 (2)    // Alternate Function: Transfer Size 1
00046 #define PINJ1_9_GPIO (0)        // Port A - 1
00047
00048 // Connector: J1 / Pin: 10 / CPU Pin: F4
00049 #define PINJ1_10_FB_BE0 (3)     // Primary Function: Byte Enable 0
00050 #define PINJ1_10_FB_TSIZ0 (2)   // Alternate Function: Transfer Size 0
00051 #define PINJ1_10_GPIO (0)       // Port A - 0
00052
00053 #endif   // ENABLE_BE2_BE3
00054
00055 // Connector: J1 / Pin: 13 / CPU Pin: H3
00056 #define PINJ1_13_FB_TA (3)    // Primary Function: Transfer Acknowledge
00057 #define PINJ1_13_NFC_RB (1)   // Alternate Function: NAND Flash Controller - Flash Ready/Busy
00058 #define PINJ1_13_GPIO (0)     // Port A - 4
00059
00060 // Connector: J1 / Pin: 31 / CPU Pin: G1
00061 #define PINJ1_31_FB_CLK (1)   // Primary Function: Internal Bus Clock
00062 #define PINJ1_31_GPIO (0)     // Port B - 7
00063
00064 // Connector: J2 / Pin:  1 / CPU Pin: NA  / Reserved For: GND
00065
00066 // Connector: J2 / Pin:  2 / CPU Pin: NA  / Reserved For: VCC3V
00067
00068 // Connector: J2 / Pin:  3 / CPU Pin: B10
00069 #define PINJ2_3_UART0_RXD (3)   // Primary Function: UART 0 - Receive
00070 #define PINJ2_3_I2C4_SDA (2)    // Alternate Function 1: I2C 4 - Serial Data
00071 #define PINJ2_3_DSPI2_SIN (1)   // Alternate Function 2: DSPI 2 - Serial Data In
00072 #define PINJ2_3_GPIO (0)        // Port F - 4
00073
00074 // Connector: J2 / Pin:  4 / CPU Pin: D11
00075 #define PINJ2_4_UART0_TXD (3)    // Primary Function: UART 0 - Transmit
00076 #define PINJ2_4_I2C4_SCL (2)     // Alternate Function 1: I2C 4 - Serial Clock
00077 #define PINJ2_4_DSPI2_SOUT (1)   // Alternate Function 2: DSPI 2 - Serial Data Out
00078 #define PINJ2_4_GPIO (0)         // Port F - 3
00079
00080 // Connector: J2 / Pin:  5 / CPU Pin: J4  / Reserved For: VDDA_DAC_ADC
00081
00082 // Connector: J2 / Pin:  6 / CPU Pin: H1  / Reserved For: ADC_IN0
00083
```

```
00084 // Connector: J2 / Pin:  7 / CPU Pin: J1  / Reserved For: ADC_IN1
00085
00086 // Connector: J2 / Pin:  8 / CPU Pin: J2  / Reserved For: ADC_IN2
00087
00088 // Connector: J2 / Pin:  9 / CPU Pin: K4  / Reserved For: ADC_IN3
00089
00090 // Connector: J2 / Pin: 10 / CPU Pin: G4  / Reserved For: ADC_IN4
00091
00092 // Connector: J2 / Pin: 11 / CPU Pin: J3  / Reserved For: ADC_IN5
00093
00094 // Connector: J2 / Pin: 12 / CPU Pin: H2  / Reserved For: ADC_IN6
00095
00096 // Connector: J2 / Pin: 13 / CPU Pin: K3  / Reserved For: ADC_IN7
00097
00098 // Connector: J2 / Pin: 14 / CPU Pin: NA  / Reserved For: AGND
00099
00100 // Connector: J2 / Pin: 15 / CPU Pin: A12
00101 #define PINJ2_15_SSI0_MCLK (3)   // Primary Function: SSI 0 - Serial Master Clock
00102 #define PINJ2_15_SSI_CLKIN (2)   // Alternate Function 1: SSI Clock Input
00103 #define PINJ2_15_SIM1_CLK (1)    // Alternate Function 2: SIM 1 - Clock
00104 #define PINJ2_15_GPIO (0)        // Port H - 4
00105
00106 // Connector: J2 / Pin: 16 / CPU Pin: A13
00107 #define PINJ2_16_SSI0_BCLK (3)   // Primary Function: SSI 0 - Serial Bit Clock
00108 #define PINJ2_16_UART7_RXD (2)   // Alternate Function 1: UART 7 - Receive
00109 #define PINJ2_16_SIM1_PD (1)     // Alternate Function 2: SIM 1 - Card Insertion Detect Signal
00110 #define PINJ2_16_GPIO (0)        // Port H - 3
00111
00112 // Connector: J2 / Pin: 17 / CPU Pin: A14 / Available On: USB- On-the-Go [v1.7 and later (H/W
      default)]
00113 // Connector: J2 / Pin: 17 / CPU Pin: A15 / Available On: USB- Host (v1.7 and later)
00114
00115 // Connector: J2 / Pin: 17 / CPU Pin: C12 / Available On: v1.6 and earlier (H/W default), v1.9 and
      later
00116 #define PINJ2_17_SSI0_RXD (3)   // Primary Function: SSI 0 - Serial Receive Data
00117 #define PINJ2_17_I2C2_SDA (2)   // Alternate Function 1: I2C 2 - Serial Data
00118 #define PINJ2_17_SIM1_VEN (1)   // Alternate Function 2: SIM 1 - Power Supply Enable Signal
00119 #define PINJ2_17_GPIO (0)       // Port H - 7
00120
00121 // Connector: J2 / Pin: 18 / CPU Pin: B14 / Available On: USB+ On-the-Go [v1.7 and later (H/W
      default)]
00122 // Connector: J2 / Pin: 18 / CPU Pin: B15 / Available On: USB+ Host (v1.7 and later)
00123
00124 // Connector: J2 / Pin: 18 / CPU Pin: C13 / Available On: v1.6 and earlier (H/W default), v1.9 and
      later
00125 #define PINJ2_18_SSI0_TXD (3)    // Primary Function: SSI 0 - Serial Transmit Data
00126 #define PINJ2_18_I2C2_SCL (2)    // Alternate Function 1: I2C 2 - Serial Clock
00127 #define PINJ2_18_SIM1_DATA (1)   // Alternate Function 2: SIM 1 - Bidirectional Transmit/Receive Data
      Signal
00128 #define PINJ2_18_GPIO (0)        // Port H - 6
00129
00130 // Connector: J2 / Pin: 19 / CPU Pin: N2
00131 #define PINJ2_19_UART2_TXD (3)   // Primary Function: UART 2 - Transmit
00132 #define PINJ2_19_PWM_B3 (2)      // Alternate Function 1: PWM B3 - Output Signal/Input Capture
00133 #define PINJ2_19_SSI1_TXD (1)    // Alternate Function 2: SSI 1 - Serial Transmit Data
00134 #define PINJ2_19_GPIO (0)        // Port E - 3
00135
00136 // Connector: J2 / Pin: 20 / CPU Pin: E15
00137 #define PINJ2_20_SSI0_FS (3)     // Primary Function: SSI 0 - Serial Frame Sync
00138 #define PINJ2_20_UART7_TXD (2)   // Alternate Function 1: UART 7 - Transmit
00139 #define PINJ2_20_SIM1_RST (1)    // Alternate Function 2: SIM 1 - Reset Signal
00140 #define PINJ2_20_GPIO (0)        // Port H - 5
00141
00142 // Connector: J2 / Pin: 21 / CPU Pin: C9
00143 #define PINJ2_21_UART1_RXD (3)   // Primary Function: UART 1 - Receive
00144 #define PINJ2_21_I2C5_SDA (2)    // Alternate Function 1: I2C 5 - Serial Data
00145 #define PINJ2_21_DSPI3_SIN (1)   // Alternate Function 2: DSPI 3 - Serial Data In
00146 #define PINJ2_21_GPIO (0)        // Port E - 0
00147
00148 // Connector: J2 / Pin: 22 / CPU Pin: D9
00149 #define PINJ2_22_UART1_TXD (3)    // Primary Function: UART 1 - Transmit
00150 #define PINJ2_22_I2C5_SCL (2)     // Alternate Function 1: I2C 5 - Serial Clock
00151 #define PINJ2_22_DSPI3_SOUT (1)   // Alternate Function 2: DSPI 3 - Serial Data Out
00152 #define PINJ2_22_GPIO (0)         // Port F - 7
00153
00154 // Connector: J2 / Pin: 23 / CPU Pin: D10
00155 #define PINJ2_23_UART1_RTS (3)    // Primary Function: UART 1 - Request to Send
00156 #define PINJ2_23_UART5_RXD (2)    // Alternate Function 1: UART 5 - Receive
00157 #define PINJ2_23_DSPI3_PCS0 (1)   // Alternate Function 2: DSPI 3 - Peripheral Chip Select 0
00158 #define PINJ2_23_GPIO (0)         // Port E - 1 / Rapid GPIO 8
00159
00160 // Connector: J2 / Pin: 24 / CPU Pin: C10
00161 #define PINJ2_24_UART1_CTS (3)   // Primary Function: UART 1 - Clear to Send
00162 #define PINJ2_24_UART5_TXD (2)   // Alternate Function 1: UART 5 - Transmit
00163 #define PINJ2_24_DSPI3_SCK (1)   // Alternate Function 2: DSPI 3 - Serial Clock
00164 #define PINJ2_24_GPIO (0)        // Port E - 2 / Rapid GPIO 7
00165
```

```
00166 // Connector: J2 / Pin: 25 / CPU Pin: A10
00167 #define PINJ2_25_SDHC_CLK (3)     // Primary Function: SDHC Clock
00168 #define PINJ2_25_PWM_A0 (2)       // Alternate Function 1: PWM A0 - Output Signal/Input Capture
00169 #define PINJ2_25_DSPI1_SCK (1)    // Alternate Function 2: DSPI 1 - Serial Clock
00170 #define PINJ2_25_GPIO (0)         // Port G - 5
00171
00172 // Connector: J2 / Pin: 26 / CPU Pin: M1
00173 #define PINJ2_26_IRQ3 (3)           // Primary Function: External Interrupt 3
00174 #define PINJ2_26_DSPI0_PCS3 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 3
00175 #define PINJ2_26_USBH_VBUS_EN (1)   // Alternate Function 2: USB Host VBUS Enable
00176 #define PINJ2_26_GPIO (0)           // Port C - 3
00177
00178 // Connector: J2 / Pin: 27 / CPU Pin: C11
00179 #define PINJ2_27_SDHC_CMD (3)     // Primary Function: SDHC Command Line
00180 #define PINJ2_27_PWM_B0 (2)       // Alternate Function 1: PWM B0 - Output Signal/Input Capture
00181 #define PINJ2_27_DSPI1_SIN (1)    // Alternate Function 2: DSPI 1 - Serial Data In
00182 #define PINJ2_27_GPIO (0)         // Port G - 6
00183
00184 // Connector: J2 / Pin: 28 / CPU Pin: B12
00185 #define PINJ2_28_SDHC_DAT0 (3)     // Primary Function: SDHC DAT0 Line / Busy-State Detect
00186 #define PINJ2_28_PWM_B2 (2)        // Alternate Function 1: PWM B2 - Output Signal/Input Capture
00187 #define PINJ2_28_DSPI1_SOUT (1)    // Alternate Function 2: DSPI 1 - Serial Data Out
00188 #define PINJ2_28_GPIO (0)          // Port G - 7
00189
00190 // Connector: J2 / Pin: 29 / CPU Pin: E13
00191 #define PINJ2_29_UART0_CTS (3)    // Primary Function: UART 0 - Clear to Send
00192 #define PINJ2_29_UART4_TXD (2)    // Alternate Function 1: UART 4 - Transmit
00193 #define PINJ2_29_DSPI2_SCK (1)    // Alternate Function 2: DSPI 2 - Serial Clock
00194 #define PINJ2_29_GPIO (0)         // Port F - 6 / Rapid GPIO 5
00195
00196 // Connector: J2 / Pin: 30 / CPU Pin: B13
00197 #define PINJ2_30_SDHC_DAT3 (3)     // Primary Function: SDHC DAT3 Line / Card Detection
00198 #define PINJ2_30_PWM_A1 (2)        // Alternate Function 1: PWM A1 - Output Signal/Input Capture
00199 #define PINJ2_30_DSPI1_PCS0 (1)    // Alternate Function 2: DSPI 1 - Peripheral Chip Select 0
00200 #define PINJ2_30_GPIO (0)          // Port F - 2
00201
00202 // Connector: J2 / Pin: 31 / CPU Pin: P1
00203 #define PINJ2_31_UART2_RXD (3)    // Primary Function: UART 2 - Receive
00204 #define PINJ2_31_PWM_A3 (2)       // Alternate Function 1: PWM A3 - Output Signal/Input Capture
00205 #define PINJ2_31_SSI1_RXD (1)     // Alternate Function 2: SSI 1 - Serial Receive Data
00206 #define PINJ2_31_GPIO (0)         // Port E - 4
00207
00208 // Connector: J2 / Pin: 32 / CPU Pin: G13
00209 #define PINJ2_32_T3IN (3)           // Primary Function: Timer Input 3
00210 #define PINJ2_32_T3OUT (2)          // Alternate Function 1: Timer Output 3
00211 #define PINJ2_32_USBO_VBUS_EN (1)   // Alternate Function 2: USB On-the-Go VBUS Enable
00212 #define PINJ2_32_GPIO (0)           // Port D - 2 / Rapid GPIO 1
00213
00214 // Connector: J2 / Pin: 33 / CPU Pin: H14
00215 #define PINJ2_33_T2IN (3)         // Primary Function: Timer Input 2
00216 #define PINJ2_33_T2OUT (2)        // Alternate Function 1: Timer Output 2
00217 #define PINJ2_33_SDHC_DAT2 (1)    // Alternate Function 2: SDHC DAT2 Line / Read Wait
00218 #define PINJ2_33_GPIO (0)         // Port D - 1 / Rapid GPIO 2
00219
00220 // Connector: J2 / Pin: 34 / CPU Pin: H13
00221 #define PINJ2_34_T1IN (3)         // Primary Function: Timer Input 1
00222 #define PINJ2_34_T1OUT (2)        // Alternate Function 1: Timer Output 1
00223 #define PINJ2_34_SDHC_DAT1 (1)    // Alternate Function 2: SDHC DAT1 Line / Interrupt Detect
00224 #define PINJ2_34_GPIO (0)         // Port D - 0 / Rapid GPIO 3
00225
00226 // Connector: J2 / Pin: 35 / CPU Pin: D12
00227 #define PINJ2_35_SDHC_DAT1 (3)     // Primary Function: SDHC DAT1 Line / Interrupt Detect
00228 #define PINJ2_35_PWM_A2 (2)        // Alternate Function 1: PWM A2 - Output Signal/Input Capture
00229 #define PINJ2_35_DSPI1_PCS1 (1)    // Alternate Function 2: DSPI 1 - Peripheral Chip Select 1
00230 #define PINJ2_35_GPIO (0)          // Port F - 0
00231
00232 // Connector: J2 / Pin: 36 / CPU Pin: H15
00233 #define PINJ2_36_T0IN (3)           // Primary Function: Timer Input 0
00234 #define PINJ2_36_T0OUT (2)          // Alternate Function 1: Timer Output 0
00235 #define PINJ2_36_USBO_VBUS_OC (1)   // Alternate Function 2: USB On-the-Go VBUS Over-Current
00236 #define PINJ2_36_GPIO (0)           // Port E - 7 / Rapid GPIO 4
00237
00238 // Connector: J2 / Pin: 37 / CPU Pin: N11
00239 #define PINJ2_37_OW_DAT (3)    // Primary Function: 1-Wire Data Signal
00240 #define PINJ2_37_DACK0 (2)     // Alternate Function: DMA Acknowledge 0
00241 #define PINJ2_37_GPIO (0)      // Port D - 3 / Rapid GPIO 0
00242
00243 // Connector: J2 / Pin: 38 / CPU Pin: B11
00244 #define PINJ2_38_UART0_RTS (3)     // Primary Function: UART 0 - Request to Send
00245 #define PINJ2_38_UART4_RXD (2)     // Alternate Function 1: UART 4 - Receive
00246 #define PINJ2_38_DSPI2_PCS0 (1)    // Alternate Function 2: DSPI 2 - Peripheral Chip Select 0
00247 #define PINJ2_38_GPIO (0)          // Port F - 5 / Rapid GPIO 6
00248
00249 // Connector: J2 / Pin: 39 / CPU Pin: G14
00250 #define PINJ2_39_I2C0_SDA (3)     // Primary Function: I2C 0 - Serial Data
00251 #define PINJ2_39_UART8_RXD (2)    // Alternate Function 1: UART 8 - Receive
00252 #define PINJ2_39_CAN0_RX (1)      // Alternate Function 2: CAN 0 - Receive
```

```
00253 #define PINJ2_39_GPIO (0)        // Port B - 1
00254
00255 // Connector: J2 / Pin: 40 / CPU Pin: E14
00256 #define PINJ2_40_SDHC_DAT2 (3)    // Primary Function: SDHC DAT2 Line / Read Wait
00257 #define PINJ2_40_PWM_B1 (2)       // Alternate Function 1: PWM B1 - Output Signal/Input Capture
00258 #define PINJ2_40_DSPI1_PCS2 (1)   // Alternate Function 2: DSPI 1 - Peripheral Chip Select 2
00259 #define PINJ2_40_GPIO (0)         // Port F - 1
00260
00261 // Connector: J2 / Pin: 41 / CPU Pin: D15
00262 #define PINJ2_41_CAN1_RX (3)      // Primary Function: CAN 1 - Receive
00263 #define PINJ2_41_UART9_RXD (2)    // Alternate Function 1: UART 9 - Receive
00264 #define PINJ2_41_I2C1_SDA (1)     // Alternate Function 2: I2C 1 - Serial Data
00265 #define PINJ2_41_GPIO (0)         // Port C - 7
00266
00267 // Connector: J2 / Pin: 42 / CPU Pin: G15
00268 #define PINJ2_42_I2C0_SCL (3)     // Primary Function: I2C 0 - Serial Clock
00269 #define PINJ2_42_UART8_TXD (2)    // Alternate Function 1: UART 8 - Transmit
00270 #define PINJ2_42_CAN0_TX (1)      // Alternate Function 2: CAN 0 - Transmit
00271 #define PINJ2_42_GPIO (0)         // Port B - 2
00272
00273 // Connector: J2 / Pin: 43 / CPU Pin: M2
00274 #define PINJ2_43_IRQ2 (3)           // Primary Function: External Interrupt 2
00275 #define PINJ2_43_DSPI0_PCS2 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 2
00276 #define PINJ2_43_USBH_VBUS_OC (1)   // Alternate Function 2: USB Host VBUS Over-Current
00277 #define PINJ2_43_GPIO (0)           // Port C - 2
00278
00279 // Connector: J2 / Pin: 44 / CPU Pin: D14
00280 #define PINJ2_44_CAN1_TX (3)      // Primary Function: CAN 1 - Transmit
00281 #define PINJ2_44_UART9_TXD (2)    // Alternate Function 1: UART 9 - Transmit
00282 #define PINJ2_44_I2C1_SCL (1)     // Alternate Function 2: I2C 1 - Serial Clock
00283 #define PINJ2_44_GPIO (0)         // Port B - 0
00284
00285 // Connector: J2 / Pin: 45 / CPU Pin: F13
00286 #define PINJ2_45_IRQ1 (3)   // Primary Function: External Interrupt 1
00287 #define PINJ2_45_GPIO (0)   // Port C - 1
00288
00289 // Connector: J2 / Pin: 46 / CPU Pin: NA  / Reserved For: GND
00290
00291 // Connector: J2 / Pin: 47 / CPU Pin: N1
00292 #define PINJ2_47_IRQ6 (3)         // Primary Function: External Interrupt 6
00293 #define PINJ2_47_USB_CLKIN (1)    // Alternate Function: USB Clock In
00294 #define PINJ2_47_GPIO (0)         // Port C - 5
00295
00296 // Connector: J2 / Pin: 48 / CPU Pin: F12
00297 #define PINJ2_48_IRQ7 (1)   // Primary Function: External Interrupt 7
00298 #define PINJ2_48_GPIO (0)   // Port C - 6
00299
00300 // Connector: J2 / Pin: 49 / CPU Pin: NA  / Reserved For: GND
00301
00302 // Connector: J2 / Pin: 50 / CPU Pin: NA  / Reserved For: VCC3V
00303
00304 #endif   // MOD54415_PIN_CONSTANT
```

24.530 MODM7AE70/include/pinconstant.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef MODM7AE70_PIN_CONSTANT
00006 #define MODM7AE70_PIN_CONSTANT
00007
00008 #define PIN_GPIO (PinIO::PIN_FN_IN)
00009 #define PIN_GPIO_OUT (PinIO::PIN_FN_OUT)
00010
00011 // Pin P2_12 is multiplexed between pins A5 and B5
00012 // Select only one signal to be controlled for P2_12
00013 #define P2_12_USE_A5
00014 //#define P2_12_USE_B5
00015
00016 // Connector: P1 / Pin: 4 / CPU Port: PC8
00017 #define PINP1_4_IN (PinIO::PIN_FN_IN)     // GPIO Input
00018 #define PINP1_4_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00019 #define PINP1_4_NWE (PinIO::PIN_FN_A)     // Peripheral A: Write Enable
00020 #define PINP1_4_TIOA7 (PinIO::PIN_FN_B)   // Peripheral B: Timer 7 Line A
00021
00022 // Connector: P1 / Pin: 5 / CPU Port: PA22
00023 // Warning: If external bus buffer is enabled, this pin must be configured for PINP1_5_NCS2
00024 // or left in its default pull-up state.
00025 #define PINP1_5_IN (PinIO::PIN_FN_IN)               // GPIO Input
00026 #define PINP1_5_OUT (PinIO::PIN_FN_OUT)             // GPIO Output
00027 #define PINP1_5_RK (PinIO::PIN_FN_A)                // Peripheral A: USART 1 SSC Receive Clock
00028 #define PINP1_5_PWMC0_PWMEXTRG1 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 External Trigger 1
00029 #define PINP1_5_NCS2 (PinIO::PIN_FN_C)              // Peripheral C: Bus Chip Select 2
00030
```

```
00031 // Connector: P1 / Pin: 6 / CPU Port: PC14
00032 // Warning: If external bus buffer is enabled, this pin must be configured for PINP1_6_NCS0
00033 // or left in its default pull-up state.
00034 #define PINP1_6_IN (PinIO::PIN_FN_IN)      // GPIO Input
00035 #define PINP1_6_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00036 #define PINP1_6_NCS0 (PinIO::PIN_FN_A)     // Peripheral A: Bus Chip Select 0
00037 #define PINP1_6_TCLK8 (PinIO::PIN_FN_B)    // Peripheral B: Timer 8 Clock
00038 #define PINP1_6_CANTX1 (PinIO::PIN_FN_C)   // Peripheral C: CAN 1 Transmit
00039
00040 // Connector: P1 / Pin: 7 / CPU Port: PD19
00041 // Warning: If external bus buffer is enabled, this pin must be configured for PINP1_7_NCS3
00042 // or left in its default pull-up state.
00043 #define PINP1_7_IN (PinIO::PIN_FN_IN)     // GPIO Input
00044 #define PINP1_7_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00045 #define PINP1_7_NCS3 (PinIO::PIN_FN_A)    // Peripheral A: Bus Chip Select 3
00046 #define PINP1_7_CTS2 (PinIO::PIN_FN_B)    // Peripheral B: USART 2 CTS
00047 #define PINP1_7_UTXD4 (PinIO::PIN_FN_C)   // Peripheral C: UART 4 Transmit
00048 #define PINP1_7_SER_TX6 (PinIO::PIN_FN_C) // Peripheral C: Serial Port 6 Transmit
00049
00050 // Connector: P1 / Pin: 8 / CPU Port: PC11
00051 // Warning: If external bus buffer is enabled, this pin must be configured for PINP1_8_NRD
00052 // or left in its default pull-up state.
00053 #define PINP1_8_IN (PinIO::PIN_FN_IN)     // GPIO Input
00054 #define PINP1_8_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00055 #define PINP1_8_NRD (PinIO::PIN_FN_A)     // Peripheral A: Read Signal
00056 #define PINP1_8_TIOA8 (PinIO::PIN_FN_B)   // Peripheral B: Timer 8 Line A
00057
00058 // Connector: P1 / Pin: 13 / CPU Port: PC13
00059 // Pin is tied to SDRAM and must only be used for EBI functionality
00060 // External bus buffer enable from bsp.h should be used to drive this signal
00061 #define PINP1_13_IN (PinIO::PIN_FN_IN)           // GPIO Input
00062 #define PINP1_13_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00063 #define PINP1_13_NWAIT (PinIO::PIN_FN_A)         // Peripheral A: External Wait Signal
00064 #define PINP1_13_PWMC0_PWMH3 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 CHannel 3 Output High
00065
00066 // Connector: P1 / Pin: 31 / CPU Port: PA6
00067 #define PINP1_31_IN (PinIO::PIN_FN_IN)     // GPIO Input
00068 #define PINP1_31_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00069 #define PINP1_31_PCK0 (PinIO::PIN_FN_B)    // Peripheral B: Programmable Clock 0 Output
00070 #define PINP1_31_UTXD1 (PinIO::PIN_FN_C)   // Peripheral C: UART 1 Transmit
00071 #define PINP1_31_SER_TX3 (PinIO::PIN_FN_C) // Peripheral C: Serial Port 3 Transmit
00072
00073 // Connector: P1 / Pin: 33 / CPU Port: PC19
00074 #define PINP1_33_IN (PinIO::PIN_FN_IN)           // GPIO Input
00075 #define PINP1_33_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00076 #define PINP1_33_A1 (PinIO::PIN_FN_A)            // Peripheral A: External Bus Interface A1
00077 #define PINP1_33_PWMC0_PWMH2 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 Channel 2 Output High
00078
00079 // Connector: P1 / Pin: 44 / CPU Port: PC30
00080 #define PINP1_44_IN (PinIO::PIN_FN_IN)     // GPIO Input
00081 #define PINP1_44_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00082 #define PINP1_44_A12 (PinIO::PIN_FN_A)     // Peripheral A: External Bus Interface A12
00083 #define PINP1_44_TIOB5 (PinIO::PIN_FN_B)   // Peripheral B: Timer 5 Line B
00084
00085 // Connector: P1 / Pin: 47 / CPU Port: PA19
00086 #define PINP1_47_IN (PinIO::PIN_FN_IN)           // GPIO Input
00087 #define PINP1_47_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00088 #define PINP1_47_PWMC0_PWML0 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 Channel 0 Output Low
00089 #define PINP1_47_A15 (PinIO::PIN_FN_C)           // Peripheral C: External Bus Interface A15
00090 #define PINP1_47_I2SC1_MCK (PinIO::PIN_FN_D)     // Peripheral D: Sound Controller 1 Master Clock
00091
00092 // Connector: P2 / Pin: 3 / CPU Port: PB0
00093 #define PINP2_3_IN (PinIO::PIN_FN_IN)           // GPIO Input
00094 #define PINP2_3_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00095 #define PINP2_3_PWMC0_PWMH0 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 0 Output High
00096 #define PINP2_3_RXD0 (PinIO::PIN_FN_C)          // Peripheral C: USART 0 Receive
00097 #define PINP2_3_SER_RX0 (PinIO::PIN_FN_C)       // Peripheral C: Serial Port 0 Receive
00098 #define PINP2_3_TF (PinIO::PIN_FN_D)            // Peripheral D: SSC Transmit Frame Sync
00099
00100 // Connector: P2 / Pin: 4 / CPU Port: PB1
00101 #define PINP2_4_IN (PinIO::PIN_FN_IN)           // GPIO Input
00102 #define PINP2_4_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00103 #define PINP2_4_PWMC0_PWMH1 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 1 Output High
00104 #define PINP2_4_GTSUCOMP (PinIO::PIN_FN_B)      // Peripheral B: TSU Timer Comparison Valid 1588
00105 #define PINP2_4_TXD0 (PinIO::PIN_FN_C)          // Peripheral C: USART 0 Transmit
00106 #define PINP2_4_SER_TX0 (PinIO::PIN_FN_C)       // Peripheral C: Serial Port 0 Transmit
00107 #define PINP2_4_TK (PinIO::PIN_FN_D)            // Peripheral D: SSC Transmit Clock
00108
00109 // Connector: P2 / Pin: 6 / CPU Port: PC12
00110 #define PINP2_6_IN (PinIO::PIN_FN_IN)      // GPIO Input
00111 #define PINP2_6_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00112 #define PINP2_6_TIOB8 (PinIO::PIN_FN_B)    // Peripheral B: Timer 8 Line B
00113 #define PINP2_6_CANRX1 (PinIO::PIN_FN_C)   // Peripheral C: CAN 1 Receive
00114
00115 // Connector: P2 / Pin: 7 / CPU Port: PD30
00116 #define PINP2_7_IN (PinIO::PIN_FN_IN)       // GPIO Input
00117 #define PINP2_7_OUT (PinIO::PIN_FN_OUT)     // GPIO Output
```

```
00118 #define PINP2_7_UTXD3 (PinIO::PIN_FN_A)     // Peripheral A: UART 3 Transmit
00119 #define PINP2_7_SER_TX5 (PinIO::PIN_FN_A)   // Peripheral A: Serial Port 5 Transmit
00120 #define PINP2_7_ISI_D10 (PinIO::PIN_FN_D)   // Peripheral D: Image Sensor Data Input 10
00121
00122 // Connector: P2 / Pin: 8 / CPU Port: PA17
00123 #define PINP2_8_IN (PinIO::PIN_FN_IN)           // GPIO Input
00124 #define PINP2_8_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00125 #define PINP2_8_QIO2 (PinIO::PIN_FN_A)          // Peripheral A: QSPI Data 2 Quad Mode
00126 #define PINP2_8_PCK1 (PinIO::PIN_FN_B)          // Peripheral B: Programmable Clock Output 1
00127 #define PINP2_8_PWMC0_PWMH3 (PinIO::PIN_FN_C)   // Peripheral C: PWM clock 0 Channel 3 Output High
00128
00129 // Connector: P2 / Pin: 9 / CPU Port: PA2
00130 #define PINP2_9_IN (PinIO::PIN_FN_IN)           // GPIO Input
00131 #define PINP2_9_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00132 #define PINP2_9_PWMC0_PWMH1 (PinIO::PIN_FN_A)   // Peripheral A: PWM Clock 0 Channel 1 Output High
00133 #define PINP2_9_DATRG (PinIO::PIN_FN_C)         // Peripheral C: DAC Trigger Input
00134
00135 // Connector: P2 / Pin: 10 / CPU Port: PD18
00136 #define PINP2_10_IN (PinIO::PIN_FN_IN)     // GPIO Input
00137 #define PINP2_10_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00138 #define PINP2_10_URXD4 (PinIO::PIN_FN_C)   // Peripheral C: UART 4 Receive
00139 #define PINP2_10_SER_RX6 (PinIO::PIN_FN_C) // Peripheral C: Serial Port 6 Receive
00140
00141 // Connector: P2 / Pin: 11 / CPU Port: PB13
00142 #define PINP2_11_IN (PinIO::PIN_FN_IN)           // GPIO Input
00143 #define PINP2_11_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00144 #define PINP2_11_PWMC0_PWML2 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 2 Output Low
00145 #define PINP2_11_PCK0 (PinIO::PIN_FN_B)          // Peripheral B: Programmable Clock output 0
00146 #define PINP2_11_SCK0 (PinIO::PIN_FN_C)          // Peripheral C: USART 0 Serial Clock
00147
00148 // Pin P2_12 is multiplexed between to CPU pins (PA5 and PB5).
00149 // The multiplexed pin used in the pins class must be defined at the top of pinconstant.h
00150 #define PINP2_12_IN (PinIO::PIN_FN_IN)     // GPIO Input
00151 #define PINP2_12_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00152 #ifdef P2_12_USE_B5
00153 // Connector: P2 / Pin: 12 / CPU Port: PB5
00154 #define PINP2_12_TWCK1 (PinIO::PIN_FN_A)         // Peripheral A: Two-wire (I2C) Channel 1 Clock
00155 #define PINP2_12_PWMC0_PWML0 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 Channel 0 Output Low
00156 #define PINP2_12_TD (PinIO::PIN_FN_D)            // Peripheral D: SSC Transmit Data
00157 #else
00158 // Connector: P2 / Pin: 12 / CPU Port: PA5
00159 #define PINP2_12_PWMC1_PWML3 (PinIO::PIN_FN_A)   // Peripheral A: PWM 1 Channel 3 Output Low
00160 #define PINP2_12_ISI_D4 (PinIO::PIN_FN_B)        // Peripheral B: Image Sensor Channel 4 Data Input
00161 #define PINP2_12_URXD1 (PinIO::PIN_FN_C)         // Peripheral C: UART 1 Receive
00162 #define PINP2_12_SER_RX3 (PinIO::PIN_FN_C)       // Peripheral C: Serial Port 3 Receive
00163 #endif
00164
00165 // Connector: P2 / Pin: 13 / CPU Port: PA8
00166 #define PINP2_13_IN (PinIO::PIN_FN_IN)           // GPIO Input
00167 #define PINP2_13_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00168 #define PINP2_13_PWMC1_PWMH3 (PinIO::PIN_FN_A)   // Peripheral A: PWM 1 Channel 3 Output High
00169 #define PINP2_13_AFE0_ADTRG (PinIO::PIN_FN_B)    // Peripheral B: AFE 0 ADC External Trigger
00170
00171 // Connector: P2 / Pin: 15 / CPU Port: PD24
00172 #define PINP2_15_IN (PinIO::PIN_FN_IN)           // GPIO Input
00173 #define PINP2_15_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00174 #define PINP2_15_PWMC0_PWML0 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 0 Output Low
00175 #define PINP2_15_RF (PinIO::PIN_FN_B)            // Peripheral B: SSC Receive Frame Sync
00176 #define PINP2_15_TCLK11 (PinIO::PIN_FN_C)        // Peripheral C: Timer 11 Clock Input
00177 #define PINP2_15_ISI_HSYNC (PinIO::PIN_FN_D)     // Peripheral D: Image Sensor Horizontal Sync
00178
00179 // Connector: P2 / Pin: 16 / CPU Port: PA28
00180 #define PINP2_16_IN (PinIO::PIN_FN_IN)            // GPIO Input
00181 #define PINP2_16_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00182 #define PINP2_16_DSR1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 DSR
00183 #define PINP2_16_TCLK1 (PinIO::PIN_FN_B)          // Peripheral B: Timer 1 Clock
00184 #define PINP2_16_MCCDA (PinIO::PIN_FN_C)          // Peripheral C: MUltimedia Card Slot A Data Command
00185 #define PINP2_16_PWMC1_PWMFI2 (PinIO::PIN_FN_D)   // Peripheral D: PWM 1 Fault Input 2
00186
00187 // Connector: P2 / Pin: 17 / CPU Port: PA26
00188 #define PINP2_17_IN (PinIO::PIN_FN_IN)            // GPIO Input
00189 #define PINP2_17_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00190 #define PINP2_17_DCD1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 DCD
00191 #define PINP2_17_TIOA2 (PinIO::PIN_FN_B)          // Peripheral B: Timer 2 Line A
00192 #define PINP2_17_MCDA2 (PinIO::PIN_FN_C)          // Peripheral C: Multimedia Card Slot A Data 2
00193 #define PINP2_17_PWMC1_PWMFI1 (PinIO::PIN_FN_D)   // Peripheral D: PWM 1 Fault Input 1
00194
00195 // Connector: P2 / Pin: 18 / CPU Port: PA27
00196 #define PINP2_18_IN (PinIO::PIN_FN_IN)      // GPIO Input
00197 #define PINP2_18_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00198 #define PINP2_18_DTR1 (PinIO::PIN_FN_A)     // Peripheral A: USART 1 DTR
00199 #define PINP2_18_TIOB2 (PinIO::PIN_FN_B)    // Peripheral B: Timer 2 Line B
00200 #define PINP2_18_MCDA3 (PinIO::PIN_FN_C)    // Peripheral C: Multimedia Card Slot A Data 3
00201 #define PINP2_18_ISI_D7 (PinIO::PIN_FN_D)   // Peripheral D: Image Sensor Data Input 7
00202
00203 // Connector: P2 / Pin: 19 / CPU Port: PA1
00204 #define PINP2_19_IN (PinIO::PIN_FN_IN)           // GPIO Input
```

```
00205 #define PINP2_19_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00206 #define PINP2_19_PWMC0_PWML0 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 0 Output Low
00207 #define PINP2_19_TIOB0 (PinIO::PIN_FN_B)          // Peripheral B: Timer 0 Line B
00208 #define PINP2_19_A18 (PinIO::PIN_FN_C)            // Peripheral C: External Bus Interface A18
00209 #define PINP2_19_I2SC0_CK (PinIO::PIN_FN_D)       // Peripheral D: Sound Controller 0 Serial Clock
00210
00211 // Connector: P2 / Pin: 20 / CPU Port: PA29
00212 #define PINP2_20_IN (PinIO::PIN_FN_IN)      // GPIO Input
00213 #define PINP2_20_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00214 #define PINP2_20_RI1 (PinIO::PIN_FN_A)      // Peripheral A: USART 1 RI
00215 #define PINP2_20_TCLK2 (PinIO::PIN_FN_B)    // Peripheral B: Timer 2 Clock
00216
00217 // Connector: P2 / Pin: 21 / CPU Port: PA21
00218 #define PINP2_21_IN (PinIO::PIN_FN_IN)             // GPIO Input
00219 #define PINP2_21_OUT (PinIO::PIN_FN_OUT)           // GPIO Output
00220 #define PINP2_21_RXD1 (PinIO::PIN_FN_A)            // Peripheral A: USART 1 RX
00221 #define PINP2_21_SER_RX1 (PinIO::PIN_FN_A)         // Peripheral A: Serial Port 1 RX
00222 #define PINP2_21_PCK1 (PinIO::PIN_FN_B)            // Peripheral B: Programmable Clock Output 1
00223 #define PINP2_21_PWMC1_PWMFI0 (PinIO::PIN_FN_C)    // Peripheral C: PWM Clock 1 Channel 0 Fault Input
00224
00225 // Connector: P2 / Pin: 22 / CPU Port: PB4
00226 #define PINP2_22_IN (PinIO::PIN_FN_IN)            // GPIO Input
00227 #define PINP2_22_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00228 #define PINP2_22_TWD1 (PinIO::PIN_FN_A)           // Peripheral A: Two-wire (I2C) Channel 1 Data
00229 #define PINP2_22_PWMC0_PWMH2 (PinIO::PIN_FN_B)    // Peripheral B: PWM 0 Channel 2 Output High
00230 #define PINP2_22_TXD1 (PinIO::PIN_FN_D)           // Peripheral D: USART 1 Transmit
00231 #define PINP2_22_SER_TX1 (PinIO::PIN_FN_D)        // Peripheral D: Serial Port 1 Transmit
00232
00233 // Connector: P2 / Pin: 23 / CPU Port: PD28
00234 #define PINP2_23_IN (PinIO::PIN_FN_IN)      // GPIO Input
00235 #define PINP2_23_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00236 #define PINP2_23_URXD3 (PinIO::PIN_FN_A)    // Peripheral A: UART 3 Receive
00237 #define PINP2_23_SER_RX5 (PinIO::PIN_FN_A)  // Peripheral A: Serial Port 5 Receive
00238 #define PINP2_23_CANRX1 (PinIO::PIN_FN_B)   // Peripheral B: CAN 1 Receive
00239 #define PINP2_23_TWCK2 (PinIO::PIN_FN_C)    // Peripheral C: Two-wire (I2C) 2 Clock
00240 #define PINP2_23_ISI_D9 (PinIO::PIN_FN_D)   // Peripheral D: Image Sensor Data Input 9
00241
00242 // Connector: P2 / Pin: 24 / CPU Port: PD31
00243 #define PINP2_24_IN (PinIO::PIN_FN_IN)       // GPIO Input
00244 #define PINP2_24_OUT (PinIO::PIN_FN_OUT)     // GPIO Output
00245 #define PINP2_24_QIO3 (PinIO::PIN_FN_A)      // Peripheral A: QSPI Quad Mode Data 3
00246 #define PINP2_24_UTXD3 (PinIO::PIN_FN_B)     // Peripheral B: UART 3 Transmit
00247 #define PINP2_24_SER_TX5 (PinIO::PIN_FN_B)   // Peripheral B: Serial Port 5 Transmit
00248 #define PINP2_24_PCK2 (PinIO::PIN_FN_C)      // Peripheral C: Programmable Clock 2 Output
00249 #define PINP2_24_ISI_D11 (PinIO::PIN_FN_D)   // Peripheral D: Image Sensor Data Input 11
00250
00251 // Connector: P2 / Pin: 25 / CPU Port: PD22
00252 #define PINP2_25_IN (PinIO::PIN_FN_IN)            // GPIO Input
00253 #define PINP2_25_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00254 #define PINP2_25_PWMC0_PWMH2 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 2 Output High
00255 #define PINP2_25_SPI0_SPCK (PinIO::PIN_FN_B)      // Peripheral B: SPI 0 Clock
00256 #define PINP2_25_TIOB11 (PinIO::PIN_FN_C)         // Peripheral C: Timer 11 Line B
00257 #define PINP2_25_ISI_D0 (PinIO::PIN_FN_D)         // Peripheral D: Image Sensor Data Input 0
00258
00259 // Connector: P2 / Pin: 26 / CPU Port: PD27
00260 #define PINP2_26_IN (PinIO::PIN_FN_IN)            // GPIO Input
00261 #define PINP2_26_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00262 #define PINP2_26_PWMC0_PWML3 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 3 Output Low
00263 #define PINP2_26_SPI0_NPCS3 (PinIO::PIN_FN_B)     // Peripheral B: SPI 0 Chip Select 3
00264 #define PINP2_26_TWD2 (PinIO::PIN_FN_C)           // Peripheral C: Two-wire (I2C) 2 Serial Data
00265 #define PINP2_26_ISI_D8 (PinIO::PIN_FN_D)         // Peripheral D: Image Sensor Data Input 8
00266
00267 // Connector: P2 / Pin: 27 / CPU Port: PD20
00268 #define PINP2_27_IN (PinIO::PIN_FN_IN)            // GPIO Input
00269 #define PINP2_27_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00270 #define PINP2_27_PWMC0_PWMH0 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 0 Output High
00271 #define PINP2_27_SPI0_MISO (PinIO::PIN_FN_B)      // Peripheral B: SPI 0 Master In Slave Out
00272 #define PINP2_27_GTSUCOMP (PinIO::PIN_FN_C)       // Peripheral C: TSU Timer Comparison Valid 1588
00273
00274 // Connector: P2 / Pin: 28 / CPU Port: PD21
00275 #define PINP2_28_IN (PinIO::PIN_FN_IN)            // GPIO Input
00276 #define PINP2_28_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00277 #define PINP2_28_PWMC0_PWMH1 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 1 Output High
00278 #define PINP2_28_SPI0_MOSI (PinIO::PIN_FN_B)      // Peripheral B: SPI 0 Master Out Slave In
00279 #define PINP2_28_TIOA11 (PinIO::PIN_FN_C)         // Peripheral C: Timer 11 Line A
00280 #define PINP2_28_ISI_D1 (PinIO::PIN_FN_D)         // Peripheral D: Image Sensor Data Input 1
00281
00282 // Connector: P2 / Pin: 29 / CPU Port: PB2
00283 #define PINP2_29_IN (PinIO::PIN_FN_IN)          // GPIO Input
00284 #define PINP2_29_OUT (PinIO::PIN_FN_OUT)        // GPIO Output
00285 #define PINP2_29_CANTX0 (PinIO::PIN_FN_A)       // Peripheral A: CAN 0 Transmit
00286 #define PINP2_29_CTS0 (PinIO::PIN_FN_C)         // Peripheral C: USART 0 CTS
00287 #define PINP2_29_SER_CTS0 (PinIO::PIN_FN_C)     // Peripheral C: Serial Port 0 CTS
00288 #define PINP2_29_SPI0_NPCS0 (PinIO::PIN_FN_D)   // Peripheral D: SPI 0 Chip Select 0
00289
00290 // Connector: P2 / Pin: 30 / CPU Port: PD12
00291 #define PINP2_30_IN (PinIO::PIN_FN_IN)          // GPIO Input
```

```
00292 #define PINP2_30_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00293 #define PINP2_30_GRX3 (PinIO::PIN_FN_A)          // Peripheral A: GMAC Receive Data 3
00294 #define PINP2_30_CANTX1 (PinIO::PIN_FN_B)        // Peripheral B: CAN 1 Transmit
00295 #define PINP2_30_SPI0_NPCS2 (PinIO::PIN_FN_C)    // Peripheral C: SPI 0 Chip Select 2
00296 #define PINP2_30_ISI_D6 (PinIO::PIN_FN_D)        // Peripheral D: Image Sensor Data Input 6
00297
00298 // Connector: P2 / Pin: 31 / CPU Port: PA23
00299 #define PINP2_31_IN (PinIO::PIN_FN_IN)            // GPIO Input
00300 #define PINP2_31_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00301 #define PINP2_31_SCK1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 Serial Clock
00302 #define PINP2_31_PWMC0_PWMH0 (PinIO::PIN_FN_B)    // Peripheral B: PWM Clock 0 Channel 0 Output High
00303 #define PINP2_31_A19 (PinIO::PIN_FN_C)            // Peripheral C: External Bus Interface A19
00304 #define PINP2_31_PWMC1_PWML2 (PinIO::PIN_FN_D)    // Peripheral D: PWM 1 Channel 2 Output Low
00305
00306 // Connector: P2 / Pin: 32 / CPU Port: PA24
00307 #define PINP2_32_IN (PinIO::PIN_FN_IN)            // GPIO Input
00308 #define PINP2_32_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00309 #define PINP2_32_RTS1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 RTS
00310 #define PINP2_32_SER_RTS1 (PinIO::PIN_FN_A)       // Peripheral A: Serial Port 1 RTS
00311 #define PINP2_32_PWMC0_PWMH1 (PinIO::PIN_FN_B)    // Peripheral B: PWM Clock 0 Channel 1 Output High
00312 #define PINP2_32_A20 (PinIO::PIN_FN_C)            // Peripheral C: External Bus Interface A20
00313 #define PINP2_32_ISI_PCK (PinIO::PIN_FN_D)        // Peripheral D: Image Sensor Data Clock
00314
00315 // Connector: P2 / Pin: 33 / CPU Port: PA25
00316 #define PINP2_33_IN (PinIO::PIN_FN_IN)            // GPIO Input
00317 #define PINP2_33_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00318 #define PINP2_33_CTS1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 CTS
00319 #define PINP2_33_SER_CTS1 (PinIO::PIN_FN_A)       // Peripheral A: Serial Port 1 CTS
00320 #define PINP2_33_PWMC0_PWMH2 (PinIO::PIN_FN_B)    // Peripheral B: PWM Clock 0 Channel 2 Output High
00321 #define PINP2_33_A23 (PinIO::PIN_FN_C)            // Peripheral C: External Bus Interface A23
00322 #define PINP2_33_MCCK (PinIO::PIN_FN_D)           // Peripheral D: Multimedia Card Clock
00323
00324 // Connector: P2 / Pin: 34 / CPU Port: PA9
00325 #define PINP2_34_IN (PinIO::PIN_FN_IN)             // GPIO Input
00326 #define PINP2_34_OUT (PinIO::PIN_FN_OUT)           // GPIO Output
00327 #define PINP2_34_URXD0 (PinIO::PIN_FN_A)           // Peripheral A: UART0 RX
00328 #define PINP2_34_SER_RX2 (PinIO::PIN_FN_A)         // Peripheral A: Serial Port 2 RX
00329 #define PINP2_34_ISI_D3 (PinIO::PIN_FN_B)          // Peripheral B: Image Sensor Channel 3 Data Input
00330 #define PINP2_34_PWMC0_PWMFI0 (PinIO::PIN_FN_C)    // Peripheral C: PWM 0 Fault Input 0
00331
00332 // Connector: P2 / Pin: 35 / CPU Port: PA10
00333 #define PINP2_35_IN (PinIO::PIN_FN_IN)                // GPIO Input
00334 #define PINP2_35_OUT (PinIO::PIN_FN_OUT)              // GPIO Output
00335 #define PINP2_35_UTXD0 (PinIO::PIN_FN_A)              // Peripheral A: UART0 TX
00336 #define PINP2_35_SER_TX2 (PinIO::PIN_FN_A)            // Peripheral A: Serial Port 2 TX
00337 #define PINP2_35_PWMC0_PWMEXTRG0 (PinIO::PIN_FN_B)    // Peripheral B: PWM 0 External Trigger 0
00338 #define PINP2_35_RD (PinIO::PIN_FN_C)                 // Peripheral C: SSC Receive Data
00339
00340 // Connector: P2 / Pin: 36 / CPU Port: PA30
00341 #define PINP2_36_IN (PinIO::PIN_FN_IN)                // GPIO Input
00342 #define PINP2_36_OUT (PinIO::PIN_FN_OUT)              // GPIO Output
00343 #define PINP2_36_PWMC0_PWML2 (PinIO::PIN_FN_A)        // Peripheral A: PWM Clock 0 Channel 2 Output Low
00344 #define PINP2_36_PWMC1_PWMEXTRG0 (PinIO::PIN_FN_B)    // Peripheral B: PWM Clock 1 Channel 0 Trigger
      Input
00345 #define PINP2_36_MCDA0 (PinIO::PIN_FN_C)              // Peripheral C: Multimedia Card Slot A Data 0
00346 #define PINP2_36_I2SC0_DO (PinIO::PIN_FN_D)           // Peripheral D: Sounds Controller 0 Date Output
00347
00348 // Connector: P2 / Pin: 37 / CPU Port: PD11
00349 #define PINP2_37_IN (PinIO::PIN_FN_IN)            // GPIO Input
00350 #define PINP2_37_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00351 #define PINP2_37_GRX2 (PinIO::PIN_FN_A)           // Peripheral A: GMAC Receive Data 2
00352 #define PINP2_37_PWMC0_PWMH0 (PinIO::PIN_FN_B)    // Peripheral B: PWM Clock 0 Channel 0 Output High
00353 #define PINP2_37_GTSUCOMP (PinIO::PIN_FN_C)       // Peripheral C: TSU Timer Comparison Valid 1588
00354 #define PINP2_37_ISI_D5 (PinIO::PIN_FN_D)         // Peripheral D: Image Sensor Data Input 5
00355
00356 // Connector: P2 / Pin: 38 / CPU Port: PB3
00357 #define PINP2_38_IN (PinIO::PIN_FN_IN)       // GPIO Input
00358 #define PINP2_38_OUT (PinIO::PIN_FN_OUT)     // GPIO Output
00359 #define PINP2_38_CANRX0 (PinIO::PIN_FN_A)    // Peripheral A: CAN 0 Receive
00360 #define PINP2_38_PCK2 (PinIO::PIN_FN_B)      // Peripheral B: Programmable Clock Output 2
00361 #define PINP2_38_RTS0 (PinIO::PIN_FN_C)      // Peripheral C: USART 0 RTS
00362 #define PINP2_38_SER_RTS0 (PinIO::PIN_FN_C)  // Peripheral C: Serial Port 0 RTS
00363 #define PINP2_38_ISI_D2 (PinIO::PIN_FN_D)    // Peripheral D: Image Sensor Data Input 2
00364
00365 // Connector: P2 / Pin: 39 / CPU Port: PA3
00366 #define PINP2_39_IN (PinIO::PIN_FN_IN)        // GPIO Input
00367 #define PINP2_39_OUT (PinIO::PIN_FN_OUT)      // GPIO Output
00368 #define PINP2_39_TWD0 (PinIO::PIN_FN_A)       // Peripheral A: Two-wire (I2C) 0 Data
00369 #define PINP2_39_LONCOL1 (PinIO::PIN_FN_B)    // Peripheral B: LON Channel 1 Collision Detect
00370 #define PINP2_39_PCK2 (PinIO::PIN_FN_C)       // Peripheral C: Programmable Clock Channel 2 Output
00371
00372 // Connector: P2 / Pin: 40 / CPU Port: PA31
00373 #define PINP2_40_IN (PinIO::PIN_FN_IN)            // GPIO Input
00374 #define PINP2_40_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00375 #define PINP2_40_SPI0_NPCS1 (PinIO::PIN_FN_A)     // Peripheral A: SPI 0 Chip Select 1
00376 #define PINP2_40_PCK2 (PinIO::PIN_FN_B)           // Peripheral B: Programmable Clock Output 2
00377 #define PINP2_40_MCDA1 (PinIO::PIN_FN_C)          // Peripheral C: Multimedia Card Slot A Data 1
```

```
00378 #define PINP2_40_PWMC1_PWMH2 (PinIO::PIN_FN_D)   // Peripheral D: PWM Clock 1 Channel 2 Output High
00379
00380 // Connector: P2 / Pin: 41 / CPU Port: PD25
00381 #define PINP2_41_IN (PinIO::PIN_FN_IN)           // GPIO Input
00382 #define PINP2_41_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00383 #define PINP2_41_PWMC0_PWML1 (PinIO::PIN_FN_A)   // Peripheral A: PWM Clock 0 Channel 1 Output Low
00384 #define PINP2_41_SPI0_NPCS1 (PinIO::PIN_FN_B)    // Peripheral B: SPI 0 Chip Select 1
00385 #define PINP2_41_URXD2 (PinIO::PIN_FN_C)         // Peripheral C: UART 2 Receive
00386 #define PINP2_41_SER_RX4 (PinIO::PIN_FN_C)       // Peripheral C: Serial Port 4 Receive
00387 #define PINP2_41_ISI_VSYNC (PinIO::PIN_FN_D)     // Peripheral D: Image Sensor Vertical Sync
00388
00389 // Connector: P2 / Pin: 42 / CPU Port: PA4
00390 #define PINP2_42_IN (PinIO::PIN_FN_IN)     // GPIO Input
00391 #define PINP2_42_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00392 #define PINP2_42_TWCK0 (PinIO::PIN_FN_A)   // Peripheral A: Two Wire (I2C) 0 Data
00393 #define PINP2_42_TCLK0 (PinIO::PIN_FN_B)   // Peripheral B: Timer 0 Clock
00394 #define PINP2_42_UTXD1 (PinIO::PIN_FN_C)   // Peripheral C: UART 1 Transmit
00395 #define PINP2_42_SER_TX3 (PinIO::PIN_FN_C) // Peripheral C: Serial Port 3 Transmit
00396
00397 // Connector: P2 / Pin: 43 / CPU Port: PA13
00398 #define PINP2_43_IN (PinIO::PIN_FN_IN)           // GPIO Input
00399 #define PINP2_43_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00400 #define PINP2_43_QIO0 (PinIO::PIN_FN_A)          // Peripheral A: QSPI MOSI Single Bit Mode, Data 0
      Quad Mode
00401 #define PINP2_43_PWMC0_PWMH2 (PinIO::PIN_FN_B)   // Peripheral B: PWM Clock 0 Channel 2 Output High
00402 #define PINP2_43_PWMC1_PWML1 (PinIO::PIN_FN_C)   // Peripheral C: PWM Clock 1 Channel 1 Output Low
00403
00404 // Connector: P2 / Pin: 44 / CPU Port: PD26
00405 #define PINP2_44_IN (PinIO::PIN_FN_IN)           // GPIO Input
00406 #define PINP2_44_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00407 #define PINP2_44_PWMC0_PWML2 (PinIO::PIN_FN_A)   // Peripheral A: PWM Clock 0 Channel 2 Output Low
00408 #define PINP2_44_TD (PinIO::PIN_FN_B)            // Peripheral B: SSC Transmit Data
00409 #define PINP2_44_UTXD2 (PinIO::PIN_FN_C)         // Peripheral C: UART 2 Transmit
00410 #define PINP2_44_SER_TX4 (PinIO::PIN_FN_C)       // Peripheral C: Serial Port 4 Transmit
00411 #define PINP2_44_UTXD1 (PinIO::PIN_FN_D)         // Peripheral D: UART 1 Transmit
00412 #define PINP2_44_SER_TX3 (PinIO::PIN_FN_D)       // Peripheral D: Serial Port 3 Transmit
00413
00414 // Connector: P2 / Pin: 45 / CPU Port: PA14
00415 #define PINP2_45_IN (PinIO::PIN_FN_IN)           // GPIO Input
00416 #define PINP2_45_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00417 #define PINP2_45_QSCK (PinIO::PIN_FN_A)          // Peripheral A: QSPI Serial Clock (QSCK)
00418 #define PINP2_45_PWMC0_PWMH3 (PinIO::PIN_FN_B)   // Peripheral B: PWM Clock 0 Channel 3 Output High
00419 #define PINP2_45_PWMC1_PWMH1 (PinIO::PIN_FN_C)   // Peripheral C: PWM Clock 1 Channel 1 Output High
00420
00421 // Connector: P2 / Pin: 47 / CPU Port: PA12
00422 #define PINP2_47_IN (PinIO::PIN_FN_IN)           // GPIO Input
00423 #define PINP2_47_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00424 #define PINP2_47_QIO1 (PinIO::PIN_FN_A)          // Peripheral A: QSPI MISO Single Bit Mode, Data 1
      Quad Mode
00425 #define PINP2_47_PWMC0_PWMH1 (PinIO::PIN_FN_B)   // Peripheral B: PWM Clock 0 Channel 1 Output High
00426 #define PINP2_47_PWMC1_PWMH0 (PinIO::PIN_FN_C)   // Peripheral C: PWM Clock 1 Channel 0 Output High
00427
00428 // Connector: P2 / Pin: 48 / CPU Port: PA11
00429 #define PINP2_48_IN (PinIO::PIN_FN_IN)           // GPIO Input
00430 #define PINP2_48_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00431 #define PINP2_48_QCS (PinIO::PIN_FN_A)           // Peripheral A: QSPI Chip Select
00432 #define PINP2_48_PWMC0_PWMH0 (PinIO::PIN_FN_B)   // Peripheral B: PWM Clock 0 Channel 0 Output High
00433 #define PINP2_48_PWMC1_PWML0 (PinIO::PIN_FN_C)   // Peripheral C: PWM CLock 1 Channel 0 Output Low
00434
00435 #endif   // MODM7AE70_PIN_CONSTANT
```

24.531 NANO54415/include/pinconstant.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef NANO54415_PIN_CONSTANT
00006 #define NANO54415_PIN_CONSTANT
00007
00008 #define PIN_GPIO (0)   // Universal GPIO Definition
00009
00010 // Pin: P1-1  / CPU Pin: J1  / Reserved For: ADC_IN1
00011
00012 // Pin: P1-2  / CPU Pin: H1  / Reserved For: ADC_IN0
00013
00014 // Pin: P1-3  / CPU Pin: K4  / Reserved For: ADC_IN3
00015
00016 // Pin: P1-4  / CPU Pin: J2  / Reserved For: ADC_IN2
00017
00018 // Pin: P1-5  / CPU Pin: K3  / Reserved For: ADC_IN7
00019
00020 // Pin: P1-6  / CPU Pin: G4  / Reserved For: ADC_IN4
00021
00022 // Pin: P1-7  / CPU Pin: H4 / Reserved For: VDDA_ADC
```

```
00023 // Pin: P1-7  / CPU Pin: J4 / Reserved For: VDDA_DAC_ADC
00024
00025 // Pin: P1-8  / CPU Pin: NA  / Reserved For: AGND
00026
00027 // Pin: P1-9  / CPU Pin: F12
00028 #define PIN_9_IRQ7 (1)   // Primary Function: External Interrupt 7
00029 #define PIN_9_GPIO (0)   // Port C - 6
00030
00031 // Pin: P1-10 / CPU Pin: M4
00032 #define PIN_10_UART2_CTS (3)   // Primary Function: UART 2 - Clear to Send
00033 #define PIN_10_UART6_TXD (2)   // Alternate Function 1: UART 6 - Transmit
00034 #define PIN_10_SSI1_BCLK (1)   // Alternate Function 2: SSI 1 - Serial Bit Clock
00035 #define PIN_10_GPIO (0)        // Port E - 6 / Rapid GPIO
00036
00037 // Pin: P1-11 / CPU Pin: K15 / Reserved For: RESET
00038
00039 // Pin: P1-12 / CPU Pin: N11
00040 #define PIN_12_OW_DAT (3)   // Primary Function: 1-Wire Data Signal
00041 #define PIN_12_DACK0 (2)    // Alternate Function: DMA Acknowledge 0
00042 #define PIN_12_GPIO (0)     // Port D - 3 / Rapid GPIO 0
00043
00044 // Pin: P1-13 / CPU Pin: P1
00045 #define PIN_13_UART2_RXD (3)   // Primary Function: UART 2 - Receive
00046 #define PIN_13_PWM_A3 (2)      // Alternate Function 1: PWM A3 - Output Signal/Input Capture
00047 #define PIN_13_SSI1_RXD (1)    // Alternate Function 2: SSI 1 - Serial Receive Data
00048 #define PIN_13_GPIO (0)        // Port E - 4
00049
00050 // Pin: P1-14 / CPU Pin: M3
00051 #define PIN_14_UART2_RTS (3)   // Primary Function: UART 2 - Request to Send
00052 #define PIN_14_UART6_RXD (2)   // Alternate Function 1: UART 6 - Receive
00053 #define PIN_14_SSI1_FS (1)     // Alternate Function 2: SSI 1 - Serial Frame Sync
00054 #define PIN_14_GPIO (0)        // Port E - 5 / Rapid GPIO 15
00055
00056 // Pin: P1-15 / CPU Pin: B13
00057 #define PIN_15_SDHC_DAT3 (3)    // Primary Function: SDHC DAT3 Line / Card Detection
00058 #define PIN_15_PWM_A1 (2)       // Alternate Function 1: PWM A1 - Output Signal/Input Capture
00059 #define PIN_15_DSPI1_PCS0 (1)   // Alternate Function 2: DSPI 1 - Peripheral Chip Select 0
00060 #define PIN_15_GPIO (0)         // Port F - 2
00061
00062 // Pin: P1-16 / CPU Pin: N2
00063 #define PIN_16_UART2_TXD (3)   // Primary Function: UART 2 - Transmit
00064 #define PIN_16_PWM_B3 (2)      // Alternate Function 1: PWM B3 - Output Signal/Input Capture
00065 #define PIN_16_SSI1_TXD (1)    // Alternate Function 2: SSI 1 - Serial Transmit Data
00066 #define PIN_16_GPIO (0)        // Port E - 3
00067
00068 // Pin: P1-17 / CPU Pin: NA / Reserved For: VCC3V
00069
00070 // Pin: P1-18 / CPU Pin: NA / Reserved For: GND
00071
00072 // Pin: P1-19 / CPU Pin: H15
00073 #define PIN_19_T0IN (3)           // Primary Function: Timer Input 0
00074 #define PIN_19_T0OUT (2)          // Alternate Function 1: Timer Output 0
00075 #define PIN_19_USB0_VBUS_OC (1)   // Alternate Function 2: USB On-the-Go VBUS Over-Current
00076 #define PIN_19_GPIO (0)           // Port E - 7 / Rapid GPIO 4
00077
00078 // Pin: P1-20 / CPU Pin: D15
00079 #define PIN_20_CAN1_RX (3)     // Primary Function: CAN 1 - Receive
00080 #define PIN_20_UART9_RXD (2)   // Alternate Function 1: UART 9 - Receive
00081 #define PIN_20_I2C1_SDA (1)    // Alternate Function 2: I2C 1 - Serial Data
00082 #define PIN_20_GPIO (0)        // Port C - 7
00083
00084 // Pin: P1-21 / CPU Pin: H13
00085 #define PIN_21_T1IN (3)        // Primary Function: Timer Input 1
00086 #define PIN_21_T1OUT (2)       // Alternate Function 1: Timer Output 1
00087 #define PIN_21_SDHC_DAT1 (1)   // Alternate Function 2: SDHC DAT1 Line / Interrupt Detect
00088 #define PIN_21_GPIO (0)        // Port D - 0 / Rapid GPIO 3
00089
00090 // Pin: P1-22 / CPU Pin: D14
00091 #define PIN_22_CAN1_TX (3)     // Primary Function: CAN 1 - Transmit
00092 #define PIN_22_UART9_TXD (2)   // Alternate Function 1: UART 9 - Transmit
00093 #define PIN_22_I2C1_SCL (1)    // Alternate Function 2: I2C 1 - Serial Clock
00094 #define PIN_22_GPIO (0)        // Port B - 0
00095
00096 // Pin: P1-23 / CPU Pin: H14
00097 #define PIN_23_T2IN (3)        // Primary Function: Timer Input 2
00098 #define PIN_23_T2OUT (2)       // Alternate Function 1: Timer Output 2
00099 #define PIN_23_SDHC_DAT2 (1)   // Alternate Function 2: SDHC DAT2 Line / Read Wait
00100 #define PIN_23_GPIO (0)        // Port D - 1 / Rapid GPIO 2
00101
00102 // Pin: P1-24 / CPU Pin: B10
00103 #define PIN_24_UART0_RXD (3)   // Primary Function: UART 0 - Receive
00104 #define PIN_24_I2C4_SDA (2)    // Alternate Function 1: I2C 4 - Serial Data
00105 #define PIN_24_DSPI2_SIN (1)   // Alternate Function 2: DSPI 2 - Serial Data In
00106 #define PIN_24_GPIO (0)        // Port F - 4
00107
00108 // Pin: P1-25 / CPU Pin: G13
00109 #define PIN_25_T3IN (3)           // Primary Function: Timer Input 3
```

```
00110 #define PIN_25_T3OUT (2)          // Alternate Function 1: Timer Output 3
00111 #define PIN_25_USBO_VBUS_EN (1)   // Alternate Function 2: USB On-the-Go VBUS Enable
00112 #define PIN_25_GPIO (0)           // Port D - 2 / Rapid GPIO 1
00113
00114 // Pin: P1-26 / CPU Pin: D11
00115 #define PIN_26_UART0_TXD (3)     // Primary Function: UART 0 - Transmit
00116 #define PIN_26_I2C4_SCL (2)      // Alternate Function 1: I2C 4 - Serial Clock
00117 #define PIN_26_DSPI2_SOUT (1)    // Alternate Function 2: DSPI 2 - Serial Data Out
00118 #define PIN_26_GPIO (0)          // Port F - 3
00119
00120 // Pin: P1-27 / CPU Pin: G15
00121 #define PIN_27_I2C0_SCL (3)     // Primary Function: I2C 0 - Serial Clock
00122 #define PIN_27_UART8_TXD (2)    // Alternate Function 1: UART 8 - Transmit
00123 #define PIN_27_CAN0_TX (1)      // Alternate Function 2: CAN 0 - Transmit
00124 #define PIN_27_GPIO (0)         // Port B - 2
00125
00126 // Pin: P1-28 / CPU Pin: B11
00127 #define PIN_28_UART0_RTS (3)     // Primary Function: UART 0 - Request to Send
00128 #define PIN_28_UART4_RXD (2)     // Alternate Function 1: UART 4 - Receive
00129 #define PIN_28_DSPI2_PCS0 (1)    // Alternate Function 2: DSPI 2 - Peripheral Chip Select 0
00130 #define PIN_28_GPIO (0)          // Port F - 5 / Rapid GPIO 6
00131
00132 // Pin: P1-29 / CPU Pin: G14
00133 #define PIN_29_I2C0_SDA (3)     // Primary Function: I2C 0 - Serial Data
00134 #define PIN_29_UART8_RXD (2)    // Alternate Function 1: UART 8 - Receive
00135 #define PIN_29_CAN0_RX (1)      // Alternate Function 2: CAN 0 - Receive
00136 #define PIN_29_GPIO (0)         // Port B - 1
00137
00138 // Pin: P1-30 / CPU Pin: E13
00139 #define PIN_30_UART0_CTS (3)    // Primary Function: UART 0 - Clear to Send
00140 #define PIN_30_UART4_TXD (2)    // Alternate Function 1: UART 4 - Transmit
00141 #define PIN_30_DSPI2_SCK (1)    // Alternate Function 2: DSPI 2 - Serial Clock
00142 #define PIN_30_GPIO (0)         // Port F - 6 / Rapid GPIO 5
00143
00144 // Pin: P1-31 / CPU Pin: A10
00145 #define PIN_31_SDHC_CLK (3)     // Primary Function: SDHC Clock
00146 #define PIN_31_PWM_A0 (2)       // Alternate Function 1: PWM A0 - Output Signal/Input Capture
00147 #define PIN_31_DSPI1_SCK (1)    // Alternate Function 2: DSPI 1 - Serial Clock
00148 #define PIN_31_GPIO (0)         // Port G - 5
00149
00150 // Pin: P1-32 / CPU Pin: C9
00151 #define PIN_32_UART1_RXD (3)    // Primary Function: UART 1 - Receive
00152 #define PIN_32_I2C5_SDA (2)     // Alternate Function 1: I2C 5 - Serial Data
00153 #define PIN_32_DSPI3_SIN (1)    // Alternate Function 2: DSPI 3 - Serial Data In
00154 #define PIN_32_GPIO (0)         // Port E - 0
00155
00156 // Pin: P1-33 / CPU Pin: C11
00157 #define PIN_33_SDHC_CMD (3)     // Primary Function: SDHC Command Line
00158 #define PIN_33_PWM_B0 (2)       // Alternate Function 1: PWM B0 - Output Signal/Input Capture
00159 #define PIN_33_DSPI1_SIN (1)    // Alternate Function 2: DSPI 1 - Serial Data In
00160 #define PIN_33_GPIO (0)         // Port G - 6
00161
00162 // Pin: P1-34 / CPU Pin: D9
00163 #define PIN_34_UART1_TXD (3)     // Primary Function: UART 1 - Transmit
00164 #define PIN_34_I2C5_SCL (2)      // Alternate Function 1: I2C 5 - Serial Clock
00165 #define PIN_34_DSPI3_SOUT (1)    // Alternate Function 2: DSPI 3 - Serial Data Out
00166 #define PIN_34_GPIO (0)          // Port F - 7
00167
00168 // Pin: P1-35 / CPU Pin: B12
00169 #define PIN_35_SDHC_DAT0 (3)     // Primary Function: SDHC DAT0 Line / Busy-State Detect
00170 #define PIN_35_PWM_B2 (2)        // Alternate Function 1: PWM B2 - Output Signal/Input Capture
00171 #define PIN_35_DSPI1_SOUT (1)    // Alternate Function 2: DSPI 1 - Serial Data Out
00172 #define PIN_35_GPIO (0)          // Port G - 7
00173
00174 // Pin: P1-36 / CPU Pin: D10
00175 #define PIN_36_UART1_RTS (3)     // Primary Function: UART 1 - Request to Send
00176 #define PIN_36_UART5_RXD (2)     // Alternate Function 1: UART 5 - Receive
00177 #define PIN_36_DSPI3_PCS0 (1)    // Alternate Function 2: DSPI 3 - Peripheral Chip Select 0
00178 #define PIN_36_GPIO (0)          // Port E - 1 / Rapid GPIO 8
00179
00180 // Pin: P1-37 / CPU Pin: D12
00181 #define PIN_37_SDHC_DAT1 (3)     // Primary Function: SDHC DAT1 Line / Interrupt Detect
00182 #define PIN_37_PWM_A2 (2)        // Alternate Function 1: PWM A2 - Output Signal/Input Capture
00183 #define PIN_37_DSPI1_PCS1 (1)    // Alternate Function 2: DSPI 1 - Peripheral Chip Select 1
00184 #define PIN_37_GPIO (0)          // Port F - 0
00185
00186 // Pin: P1-38 / CPU Pin: C10
00187 #define PIN_38_UART1_CTS (3)    // Primary Function: UART 1 - Clear to Send
00188 #define PIN_38_UART5_TXD (2)    // Alternate Function 1: UART 5 - Transmit
00189 #define PIN_38_DSPI3_SCK (1)    // Alternate Function 2: DSPI 3 - Serial Clock
00190 #define PIN_38_GPIO (0)         // Port E - 2 / Rapid GPIO 7
00191
00192 // Pin: P1-39 / CPU Pin: E14
00193 #define PIN_39_SDHC_DAT2 (3)     // Primary Function: SDHC DAT2 Line / Read Wait
00194 #define PIN_39_PWM_B1 (2)        // Alternate Function 1: PWM B1 - Output Signal/Input Capture
00195 #define PIN_39_DSPI1_PCS2 (1)    // Alternate Function 2: DSPI 1 - Peripheral Chip Select 2
00196 #define PIN_39_GPIO (0)          // Port F - 1
```

```
00197
00198 // Pin: P1-40 / CPU Pin: NA / Reserved For: SPEEDLED
00199
00200 // Pin: P1-41 / CPU Pin: NA / Reserved For: VCC3VE
00201
00202 // Pin: P1-42 / CPU Pin: NA / Reserved For: LINKLED
00203
00204 // Pin: P1-43 / CPU Pin: NA / Reserved For: ERXJP
00205
00206 // Pin: P1-44 / CPU Pin: NA / Reserved For: ERXJM
00207
00208 // Pin: P1-45 / CPU Pin: NA / Reserved For: ETXJP
00209
00210 // Pin: P1-46 / CPU Pin: NA / Reserved For: ETXJM
00211
00212 // Pin: P1-47 / CPU Pin: NA / Reserved For: VCC3V
00213
00214 // Pin: P1-48 / CPU Pin: NA / Reserved For: GND
00215
00216 // Pin: P1-49 / CPU Pin: M1
00217 #define PIN_49_IRQ3 (3)           // Primary Function: External Interrupt 3
00218 #define PIN_49_DSPI0_PCS3 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 3
00219 #define PIN_49_USBH_VBUS_EN (1)   // Alternate Function 2: USB Host VBUS Enable
00220 #define PIN_49_GPIO (0)           // Port C - 3
00221
00222 // Pin: P1-50 / CPU Pin: M2
00223 #define PIN_50_IRQ2 (3)           // Primary Function: External Interrupt 2
00224 #define PIN_50_DSPI0_PCS2 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 2
00225 #define PIN_50_USBH_VBUS_OC (1)   // Alternate Function 2: USB Host VBUS Over-Current
00226 #define PIN_50_GPIO (0)           // Port C - 2
00227
00228 // Pin: P1-51 / CPU Pin: A14 / Reserved For: USB On-the-Go Data Minus (Stuffed by Default)
00229 // Pin: P1-51 / CPU Pin: A15 / Reserved For: USB Host Data Minus
00230
00231 // Pin: P1-52 / CPU Pin: B14 / Reserved For: USB On-the-Go Data Plus (Stuffed by Default)
00232 // Pin: P1-52 / CPU Pin: B15 / Reserved For: USB Host Data Plus
00233
00234 #endif /* NANO54415_PIN_CONSTANT */
```

24.532 SB800EX/include/pinconstant.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef SB800EX_PIN_CONSTANT
00006 #define SB800EX_PIN_CONSTANT
00007
00008 #define PIN_GPIO (0)   // Universal GPIO Definition
00009
00010 // Pin: JP3-1 / CPU Pin: G13
00011 #define PIN_1_T3IN (3)           // Primary Function: Timer Input 3
00012 #define PIN_1_T3OUT (2)          // Alternate Function 1: Timer Output 3
00013 #define PIN_1_USBO_VBUS_EN (1)   // Alternate Function 2: USB On-the-Go VBUS Enable
00014 #define PIN_1_GPIO (0)           // Port D - 2 / Rapid GPIO 1
00015
00016 // Pin: JP3-2 / CPU Pin: B11
00017 #define PIN_2_UART0_RTS (3)    // Primary Function: UART 0 - Request to Send
00018 #define PIN_2_UART4_RXD (2)    // Alternate Function 1: UART 4 - Receive
00019 #define PIN_2_DSPI2_PCS0 (1)   // Alternate Function 2: DSPI 2 - Peripheral Chip Select 0
00020 #define PIN_2_GPIO (0)         // Port F - 5 / Rapid GPIO 6
00021
00022 // Pin: JP3-3 / CPU Pin: E13
00023 #define PIN_3_UART0_CTS (3)   // Primary Function: UART 0 - Clear to Send
00024 #define PIN_3_UART4_TXD (2)   // Alternate Function 1: UART 4 - Transmit
00025 #define PIN_3_DSPI2_SCK (1)   // Alternate Function 2: DSPI 2 - Serial Clock
00026 #define PIN_3_GPIO (0)        // Port F - 6 / Rapid GPIO 5
00027
00028 // Pin: JP3-4 / CPU Pin: B10
00029 #define PIN_4_UART0_RXD (3)   // Primary Function: UART 0 - Receive
00030 #define PIN_4_I2C4_SDA (2)    // Alternate Function 1: I2C 4 - Serial Data
00031 #define PIN_4_DSPI2_SIN (1)   // Alternate Function 2: DSPI 2 - Serial Data In
00032 #define PIN_4_GPIO (0)        // Port F - 4
00033
00034 // Pin: JP3-5 / CPU Pin: D11
00035 #define PIN_5_UART0_TXD (3)    // Primary Function: UART 0 - Transmit
00036 #define PIN_5_I2C4_SCL (2)     // Alternate Function 1: I2C 4 - Serial Clock
00037 #define PIN_5_DSPI2_SOUT (1)   // Alternate Function 2: DSPI 2 - Serial Data Out
00038 #define PIN_5_GPIO (0)         // Port F - 3
00039
00040 // Pin: JP3-6 / CPU Pin: E12
00041 #define PIN_6_SIM0_DATA (3)    // Primary Function:
00042 #define PIN_6_PWM_FAULT2 (2)   // Alternate Function 1:
00043 #define PIN_6_SDHC_DAT7 (1)    // Alternate Function 2:
00044 #define PIN_6_GPIO (0)         // Port G - 4
```

```
00045
00046 // Pin: JP3-7 / CPU Pin: M1
00047 #define PIN_7_IRQ3 (3)           // Primary Function: External Interrupt 3
00048 #define PIN_7_DSPI0_PCS3 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 3
00049 #define PIN_7_USBH_VBUS_EN (1)   // Alternate Function 2: USB Host VBUS Enable
00050 #define PIN_7_GPIO (0)           // Port C - 3
00051
00052 // Pin: JP3-8 / CPU Pin: NA / Reserved For: VCC3V
00053
00054 // Pin: JP3-9 / CPU Pin: NA / Reserved For: GND
00055
00056 // CPU Pin: M1
00057 #define PORTC_3_IRQ3 (3)           // Primary Function: External Interrupt 3
00058 #define PORTC_3_DSPI0_PCS3 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 3
00059 #define PORTC_3_USBH_VBUS_EN (1)   // Alternate Function 2: USB Host VBUS Enable
00060 #define PORTC_3_GPIO (0)           // Port C - 3
00061
00062 // CPU Pin: M2
00063 #define PORTC_2_IRQ2 (3)           // Primary Function: External Interrupt 2
00064 #define PORTC_2_DSPI0_PCS2 (2)     // Alternate Function 1: DSPI 0 - Peripheral Chip Select 2
00065 #define PORTC_2_USBH_VBUS_OC (1)   // Alternate Function 2: USB Host VBUS Over-Current
00066 #define PORTC_2_GPIO (0)           // Port C - 2
00067 #endif
```

24.533 SBE70LC/include/pinconstant.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef SBE70LC_PIN_CONSTANT
00006 #define SBE70LC_PIN_CONSTANT
00007
00008 #define PIN_GPIO (PinIO::PIN_FN_IN)
00009 #define PIN_GPIO_OUT (PinIO::PIN_FN_OUT)
00010
00011 // Connector: J1 / Pin: 3 / CPU Port: PD12
00012 #define PINJ1_3_IN (PinIO::PIN_FN_IN)          // GPIO Input
00013 #define PINJ1_3_OUT (PinIO::PIN_FN_OUT)        // GPIO Output
00014 #define PINJ1_3_GRX3 (PinIO::PIN_FN_A)         // Peripheral A: GMAC Receive Data 3
00015 #define PINJ1_3_CANTX1 (PinIO::PIN_FN_B)       // Peripheral B: CAN 1 Transmit
00016 #define PINJ1_3_SPI0_NPCS2 (PinIO::PIN_FN_C)   // Peripheral C: SPI 0 Chip Select 2
00017 #define PINJ1_3_ISI_D6 (PinIO::PIN_FN_D)       // Peripheral D: Image Sensor Data Input 6
00018
00019 // Connector: J1 / Pin: 4 / CPU Port: PD21
00020 #define PINJ1_4_IN (PinIO::PIN_FN_IN)           // GPIO Input
00021 #define PINJ1_4_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00022 #define PINJ1_4_PWMC0_PWMH1 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 1 Output High
00023 #define PINJ1_4_SPI0_MOSI (PinIO::PIN_FN_B)     // Peripheral B: SPI 0 Master Out Slave In
00024 #define PINJ1_4_TIOA11 (PinIO::PIN_FN_C)        // Peripheral C: Timer 11 Line A
00025 #define PINJ1_4_ISI_D1 (PinIO::PIN_FN_D)        // Peripheral D: Image Sensor Data Input 1
00026
00027 // Connector: J1 / Pin: 5 / CPU Port: PB3
00028 #define PINJ1_5_IN (PinIO::PIN_FN_IN)      // GPIO Input
00029 #define PINJ1_5_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00030 #define PINJ1_5_CANRX0 (PinIO::PIN_FN_A)   // Peripheral A: CAN 0 Receive
00031 #define PINJ1_5_PCK2 (PinIO::PIN_FN_B)     // Peripheral B: Programmable Clock Output 2
00032 #define PINJ1_5_RTS0 (PinIO::PIN_FN_C)     // Peripheral C: USART 0 RTS
00033 #define PINJ1_5_ISI_D2 (PinIO::PIN_FN_D)   // Peripheral D: Image Sensor Data Input 2
00034
00035 // Connector: J1 / Pin: 6 / CPU Port: PD20
00036 #define PINJ1_6_IN (PinIO::PIN_FN_IN)           // GPIO Input
00037 #define PINJ1_6_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00038 #define PINJ1_6_PWMC0_PWMH0 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 0 Output High
00039 #define PINJ1_6_SPI0_MISO (PinIO::PIN_FN_B)     // Peripheral B: SPI 0 Master In Slave Out
00040 #define PINJ1_6_GTSUCOMP (PinIO::PIN_FN_C)      // Peripheral C: TSU Timer Comparison Valid 1588
00041
00042 // Connector: J1 / Pin: 7 / CPU Port: PD22
00043 #define PINJ1_7_IN (PinIO::PIN_FN_IN)           // GPIO Input
00044 #define PINJ1_7_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00045 #define PINJ1_7_PWMC0_PWMH2 (PinIO::PIN_FN_A)   // Peripheral A: PWM 0 Channel 2 Output High
00046 #define PINJ1_7_SPI0_SPCK (PinIO::PIN_FN_B)     // Peripheral B: SPI 0 Clock
00047 #define PINJ1_7_TIOB11 (PinIO::PIN_FN_C)        // Peripheral C: Timer 11 Line B
00048 #define PINJ1_7_ISI_D0 (PinIO::PIN_FN_D)        // Peripheral D: Image Sensor Data Input 0
00049
00050 // Connector: J1 / Pin: 8 / CPU Port: PA25
00051 #define PINJ1_8_IN (PinIO::PIN_FN_IN)           // GPIO Input
00052 #define PINJ1_8_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00053 #define PINJ1_8_CTS1 (PinIO::PIN_FN_A)          // Peripheral A: USART 1 CTS
00054 #define PINJ1_8_PWMC0_PWMH2 (PinIO::PIN_FN_B)   // Peripheral B: PWM Clock 0 Channel 2 Output High
00055 #define PINJ1_8_A23 (PinIO::PIN_FN_C)           // Peripheral C: External Bus Interface A23
00056 #define PINJ1_8_MCCK (PinIO::PIN_FN_D)          // Peripheral D: Multimedia Card Clock
00057
00058 // Connector: J1 / Pin: 9 / CPU Port: PA24
00059 #define PINJ1_9_IN (PinIO::PIN_FN_IN)           // GPIO Input
```

```
00060 #define PINJ1_9_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00061 #define PINJ1_9_RTS1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 RTS
00062 #define PINJ1_9_PWMC0_PWMH1 (PinIO::PIN_FN_B)    // Peripheral B: PWM Clock 0 Channel 1 Output High
00063 #define PINJ1_9_A20 (PinIO::PIN_FN_C)            // Peripheral C: External Bus Interface A20
00064 #define PINJ1_9_ISI_PCK (PinIO::PIN_FN_D)        // Peripheral D: Image Sensor Data Clock
00065
00066 // Connector: J1 / Pin: 10 / CPU Port: PB1
00067 #define PINJ1_10_IN (PinIO::PIN_FN_IN)            // GPIO Input
00068 #define PINJ1_10_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00069 #define PINJ1_10_PWMC0_PWMH1 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 1 Output High
00070 #define PINJ1_10_GTSUCOMP (PinIO::PIN_FN_B)       // Peripheral B: TSU Timer Comparison Valid 1588
00071 #define PINJ1_10_TXD0 (PinIO::PIN_FN_C)           // Peripheral C: USART 0 Transmit
00072 #define PINJ1_10_TK (PinIO::PIN_FN_D)             // Peripheral D: SSC Transmit Clock
00073
00074 // Connector: J1 / Pin: 11 / CPU Port: PB0
00075 #define PINJ1_11_IN (PinIO::PIN_FN_IN)            // GPIO Input
00076 #define PINJ1_11_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00077 #define PINJ1_11_PWMC0_PWMH0 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 0 Output High
00078 #define PINJ1_11_RXD0 (PinIO::PIN_FN_C)           // Peripheral C: USART 0 Receive
00079 #define PINJ1_11_TF (PinIO::PIN_FN_D)             // Peripheral D: SSC Transmit Frame Sync
00080
00081 // Connector: J1 / Pin: 12 / CPU Port: PB4
00082 #define PINJ1_12_IN (PinIO::PIN_FN_IN)            // GPIO Input
00083 #define PINJ1_12_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00084 #define PINJ1_12_TWD1 (PinIO::PIN_FN_A)           // Peripheral A: Two-wire Channel 1 Data
00085 #define PINJ1_12_PWMC0_PWMH2 (PinIO::PIN_FN_B)    // Peripheral B: PWM 0 Channel 2 Output High
00086 #define PINJ1_12_TXD1 (PinIO::PIN_FN_D)           // Peripheral D: USART 1 Transmit
00087
00088 /*
00089 // Connector: J1 / Pin: 12 / CPU Port: PD31
00090 #define PINJ1_12_IN             ( PinIO::PIN_FN_IN )     // GPIO Input
00091 #define PINJ1_12_OUT            ( PinIO::PIN_FN_OUT )    // GPIO Output
00092 #define PINJ1_12_QIO3           ( PinIO::PIN_FN_A ) // Peripheral A: QSPI Quad Mode Data 3
00093 #define PINJ1_12_UTXD3          ( PinIO::PIN_FN_B ) // Peripheral B: UART 3 Transmit
00094 #define PINJ1_12_PCK2           ( PinIO::PIN_FN_C ) // Peripheral C: Programmable Clock 2 Output
00095 #define PINJ1_12_ISI_D11        ( PinIO::PIN_FN_D ) // Peripheral D: Image Sensor Data Input 11
00096 */
00097
00098 // Connector: J1 / Pin: 13 / CPU Port: PA21
00099 #define PINJ1_13_IN (PinIO::PIN_FN_IN)             // GPIO Input
00100 #define PINJ1_13_OUT (PinIO::PIN_FN_OUT)           // GPIO Output
00101 #define PINJ1_13_RXD1 (PinIO::PIN_FN_A)            // Peripheral A: USART 1 RX
00102 #define PINJ1_13_PCK1 (PinIO::PIN_FN_B)            // Peripheral B: Programmable Clock Output 1
00103 #define PINJ1_13_PWMC1_PWMFI0 (PinIO::PIN_FN_C)    // Peripheral C: PWM Clock 1 Channel 0 Fault Input
00104
00105 // Connector: J1 / Pin: 14 / CPU Port: PB2
00106 #define PINJ1_14_IN (PinIO::PIN_FN_IN)           // GPIO Input
00107 #define PINJ1_14_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00108 #define PINJ1_14_CANTX0 (PinIO::PIN_FN_A)        // Peripheral A: CAN 0 Transmit
00109 #define PINJ1_14_CTS0 (PinIO::PIN_FN_C)          // Peripheral C: USART 0 CTS
00110 #define PINJ1_14_SPI0_NPCS0 (PinIO::PIN_FN_D)    // Peripheral D: SPI 0 Chip Select 0
00111
00112 // Connector: J1 / Pin: 15 / CPU Port: PD24
00113 #define PINJ1_15_IN (PinIO::PIN_FN_IN)            // GPIO Input
00114 #define PINJ1_15_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00115 #define PINJ1_15_PWMC0_PWML0 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 0 Output Low
00116 #define PINJ1_15_RF (PinIO::PIN_FN_B)             // Peripheral B: SSC Receive Frame Sync
00117 #define PINJ1_15_TCLK11 (PinIO::PIN_FN_C)         // Peripheral C: Timer 11 Clock Input
00118 #define PINJ1_15_ISI_HSYNC (PinIO::PIN_FN_D)      // Peripheral D: Image Sensor Horizontal Sync
00119
00120 // Connector: J1 / Pin: 16 / CPU Port: PA3
00121 #define PINJ1_16_IN (PinIO::PIN_FN_IN)        // GPIO Input
00122 #define PINJ1_16_OUT (PinIO::PIN_FN_OUT)      // GPIO Output
00123 #define PINJ1_16_TWD0 (PinIO::PIN_FN_A)       // Peripheral A: Two-wire 0 Data
00124 #define PINJ1_16_LONCOL1 (PinIO::PIN_FN_B)    // Peripheral B: LON Channel 1 Collision Detect
00125 #define PINJ1_16_PCK2 (PinIO::PIN_FN_C)       // Peripheral C: Programmable Clock Channel 2 Output
00126
00127 // Connector: J1 / Pin: 17 / CPU Port: PA4
00128 #define PINJ1_17_IN (PinIO::PIN_FN_IN)      // GPIO Input
00129 #define PINJ1_17_OUT (PinIO::PIN_FN_OUT)    // GPIO Output
00130 #define PINJ1_17_TWCK0 (PinIO::PIN_FN_A)    // Peripheral A: Two Wire (SCL) 0 Clock
00131 #define PINJ1_17_TCLK0 (PinIO::PIN_FN_B)    // Peripheral B: Timer 0 Clock
00132 #define PINJ1_17_UTXD1 (PinIO::PIN_FN_C)    // Peripheral C: UART 1 Transmit
00133
00134 // Connector: J3 / Pin: 2 / CPU Port: PB13
00135 #define PINJ3_2_IN (PinIO::PIN_FN_IN)            // GPIO Input
00136 #define PINJ3_2_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00137 #define PINJ3_2_PWMC0_PWML2 (PinIO::PIN_FN_A)    // Peripheral A: PWM 0 Channel 2 Output Low
00138 #define PINJ3_2_PCK0 (PinIO::PIN_FN_B)           // Peripheral B: Programmable Clock output 0
00139 #define PINJ3_2_SCK0 (PinIO::PIN_FN_C)           // Peripheral C: USART 0 Serial Clock
00140
00141 // Connector: J3 / Pin: 3 / CPU Port: PC12
00142 #define PINJ3_3_IN (PinIO::PIN_FN_IN)       // GPIO Input
00143 #define PINJ3_3_OUT (PinIO::PIN_FN_OUT)     // GPIO Output
00144 #define PINJ3_3_TIOB8 (PinIO::PIN_FN_B)     // Peripheral B: Timer 8 Line B
00145 #define PINJ3_3_CANRX1 (PinIO::PIN_FN_C)    // Peripheral C: CAN 1 Receive
00146
```

```
00147 // Connector: J3 / Pin: 4 / CPU Port: PB4
00148 #define PINJ3_4_IN (PinIO::PIN_FN_IN)           // GPIO Input
00149 #define PINJ3_4_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00150 #define PINJ3_4_TWD1 (PinIO::PIN_FN_A)          // Peripheral A: Two-wire Channel 1 Data
00151 #define PINJ3_4_PWMC0_PWMH2 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 Channel 2 Output High
00152 #define PINJ3_4_TXD1 (PinIO::PIN_FN_D)          // Peripheral D: USART 1 Transmit
00153
00154 // Connector: J3 / Pin: 5 / CPU Port: PD30
00155 #define PINJ3_5_IN (PinIO::PIN_FN_IN)       // GPIO Input
00156 #define PINJ3_5_OUT (PinIO::PIN_FN_OUT)     // GPIO Output
00157 #define PINJ3_5_UTXD3 (PinIO::PIN_FN_A)     // Peripheral A: UART 3 Transmit
00158 #define PINJ3_5_ISI_D10 (PinIO::PIN_FN_D)   // Peripheral D: Image Sensor Data Input 10
00159
00160 // Connector: J3 / Pin: 6 / CPU Port: PB5
00161 #define PINJ3_6_IN (PinIO::PIN_FN_IN)           // GPIO Input
00162 #define PINJ3_6_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00163 #define PINJ3_6_TWCK1 (PinIO::PIN_FN_A)         // Peripheral A: Two-wire Channel 1 Clock
00164 #define PINJ3_6_PWMC0_PWML0 (PinIO::PIN_FN_B)   // Peripheral B: PWM 0 Channel 0 Output Low
00165 #define PINJ3_6_TD (PinIO::PIN_FN_D)            // Peripheral D: SSC Transmit Data
00166
00167 // Connector: J3 / Pin: 7 / CPU Port: PA17
00168 #define PINJ3_7_IN (PinIO::PIN_FN_IN)           // GPIO Input
00169 #define PINJ3_7_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00170 #define PINJ3_7_QIO2 (PinIO::PIN_FN_A)          // Peripheral A: QSPI Data 2 Quad Mode
00171 #define PINJ3_7_PCK1 (PinIO::PIN_FN_B)          // Peripheral B: Programmable Clock Output 1
00172 #define PINJ3_7_PWMC0_PWMH3 (PinIO::PIN_FN_C)   // Peripheral C: PWM clock 0 Channel 3 Output High
00173
00174 // Connector: J4 / Pin: 1 / CPU Port: PC30
00175 #define PINJ4_1_IN (PinIO::PIN_FN_IN)     // GPIO Input
00176 #define PINJ4_1_OUT (PinIO::PIN_FN_OUT)   // GPIO Output
00177 #define PINJ4_1_A12 (PinIO::PIN_FN_A)     // Peripheral A: External Bus Interface A12
00178 #define PINJ4_1_TIOB5 (PinIO::PIN_FN_B)   // Peripheral B: Timer 5 Line B
00179
00180 // Connector: J4 / Pin: 2 / CPU Port: PD25
00181 #define PINJ4_2_IN (PinIO::PIN_FN_IN)           // GPIO Input
00182 #define PINJ4_2_OUT (PinIO::PIN_FN_OUT)         // GPIO Output
00183 #define PINJ4_2_PWMC0_PWML1 (PinIO::PIN_FN_A)   // Peripheral A: PWM Clock 0 Channel 1 Output Low
00184 #define PINJ4_2_SPI0_NPCS1 (PinIO::PIN_FN_B)    // Peripheral B: SPI 0 Chip Select 1
00185 #define PINJ4_2_URXD2 (PinIO::PIN_FN_C)         // Peripheral C: UART 2 Receive
00186 #define PINJ4_2_ISI_VSYNC (PinIO::PIN_FN_D)     // Peripheral D: Image Sensor Vertical Sync
00187
00188 // Connector: J4 / Pin: 6 / CPU Port: PA26
00189 #define PINJ4_6_IN (PinIO::PIN_FN_IN)            // GPIO Input
00190 #define PINJ4_6_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00191 #define PINJ4_6_DCD1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 DCD
00192 #define PINJ4_6_TIOA2 (PinIO::PIN_FN_B)          // Peripheral B: Timer 2 Line A
00193 #define PINJ4_6_MCDA2 (PinIO::PIN_FN_C)          // Peripheral C: Multimedia Card Slot A Data 2
00194 #define PINJ4_6_PWMC1_PWMFI1 (PinIO::PIN_FN_D)   // Peripheral D: PWM 1 Fault Input 1
00195
00196 // Connector: J4 / Pin: 7 / CPU Port: PA28
00197 #define PINJ4_7_IN (PinIO::PIN_FN_IN)            // GPIO Input
00198 #define PINJ4_7_OUT (PinIO::PIN_FN_OUT)          // GPIO Output
00199 #define PINJ4_7_DSR1 (PinIO::PIN_FN_A)           // Peripheral A: USART 1 DSR
00200 #define PINJ4_7_TCLK1 (PinIO::PIN_FN_B)          // Peripheral B: Timer 1 Clock
00201 #define PINJ4_7_MCCDA (PinIO::PIN_FN_C)          // Peripheral C: MUltimedia Card Slot A Data Command
00202 #define PINJ4_7_PWMC1_PWMFI2 (PinIO::PIN_FN_D)   // Peripheral D: PWM 1 Fault Input 2
00203
00204 // Old constants for compatibillity
00205
00206 // Pin:  3
00207 #define PIN3_GPIO (PinIO::PIN_FN_IN)
00208
00209 // Pin:  4
00210 #define PIN4_SPI_DOUT (PinIO::PIN_FN_B)   // Primary Function: SPI Data Out
00211 #define PIN4_GPIO (PinIO::PIN_FN_IN)
00212
00213 // Pin:  5
00214 #define PIN5_GPIO (PinIO::PIN_FN_IN)
00215 #define PIN5_URT0_RTS (PinIO::PIN_FN_C)   // Primary Function: UART 0 - Request to Send
00216
00217 // Pin:  6
00218 #define PIN6_SPI_DIN (PinIO::PIN_FN_B)   // Primary Function: SPI Data In
00219 #define PIN6_GPIO (PinIO::PIN_FN_IN)
00220
00221 // Pin:  7
00222 #define PIN7_SPI_CLK (PinIO::PIN_FN_B)   // Primary Function: SPI Clock
00223 #define PIN7_GPIO (PinIO::PIN_FN_IN)
00224
00225 // Pin:  8
00226 #define PIN8_UCTS1 (PinIO::PIN_FN_A)   // Primary Function: UART 1 - Clear to Send
00227 #define PIN8_GPIO (PinIO::PIN_FN_IN)
00228
00229 // Pin:  9
00230 #define PIN9_URT1_RTS (PinIO::PIN_FN_A)   // Primary Function: UART 1 - Request to Send
00231 #define PIN9_GPIO (PinIO::PIN_FN_IN)
00232
00233 // Pin: 10
```

```
00234 #define PIN10_UTXD0 (PinIO::PIN_FN_C)    // Primary Function: UART 0 - Transmit
00235 #define PIN10_GPIO (PinIO::PIN_FN_IN)
00236
00237 // Pin: 11
00238 #define PIN11_URXD0 (PinIO::PIN_FN_C)    // Primary Function: UART 0 - Receive
00239 #define PIN11_GPIO (PinIO::PIN_FN_IN)
00240
00241 // Pin: 12
00242 #define PIN12_UTXD1 (PinIO::PIN_FN_D)    // Primary Function: UART 1 - Transmit
00243 #define PIN12_GPIO (PinIO::PIN_FN_IN)
00244
00245 // Pin: 13
00246 #define PIN13_URXD1         ( PinIO::PIN_FN_A ) )    // Primary Function: UART 1 - Receive
00247 #define PIN13_GPIO (PinIO::PIN_FN_IN)
00248
00249 // Pin: 14
00250 #define PIN14_URT0_CTS (PinIO::PIN_FN_C)    // Primary Function: UART 0 - Clear to Send
00251 #define PIN14_GPIO (PinIO::PIN_FN_IN)
00252
00253 // Pin: 15
00254 #define PIN15_GPIO (PinIO::PIN_FN_IN)
00255
00256 // Pin: 16
00257 #define PIN16_SDA (PinIO::PIN_FN_A)
00258 #define PIN16_GPIO (PinIO::PIN_FN_IN)
00259
00260 // Pin: 17
00261 #define PIN17_SCL (PinIO::PIN_FN_A)    // Primary Function: I2C Serial Clock
00262 #define PIN17_GPIO (PinIO::PIN_FN_IN)
00263
00264 #endif    // SBE70LC
```

24.534 SOMRT1061/include/pinconstant.h

```
00001 /*****************************************************************************/
00031 // GPIO_EMC_18
00032 #define PIN_02_SEMC_ADDR09                  PinIO::PIN_FN_0
00033 #define PIN_02_FLEXPWM4_PWMB3               PinIO::PIN_FN_1
00034 #define PIN_02_LPUART4_RTS_B                PinIO::PIN_FN_2
00035 #define PIN_02_SER_PORT_2_RTS               PinIO::PIN_FN_2
00036 #define PIN_02_FLEXCAN1_RX                  PinIO::PIN_FN_3
00037 #define PIN_02_QTIMER3_TIMER3               PinIO::PIN_FN_4
00038 #define PIN_02_GPIO4_IO18                   PinIO::PIN_FN_5
00039 #define PIN_02_SNVS_VIO_5_CTL               PinIO::PIN_FN_6
00040
00041 // GPIO_EMC_17
00042 #define PIN_03_SEMC_ADDR08                  PinIO::PIN_FN_0
00043 #define PIN_03_FLEXPWM4_PWMA3               PinIO::PIN_FN_1
00044 #define PIN_03_LPUART4_CTS_B                PinIO::PIN_FN_2
00045 #define PIN_03_SER_PORT_2_CTS               PinIO::PIN_FN_2
00046 #define PIN_03_FLEXCAN1_TX                  PinIO::PIN_FN_3
00047 #define PIN_03_QTIMER3_TIMER2               PinIO::PIN_FN_4
00048 #define PIN_03_GPIO4_IO17                   PinIO::PIN_FN_5
00049
00050 // GPIO_EMC_16
00051 #define PIN_04_SEMC_ADDR07                  PinIO::PIN_FN_0
00052 #define PIN_04_XBAR1_IN21                   PinIO::PIN_FN_1
00053 #define PIN_04_LPUART3_RTS_B                PinIO::PIN_FN_2
00054 #define PIN_04_SER_PORT_0_RTS               PinIO::PIN_FN_2
00055 #define PIN_04_SPDIF_IN                     PinIO::PIN_FN_3
00056 #define PIN_04_QTIMER3_TIMER1               PinIO::PIN_FN_4
00057 #define PIN_04_GPIO4_IO16                   PinIO::PIN_FN_5
00058 #define PIN_04_FLEXSPI2_B_DATA03            PinIO::PIN_FN_8
00059
00060 // GPIO_EMC_15
00061 #define PIN_06_SEMC_ADDR06                  PinIO::PIN_FN_0
00062 #define PIN_06_XBAR1_IN20                   PinIO::PIN_FN_1
00063 #define PIN_06_LPUART3_CTS_B                PinIO::PIN_FN_2
00064 #define PIN_06_SER_PORT_0_CTS               PinIO::PIN_FN_2
00065 #define PIN_06_SPDIF_OUT                    PinIO::PIN_FN_3
00066 #define PIN_06_QTIMER3_TIMER0               PinIO::PIN_FN_4
00067 #define PIN_06_GPIO4_IO15                   PinIO::PIN_FN_5
00068 #define PIN_06_USB_PHY2_TSTO_RX_DISCON_DET  PinIO::PIN_FN_5
00069 #define PIN_06_FLEXSPI2_B_DATA02            PinIO::PIN_FN_8
00070
00071 // GPIO_EMC_14
00072 #define PIN_07_SEMC_ADDR05                  PinIO::PIN_FN_0
00073 #define PIN_07_XBAR1_INOUT19                PinIO::PIN_FN_1
00074 #define PIN_07_LPUART3_RX                   PinIO::PIN_FN_2
00075 #define PIN_07_SER_PORT_0_RX                PinIO::PIN_FN_2
00076 #define PIN_07_MQS_LEFT                     PinIO::PIN_FN_3
00077 #define PIN_07_LPSPI2_PCS1                  PinIO::PIN_FN_4
00078 #define PIN_07_GPIO4_IO14                   PinIO::PIN_FN_5
00079 #define PIN_07_USB_PHY2_TSTO_RX_SQUELCH     PinIO::PIN_FN_6
00080 #define PIN_07_FLEXSPI2_B_DATA01            PinIO::PIN_FN_8
```

```
00081
00082 // GPIO_EMC_13
00083 #define PIN_08_SEMC_ADDR04                  PinIO::PIN_FN_0
00084 #define PIN_08_XBAR1_IN25                   PinIO::PIN_FN_1
00085 #define PIN_08_LPUART3_TX                   PinIO::PIN_FN_2
00086 #define PIN_08_SER_PORT_0_TX                PinIO::PIN_FN_2
00087 #define PIN_08_MQS_RIGHT                    PinIO::PIN_FN_3
00088 #define PIN_08_FLEXPWM1_PWMB3               PinIO::PIN_FN_4
00089 #define PIN_08_GPIO4_IO13                   PinIO::PIN_FN_5
00090 #define PIN_08_USB_PHY2_TSTO_PLL_CLK20DIV   PinIO::PIN_FN_6
00091 #define PIN_08_FLEXSPI2_B_DATA00            PinIO::PIN_FN_8
00092
00093 // GPIO_EMC_12
00094 #define PIN_09_SEMC_ADDR03                  PinIO::PIN_FN_0
00095 #define PIN_09_XBAR1_IN24                   PinIO::PIN_FN_1
00096 #define PIN_09_LPI2C4_SCL                   PinIO::PIN_FN_2
00097 #define PIN_09_USDHC1_WP                    PinIO::PIN_FN_3
00098 #define PIN_09_FLEXPWM1_PWMA3               PinIO::PIN_FN_4
00099 #define PIN_09_GPIO4_IO12                   PinIO::PIN_FN_5
00100 #define PIN_09_USB_PHY1_TSTO_PLL_CLK20DIV   PinIO::PIN_FN_6
00101 #define PIN_09_FLEXSPI2_B_SCLK              PinIO::PIN_FN_8
00102
00103 // GPIO_EMC_11
00104 #define PIN_10_SEMC_ADDR02                  PinIO::PIN_FN_0
00105 #define PIN_10_FLEXPWM2_PWMB2               PinIO::PIN_FN_1
00106 #define PIN_10_LPI2C4_SDA                   PinIO::PIN_FN_2
00107 //#define PIN_10_USDHC2_RESET_B               PinIO::PIN_FN_3
00108 #define PIN_10_FLEXIO1_FLEXIO11             PinIO::PIN_FN_4
00109 #define PIN_10_GPIO4_IO11                   PinIO::PIN_FN_5
00110 #define PIN_10_USB_PHY2_TSTO_RX_HS_RXD      PinIO::PIN_FN_6
00111 #define PIN_10_FLEXSPI2_B_DQS               PinIO::PIN_FN_8
00112
00113 // GPIO_EMC_10
00114 #define PIN_11_SEMC_ADDR01                  PinIO::PIN_FN_0
00115 #define PIN_11_FLEXPWM2_PWMA2               PinIO::PIN_FN_1
00116 #define PIN_11_SAI2_RX_BCLK                 PinIO::PIN_FN_2
00117 #define PIN_11_FLEXCAN2_RX                  PinIO::PIN_FN_3
00118 #define PIN_11_FLEXIO1_FLEXIO10             PinIO::PIN_FN_4
00119 #define PIN_11_GPIO4_IO10                   PinIO::PIN_FN_5
00120 #define PIN_11_USB_PHY1_TSTI_TX_HIZ         PinIO::PIN_FN_6
00121 #define PIN_11_FLEXSPI2_B_SS0_B             PinIO::PIN_FN_8
00122
00123 // GPIO_EMC_09
00124 #define PIN_12_SEMC_ADDR00                  PinIO::PIN_FN_0
00125 #define PIN_12_FLEXPWM2_PWMB1               PinIO::PIN_FN_1
00126 #define PIN_12_SAI2_RX_SYNC                 PinIO::PIN_FN_2
00127 #define PIN_12_FLEXCAN2_TX                  PinIO::PIN_FN_3
00128 #define PIN_12_FLEXIO1_FLEXIO09             PinIO::PIN_FN_4
00129 #define PIN_12_GPIO4_IO09                   PinIO::PIN_FN_5
00130 #define PIN_12_USB_PHY1_TSTI_TX_EN          PinIO::PIN_FN_6
00131 #define PIN_12_FLEXSPI2_B_SS1_B             PinIO::PIN_FN_8
00132
00133 // GPIO_EMC_08
00134 #define PIN_13_SEMC_DM00                    PinIO::PIN_FN_0
00135 #define PIN_13_FLEXPWM2_PWMA1               PinIO::PIN_FN_1
00136 #define PIN_13_SAI2_RX_DATA                 PinIO::PIN_FN_2
00137 #define PIN_13_XBAR1_INOUT17                PinIO::PIN_FN_3
00138 #define PIN_13_FLEXIO1_FLEXIO08             PinIO::PIN_FN_4
00139 #define PIN_13_GPIO4_IO08                   PinIO::PIN_FN_5
00140 #define PIN_13_USB_PHY1_TSTI_TX_DP          PinIO::PIN_FN_6
00141
00142 // GPIO_EMC_00
00143 #define PIN_14_SEMC_DATA00                  PinIO::PIN_FN_0
00144 #define PIN_14_FLEXPWM4_PWMA0               PinIO::PIN_FN_1
00145 #define PIN_14_LPSPI2_SCK                   PinIO::PIN_FN_2
00146 #define PIN_14_XBAR1_XBAR_IN02              PinIO::PIN_FN_3
00147 #define PIN_14_FLEXIO1_FLEXIO00             PinIO::PIN_FN_4
00148 #define PIN_14_GPIO4_IO00                   PinIO::PIN_FN_5
00149 #define PIN_14_USB_PHY1_TSTI_TX_LS_MODE     PinIO::PIN_FN_6
00150
00151 // GPIO_EMC_01
00152 #define PIN_15_SEMC_DATA01                  PinIO::PIN_FN_0
00153 #define PIN_15_FLEXPWM4_PWMB0               PinIO::PIN_FN_1
00154 #define PIN_15_LPSPI2_PCS0                  PinIO::PIN_FN_2
00155 #define PIN_15_XBAR1_IN03                   PinIO::PIN_FN_3
00156 #define PIN_15_FLEXIO1_FLEXIO01             PinIO::PIN_FN_4
00157 #define PIN_15_GPIO4_IO01                   PinIO::PIN_FN_5
00158 #define PIN_15_USB_PHY1_TSTI_TX_HS_MODE     PinIO::PIN_FN_6
00159
00160 // GPIO_EMC_02
00161 #define PIN_16_SEMC_DATA02                  PinIO::PIN_FN_0
00162 #define PIN_16_FLEXPWM4_PWMA1               PinIO::PIN_FN_1
00163 #define PIN_16_LPSPI2_SDO                   PinIO::PIN_FN_2
00164 #define PIN_16_XBAR1_INOUT04                PinIO::PIN_FN_3
00165 #define PIN_16_FLEXIO1_FLEXIO02             PinIO::PIN_FN_4
00166 #define PIN_16_GPIO4_IO02                   PinIO::PIN_FN_5
00167 #define PIN_16_USB_PHY1_TSTI_TX_DN          PinIO::PIN_FN_6
```

```
00168
00169 // GPIO_EMC_03
00170 #define PIN_18_SEMC_DATA03                  PinIO::PIN_FN_0
00171 #define PIN_18_FLEXPWM4_PWMB1               PinIO::PIN_FN_1
00172 #define PIN_18_LPSPI2_SDI                   PinIO::PIN_FN_2
00173 #define PIN_18_XBAR1_INOUT05                PinIO::PIN_FN_3
00174 #define PIN_18_FLEXIO1_FLEXIO03             PinIO::PIN_FN_4
00175 #define PIN_18_GPIO4_IO03                   PinIO::PIN_FN_5
00176 #define PIN_18_USB_PHY1_TSTO_RX_SQUELCH     PinIO::PIN_FN_6
00177
00178 // GPIO_EMC_04
00179 #define PIN_19_SEMC_DATA04                  PinIO::PIN_FN_0
00180 #define PIN_19_FLEXPWM4_PWMA2               PinIO::PIN_FN_1
00181 #define PIN_19_SAI2_TX_DATA                 PinIO::PIN_FN_2
00182 #define PIN_19_XBAR1_INOUT06                PinIO::PIN_FN_3
00183 #define PIN_19_FLEXIO1_FLEXIO04             PinIO::PIN_FN_4
00184 #define PIN_19_GPIO4_IO04                   PinIO::PIN_FN_5
00185 #define PIN_19_USB_PHY1_TSTO_RX_DISCON_DET  PinIO::PIN_FN_6
00186
00187 // GPIO_EMC_05
00188 #define PIN_20_SEMC_DATA05                  PinIO::PIN_FN_0
00189 #define PIN_20_FLEXPWM4_PWMB2               PinIO::PIN_FN_1
00190 #define PIN_20_SAI2_TX_SYNC                 PinIO::PIN_FN_2
00191 #define PIN_20_XBAR1_INOUT07                PinIO::PIN_FN_3
00192 #define PIN_20_FLEXIO1_FLEXIO05             PinIO::PIN_FN_4
00193 #define PIN_20_GPIO4_IO05                   PinIO::PIN_FN_5
00194 #define PIN_20_USB_PHY1_TSTO_RX_HS_RXD      PinIO::PIN_FN_6
00195
00196 // GPIO_EMC_06
00197 #define PIN_21_SEMC_DATA06                  PinIO::PIN_FN_0
00198 #define PIN_21_FLEXPWM2_PWMA0               PinIO::PIN_FN_1
00199 #define PIN_21_SAI2_TX_BCLK                 PinIO::PIN_FN_2
00200 #define PIN_21_XBAR1_INOUT08                PinIO::PIN_FN_3
00201 #define PIN_21_FLEXIO1_FLEXIO06             PinIO::PIN_FN_4
00202 #define PIN_21_GPIO4_IO06                   PinIO::PIN_FN_5
00203 #define PIN_21_USB_PHY2_TSTO_RX_FS_RXD      PinIO::PIN_FN_6
00204
00205 // GPIO_EMC_07
00206 #define PIN_22_SEMC_DATA07                  PinIO::PIN_FN_0
00207 #define PIN_22_FLEXPWM2_PWMB0               PinIO::PIN_FN_1
00208 #define PIN_22_SAI2_MCLK                    PinIO::PIN_FN_2
00209 #define PIN_22_XBAR1_INOUT09                PinIO::PIN_FN_3
00210 #define PIN_22_FLEXIO1_FLEXIO07             PinIO::PIN_FN_4
00211 #define PIN_22_GPIO4_IO07                   PinIO::PIN_FN_5
00212 #define PIN_22_USB_PHY1_TSTO_RX_FS_RXD      PinIO::PIN_FN_6
00213
00214 // GPIO_SD_B0_03
00215 #define PIN_24_USDHC1_DATA1                 PinIO::PIN_FN_0
00216 #define PIN_24_FLEXPWM1_PWMB1               PinIO::PIN_FN_1
00217 #define PIN_24_LPUART8_RTS_B                PinIO::PIN_FN_2
00218 #define PIN_24_SER_PORT_6_RTS               PinIO::PIN_FN_2
00219 #define PIN_24_XBAR1_INOUT07                PinIO::PIN_FN_3
00220 #define PIN_24_LPSPI1_SDI                   PinIO::PIN_FN_4
00221 #define PIN_24_GPIO3_IO15                   PinIO::PIN_FN_5
00222 #define PIN_24_ENET2_RDATA00                PinIO::PIN_FN_8
00223 #define PIN_24_SEMC_CLK6                    PinIO::PIN_FN_9
00224
00225 // GPIO_SD_B0_02
00226 #define PIN_25_USDHC1_DATA0                 PinIO::PIN_FN_0
00227 #define PIN_25_FLEXPWM1_PWMA1               PinIO::PIN_FN_1
00228 #define PIN_25_LPUART8_CTS_B                PinIO::PIN_FN_2
00229 #define PIN_25_SER_PORT_6_CTS               PinIO::PIN_FN_2
00230 #define PIN_25_XBAR1_INOUT06                PinIO::PIN_FN_3
00231 #define PIN_25_LPSPI1_SDO                   PinIO::PIN_FN_4
00232 #define PIN_25_GPIO3_IO14                   PinIO::PIN_FN_5
00233 #define PIN_25_ENET2_RX_ER                  PinIO::PIN_FN_8
00234 #define PIN_25_SEMC_CLK5                    PinIO::PIN_FN_9
00235
00236 // GPIO_SD_B0_01
00237 #define PIN_26_USDHC1_CLK                   PinIO::PIN_FN_0
00238 #define PIN_26_FLEXPWM1_PWMB0               PinIO::PIN_FN_1
00239 #define PIN_26_LPI2C3_SDA                   PinIO::PIN_FN_2
00240 #define PIN_26_XBAR1_INOUT05                PinIO::PIN_FN_3
00241 #define PIN_26_LPSPI1_PCS0                  PinIO::PIN_FN_4
00242 #define PIN_26_GPIO3_IO13                   PinIO::PIN_FN_5
00243 //#define PIN_26_FLEXSPIB_SS1_B               PinIO::PIN_FN_6
00244 #define PIN_26_ENET2_TX_CLK                 PinIO::PIN_FN_8
00245 #define PIN_26_ENET2_REF_CLK2               PinIO::PIN_FN_9
00246
00247 // GPIO_SD_B0_00
00248 #define PIN_27_USDHC1_CMD                   PinIO::PIN_FN_0
00249 #define PIN_27_FLEXPWM1_PWMA0               PinIO::PIN_FN_1
00250 #define PIN_27_LPI2C3_SCL                   PinIO::PIN_FN_2
00251 #define PIN_27_XBAR1_INOUT04                PinIO::PIN_FN_3
00252 #define PIN_27_LPSPI1_SCK                   PinIO::PIN_FN_4
00253 #define PIN_27_GPIO3_IO12                   PinIO::PIN_FN_5
00254 //#define PIN_27_FLEXSPIA_SS1_B               PinIO::PIN_FN_6
```

```
00255 #define PIN_27_ENET2_TX_EN                  PinIO::PIN_FN_8
00256 #define PIN_27_SEMC_DQS4                    PinIO::PIN_FN_9
00257
00258 // GPIO_SD_B0_05
00259 #define PIN_28_USDHC1_DATA3                 PinIO::PIN_FN_0
00260 #define PIN_28_FLEXPWM1_PWMB2               PinIO::PIN_FN_1
00261 #define PIN_28_LPUART8_RX                   PinIO::PIN_FN_2
00262 #define PIN_28_SER_PORT_6_RX                PinIO::PIN_FN_2
00263 #define PIN_28_XBAR1_INOUT09                PinIO::PIN_FN_3
00264 //#define PIN_28_FLEXSPIB_DQS                 PinIO::PIN_FN_4
00265 #define PIN_28_GPIO3_IO17                   PinIO::PIN_FN_5
00266 #define PIN_28_CCM_CLKO2                    PinIO::PIN_FN_6
00267 #define PIN_28_ENET2_RX_EN                  PinIO::PIN_FN_8
00268
00269 // GPIO_SD_B0_04
00270 #define PIN_29_USDHC1_DATA2                 PinIO::PIN_FN_0
00271 #define PIN_29_FLEXPWM1_PWMA2               PinIO::PIN_FN_1
00272 #define PIN_29_LPUART8_TX                   PinIO::PIN_FN_2
00273 #define PIN_29_SER_PORT_6_TX                PinIO::PIN_FN_2
00274 #define PIN_29_XBAR1_INOUT08                PinIO::PIN_FN_3
00275 //#define PIN_29_FLEXSPIB_SS0_B               PinIO::PIN_FN_4
00276 #define PIN_29_GPIO3_IO16                   PinIO::PIN_FN_5
00277 #define PIN_29_CCM_CLKO1                    PinIO::PIN_FN_6
00278 #define PIN_29_ENET2_RDATA01                PinIO::PIN_FN_8
00279
00280 // WAKEUP
00281 #define PIN_30_GPIO5_IO00                   PinIO::PIN_FN_5
00282 #define PIN_30_NMI                          PinIO::PIN_FN_7
00283
00284 // PMIC_STBY_REQ
00285 #define PIN_31_CCM_PMIC_STBY_REQ            PinIO::PIN_FN_0
00286 #define PIN_31_GPIO5_IO02                   PinIO::PIN_FN_5
00287
00288 // GPIO_AD_B1_00
00289 #define PIN_40_USB_OTG2_ID                  PinIO::PIN_FN_0
00290 #define PIN_40_QTIMER3_TIMER0               PinIO::PIN_FN_1
00291 #define PIN_40_LPUART2_CTS_B                PinIO::PIN_FN_2
00292 #define PIN_40_SER_PORT_1_CTS               PinIO::PIN_FN_2
00293 #define PIN_40_LPI2C1_SCL                   PinIO::PIN_FN_3
00294 #define PIN_40_WDOG1_B                      PinIO::PIN_FN_4
00295 #define PIN_40_GPIO1_IO16                   PinIO::PIN_FN_5
00296 #define PIN_40_USDHC1_WP                    PinIO::PIN_FN_6
00297 #define PIN_40_KPP_ROW07                    PinIO::PIN_FN_7
00298 #define PIN_40_ENET2_1588_EVENT0_OUT        PinIO::PIN_FN_8
00299 #define PIN_40_FLEXIO3_FLEXIO00             PinIO::PIN_FN_9
00300
00301 // GPIO_AD_B1_01
00302 #define PIN_41_USB_OTG1_PWR                 PinIO::PIN_FN_0
00303 #define PIN_41_QTIMER3_TIMER1               PinIO::PIN_FN_1
00304 #define PIN_41_LPUART2_RTS_B                PinIO::PIN_FN_2
00305 #define PIN_41_SER_PORT_1_RTS               PinIO::PIN_FN_2
00306 #define PIN_41_LPI2C1_SDA                   PinIO::PIN_FN_3
00307 #define PIN_41_CCM_PMIC_READY               PinIO::PIN_FN_4
00308 #define PIN_41_GPIO1_IO17                   PinIO::PIN_FN_5
00309 #define PIN_41_USDHC1_VSELECT               PinIO::PIN_FN_6
00310 #define PIN_41_KPP_COL07                    PinIO::PIN_FN_7
00311 #define PIN_41_ENET2_1588_EVENT0_IN         PinIO::PIN_FN_8
00312 #define PIN_41_FLEXIO3_FLEXIO01             PinIO::PIN_FN_9
00313
00314 // GPIO_AD_B1_02
00315 #define PIN_42_USB_OTG1_ID                  PinIO::PIN_FN_0
00316 #define PIN_42_QTIMER3_TIMER2               PinIO::PIN_FN_1
00317 #define PIN_42_LPUART2_TX                   PinIO::PIN_FN_2
00318 #define PIN_42_SER_PORT_1_TX                PinIO::PIN_FN_2
00319 #define PIN_42_SPDIF_OUT                    PinIO::PIN_FN_3
00320 #define PIN_42_ENET_1588_EVENT2_OUT         PinIO::PIN_FN_4
00321 #define PIN_42_GPIO1_IO18                   PinIO::PIN_FN_5
00322 #define PIN_42_USDHC1_CD_B                  PinIO::PIN_FN_6
00323 #define PIN_42_KPP_ROW06                    PinIO::PIN_FN_7
00324 #define PIN_42_GPT2_CLK                     PinIO::PIN_FN_8
00325 #define PIN_42_FLEXIO3_FLEXIO02             PinIO::PIN_FN_9
00326
00327 // GPIO_AD_B1_03
00328 #define PIN_43_USB_OTG1_OC                  PinIO::PIN_FN_0
00329 #define PIN_43_QTIMER3_TIMER3               PinIO::PIN_FN_1
00330 #define PIN_43_LPUART2_RX                   PinIO::PIN_FN_2
00331 #define PIN_43_SER_PORT_1_RX                PinIO::PIN_FN_2
00332 #define PIN_43_SPDIF_IN                     PinIO::PIN_FN_3
00333 #define PIN_43_ENET_1588_EVENT2_IN          PinIO::PIN_FN_4
00334 #define PIN_43_GPIO1_IO19                   PinIO::PIN_FN_5
00335 //#define PIN_43_USDHC2_CD_B                  PinIO::PIN_FN_6
00336 #define PIN_43_KPP_COL06                    PinIO::PIN_FN_7
00337 #define PIN_43_GPT2_CAPTURE1                PinIO::PIN_FN_8
00338 #define PIN_43_FLEXIO3_FLEXIO03             PinIO::PIN_FN_9
00339
00340 // GPIO_AD_B1_04
00341 //#define PIN_44_FLEXSPIB_DATA03              PinIO::PIN_FN_0
```

```
00342 #define PIN_44_ENET_MDC                 PinIO::PIN_FN_1
00343 #define PIN_44_LPUART3_CTS_B            PinIO::PIN_FN_2
00344 #define PIN_44_SER_PORT_0_CTS           PinIO::PIN_FN_2
00345 #define PIN_44_SPDIF_SR_CLK             PinIO::PIN_FN_3
00346 #define PIN_44_CSI_PIXCLK               PinIO::PIN_FN_4
00347 #define PIN_44_GPIO1_IO20               PinIO::PIN_FN_5
00348 //#define PIN_44_USDHC2_DATA0             PinIO::PIN_FN_6
00349 #define PIN_44_KPP_ROW05                PinIO::PIN_FN_7
00350 #define PIN_44_GPT2_CAPTURE2            PinIO::PIN_FN_8
00351 #define PIN_44_FLEXIO3_FLEXIO04         PinIO::PIN_FN_9
00352
00353 // GPIO_AD_B1_05
00354 //#define PIN_45_FLEXSPIB_DATA02          PinIO::PIN_FN_0
00355 #define PIN_45_ENET_MDIO                PinIO::PIN_FN_1
00356 #define PIN_45_LPUART3_RTS_B            PinIO::PIN_FN_2
00357 #define PIN_45_SER_PORT_0_RTS           PinIO::PIN_FN_2
00358 #define PIN_45_SPDIF_OUT                PinIO::PIN_FN_3
00359 #define PIN_45_CSI_MCLK                 PinIO::PIN_FN_4
00360 #define PIN_45_GPIO1_IO21               PinIO::PIN_FN_5
00361 //#define PIN_45_USDHC2_DATA1             PinIO::PIN_FN_6
00362 #define PIN_45_KPP_COL05                PinIO::PIN_FN_7
00363 #define PIN_45_GPT2_COMPARE1            PinIO::PIN_FN_8
00364 #define PIN_45_FLEXIO3_FLEXIO05         PinIO::PIN_FN_9
00365
00366 // GPIO_AD_B1_06
00367 //#define PIN_46_FLEXSPIB_DATA01          PinIO::PIN_FN_0
00368 #define PIN_46_LPI2C3_SDA               PinIO::PIN_FN_1
00369 #define PIN_46_LPUART3_TX               PinIO::PIN_FN_2
00370 #define PIN_46_SER_PORT_0_TX            PinIO::PIN_FN_2
00371 #define PIN_46_SPDIF_LOCK               PinIO::PIN_FN_3
00372 #define PIN_46_CSI_VSYNC                PinIO::PIN_FN_4
00373 #define PIN_46_GPIO1_IO22               PinIO::PIN_FN_5
00374 //#define PIN_46_USDHC2_DATA2             PinIO::PIN_FN_6
00375 #define PIN_46_KPP_ROW04                PinIO::PIN_FN_7
00376 #define PIN_46_GPT2_COMPARE2            PinIO::PIN_FN_8
00377 #define PIN_46_FLEXIO3_FLEXIO06         PinIO::PIN_FN_9
00378
00379 // GPIO_AD_B1_07
00380 //#define PIN_47_FLEXSPIB_DATA00          PinIO::PIN_FN_0
00381 #define PIN_47_LPI2C3_SCL               PinIO::PIN_FN_1
00382 #define PIN_47_LPUART3_RX               PinIO::PIN_FN_2
00383 #define PIN_47_SER_PORT_0_RX            PinIO::PIN_FN_2
00384 #define PIN_47_SPDIF_EXT_CLK            PinIO::PIN_FN_3
00385 #define PIN_47_CSI_HSYNC                PinIO::PIN_FN_4
00386 #define PIN_47_GPIO1_IO23               PinIO::PIN_FN_5
00387 //#define PIN_47_USDHC2_DATA3             PinIO::PIN_FN_6
00388 #define PIN_47_KPP_COL04                PinIO::PIN_FN_7
00389 #define PIN_47_GPT2_COMPARE3            PinIO::PIN_FN_8
00390 #define PIN_47_FLEXIO3_FLEXIO07         PinIO::PIN_FN_9
00391
00392 // GPIO_B0_01
00393 //#define PIN_50_LCD_ENABLE               PinIO::PIN_FN_0
00394 #define PIN_50_QTIMER1_TIMER1           PinIO::PIN_FN_1
00395 #define PIN_50_MQS_LEFT                 PinIO::PIN_FN_2
00396 #define PIN_50_LPSPI4_SDI               PinIO::PIN_FN_3
00397 #define PIN_50_FLEXIO2_FLEXIO01         PinIO::PIN_FN_4
00398 #define PIN_50_GPIO2_IO01               PinIO::PIN_FN_5
00399 #define PIN_50_SEMC_CSX02               PinIO::PIN_FN_6
00400 #define PIN_50_ENET2_MDIO               PinIO::PIN_FN_8
00401
00402 // GPIO_B0_00
00403 //#define PIN_51_LCD_CLK                  PinIO::PIN_FN_0
00404 #define PIN_51_QTIMER1_TIMER0           PinIO::PIN_FN_1
00405 #define PIN_51_MQS_RIGHT                PinIO::PIN_FN_2
00406 #define PIN_51_LPSPI4_PCS0              PinIO::PIN_FN_3
00407 #define PIN_51_FLEXIO2_FLEXIO00         PinIO::PIN_FN_4
00408 #define PIN_51_GPIO2_IO00               PinIO::PIN_FN_5
00409 #define PIN_51_SEMC_CSX01               PinIO::PIN_FN_6
00410 #define PIN_51_ENET2_MDC                PinIO::PIN_FN_8
00411
00412 // GPIO_B0_15
00413 //#define PIN_52_LCD_DATA11               PinIO::PIN_FN_0
00414 #define PIN_52_XBAR1_INOUT13            PinIO::PIN_FN_1
00415 #define PIN_52_ARM_RXEV                 PinIO::PIN_FN_2
00416 #define PIN_52_SAI1_RX_BCLK             PinIO::PIN_FN_3
00417 #define PIN_52_FLEXIO2_FLEXIO15         PinIO::PIN_FN_4
00418 #define PIN_52_GPIO2_IO15               PinIO::PIN_FN_5
00419 #define PIN_52_SRC_BOOT_CFG11           PinIO::PIN_FN_6
00420 #define PIN_52_ENET2_TX_CLK             PinIO::PIN_FN_8
00421 #define PIN_52_ENET2_REF_CLK2           PinIO::PIN_FN_9
00422
00423 // GPIO_B0_14
00424 //#define PIN_53_LCD_DATA10               PinIO::PIN_FN_0
00425 #define PIN_53_XBAR1_INOUT12            PinIO::PIN_FN_1
00426 #define PIN_53_ARM_TXEV                 PinIO::PIN_FN_2
00427 #define PIN_53_SAI1_RX_SYNC             PinIO::PIN_FN_3
00428 #define PIN_53_FLEXIO2_FLEXIO14         PinIO::PIN_FN_4
```

```
00429 #define PIN_53_GPIO2_IO14                PinIO::PIN_FN_5
00430 #define PIN_53_SRC_BOOT_CFG10            PinIO::PIN_FN_6
00431 #define PIN_53_ENET2_TX_EN               PinIO::PIN_FN_8
00432
00433 // GPIO_B0_13
00434 //#define PIN_54_LCD_DATA09                PinIO::PIN_FN_0
00435 #define PIN_54_XBAR1_INOUT11             PinIO::PIN_FN_1
00436 #define PIN_54_ARM_TRACE_SWO             PinIO::PIN_FN_2
00437 #define PIN_54_SAI1_MCLK                 PinIO::PIN_FN_3
00438 #define PIN_54_FLEXIO2_FLEXIO13          PinIO::PIN_FN_4
00439 #define PIN_54_GPIO2_IO13                PinIO::PIN_FN_5
00440 #define PIN_54_SRC_BOOT_CFG09            PinIO::PIN_FN_6
00441 #define PIN_54_ENET2_TDATA01             PinIO::PIN_FN_8
00442
00443 // GPIO_B0_12
00444 //#define PIN_55_LCD_DATA08                PinIO::PIN_FN_0
00445 #define PIN_55_XBAR1_INOUT10             PinIO::PIN_FN_1
00446 #define PIN_55_ARM_TRACE_CLK             PinIO::PIN_FN_2
00447 #define PIN_55_SAI1_TX_DATA01            PinIO::PIN_FN_3
00448 #define PIN_55_FLEXIO2_FLEXIO12          PinIO::PIN_FN_4
00449 #define PIN_55_GPIO2_IO12                PinIO::PIN_FN_5
00450 #define PIN_55_SRC_BOOT_CFG08            PinIO::PIN_FN_6
00451 #define PIN_55_ENET2_TDATA00             PinIO::PIN_FN_8
00452
00453 // GPIO_B1_00
00454 //#define PIN_56_LCD_DATA12                PinIO::PIN_FN_0
00455 #define PIN_56_XBAR1_INOUT14             PinIO::PIN_FN_1
00456 #define PIN_56_LPUART4_TX                PinIO::PIN_FN_2
00457 #define PIN_56_SER_PORT_2_TX             PinIO::PIN_FN_2
00458 #define PIN_56_SAI1_RX_DATA00            PinIO::PIN_FN_3
00459 #define PIN_56_FLEXIO2_FLEXIO16          PinIO::PIN_FN_4
00460 #define PIN_56_GPIO2_IO16                PinIO::PIN_FN_5
00461 #define PIN_56_FLEXPWM1_PWMA3            PinIO::PIN_FN_6
00462 #define PIN_56_ENET2_RX_ER               PinIO::PIN_FN_8
00463 #define PIN_56_FLEXIO3_FLEXIO16          PinIO::PIN_FN_9
00464
00465 // GPIO_B1_02
00466 //#define PIN_57_LCD_DATA14                PinIO::PIN_FN_0
00467 #define PIN_57_XBAR1_INOUT16             PinIO::PIN_FN_1
00468 #define PIN_57_LPSPI4_PCS2               PinIO::PIN_FN_2
00469 #define PIN_57_SAI1_TX_BCLK              PinIO::PIN_FN_3
00470 #define PIN_57_FLEXIO2_FLEXIO18          PinIO::PIN_FN_4
00471 #define PIN_57_GPIO2_IO18                PinIO::PIN_FN_5
00472 #define PIN_57_FLEXPWM2_PWMA3            PinIO::PIN_FN_6
00473 #define PIN_57_ENET2_RDATA01             PinIO::PIN_FN_8
00474 #define PIN_57_FLEXIO3_FLEXIO18          PinIO::PIN_FN_9
00475
00476 // GPIO_B1_01
00477 //#define PIN_58_LCD_DATA13                PinIO::PIN_FN_0
00478 #define PIN_58_XBAR1_INOUT15             PinIO::PIN_FN_1
00479 #define PIN_58_LPUART4_RX                PinIO::PIN_FN_2
00480 #define PIN_58_SER_PORT_2_RX             PinIO::PIN_FN_2
00481 #define PIN_58_SAI1_TX_DATA00            PinIO::PIN_FN_3
00482 #define PIN_58_FLEXIO2_FLEXIO17          PinIO::PIN_FN_4
00483 #define PIN_58_GPIO2_IO17                PinIO::PIN_FN_5
00484 #define PIN_58_FLEXPWM1_PWMB3            PinIO::PIN_FN_6
00485 #define PIN_58_ENET2_RDATA00             PinIO::PIN_FN_8
00486 #define PIN_58_FLEXIO3_FLEXIO17          PinIO::PIN_FN_9
00487
00488 // GPIO_B1_03
00489 //#define PIN_59_LCD_DATA15                PinIO::PIN_FN_0
00490 #define PIN_59_XBAR1_INOUT17             PinIO::PIN_FN_1
00491 #define PIN_59_LPSPI4_PCS1               PinIO::PIN_FN_2
00492 #define PIN_59_SAI1_TX_SYNC              PinIO::PIN_FN_3
00493 #define PIN_59_FLEXIO2_FLEXIO19          PinIO::PIN_FN_4
00494 #define PIN_59_GPIO2_IO19                PinIO::PIN_FN_5
00495 #define PIN_59_FLEXPWM2_PWMB3            PinIO::PIN_FN_6
00496 #define PIN_59_ENET2_RX_EN               PinIO::PIN_FN_8
00497 #define PIN_59_FLEXIO3_FLEXIO19          PinIO::PIN_FN_9
00498
00499 // GPIO_AD_B0_14
00500 #define PIN_61_USB_OTG2_OC               PinIO::PIN_FN_0
00501 #define PIN_61_XBAR1_IN24                PinIO::PIN_FN_1
00502 //#define PIN_61_LPUART1_CTS_B             PinIO::PIN_FN_2
00503 #define PIN_61_ENET_1588_EVENT0_OUT      PinIO::PIN_FN_3
00504 #define PIN_61_CSI_VSYNC                 PinIO::PIN_FN_4
00505 #define PIN_61_GPIO1_IO14                PinIO::PIN_FN_5
00506 #define PIN_61_FLEXCAN2_TX               PinIO::PIN_FN_6
00507 #define PIN_61_FLEXCAN3_TX               PinIO::PIN_FN_8
00508
00509 // GPIO_EMC_41
00510 #define PIN_67_SEMC_CSX00                PinIO::PIN_FN_0
00511 #define PIN_67_GPT2_CAPTURE1             PinIO::PIN_FN_1
00512 #define PIN_67_LPSPI1_PCS3               PinIO::PIN_FN_2
00513 #define PIN_67_USB_OTG2_PWR              PinIO::PIN_FN_3
00514 #define PIN_67_ENET_MDIO                 PinIO::PIN_FN_4
00515 #define PIN_67_GPIO3_IO27                PinIO::PIN_FN_5
```

```
00516 #define PIN_67_USDHC1_VSELECT              PinIO::PIN_FN_6
00517
00518 // GPIO_EMC_40
00519 #define PIN_68_SEMC_RDY                    PinIO::PIN_FN_0
00520 #define PIN_68_GPT2_CAPTURE2               PinIO::PIN_FN_1
00521 #define PIN_68_LPSPI1_PCS2                 PinIO::PIN_FN_2
00522 #define PIN_68_USB_OTG2_OC                 PinIO::PIN_FN_3
00523 #define PIN_68_ENET_MDC                    PinIO::PIN_FN_4
00524 #define PIN_68_GPIO3_IO26                  PinIO::PIN_FN_5
00525 //#define PIN_68_USDHC2_RESET_B              PinIO::PIN_FN_6
00526 #define PIN_68_SEMC_CLK5                   PinIO::PIN_FN_9
00527
00528 // GPIO_EMC_39
00529 #define PIN_69_SEMC_DQS                    PinIO::PIN_FN_0
00530 #define PIN_69_FLEXPWM1_PWMB3              PinIO::PIN_FN_1
00531 #define PIN_69_LPUART8_RX                  PinIO::PIN_FN_2
00532 #define PIN_69_SER_PORT_6_RX               PinIO::PIN_FN_2
00533 #define PIN_69_SAI3_TX_SYNC                PinIO::PIN_FN_3
00534 #define PIN_69_WDOG1_WDOG_B                PinIO::PIN_FN_4
00535 #define PIN_69_GPIO3_IO25                  PinIO::PIN_FN_5
00536 //#define PIN_69_USDHC2_CD_B                 PinIO::PIN_FN_6
00537 #define PIN_69_ENET2_MDIO                  PinIO::PIN_FN_8
00538 #define PIN_69_SEMC_DQS4                   PinIO::PIN_FN_9
00539
00540 // GPIO_EMC_38
00541 #define PIN_70_SEMC_DM01                   PinIO::PIN_FN_0
00542 #define PIN_70_FLEXPWM1_PWMA3              PinIO::PIN_FN_1
00543 #define PIN_70_LPUART8_TX                  PinIO::PIN_FN_2
00544 #define PIN_70_SER_PORT_6_TX               PinIO::PIN_FN_2
00545 #define PIN_70_SAI3_TX_BCLK                PinIO::PIN_FN_3
00546 #define PIN_70_CSI_FIELD                   PinIO::PIN_FN_4
00547 #define PIN_70_GPIO3_IO24                  PinIO::PIN_FN_5
00548 //#define PIN_70_USDHC2_VSELECT              PinIO::PIN_FN_6
00549 #define PIN_70_ENET2_MDC                   PinIO::PIN_FN_8
00550
00551 // GPIO_EMC_37
00552 #define PIN_71_SEMC_DATA15                 PinIO::PIN_FN_0
00553 #define PIN_71_XBAR1_IN23                  PinIO::PIN_FN_1
00554 #define PIN_71_GPT1_COMPARE3               PinIO::PIN_FN_2
00555 #define PIN_71_SAI3_MCLK                   PinIO::PIN_FN_3
00556 #define PIN_71_CSI_DATA16                  PinIO::PIN_FN_4
00557 #define PIN_71_GPIO3_IO23                  PinIO::PIN_FN_5
00558 //#define PIN_71_USDHC2_WP                   PinIO::PIN_FN_6
00559 #define PIN_71_ENET2_RX_EN                 PinIO::PIN_FN_8
00560 #define PIN_71_FLEXCAN3_RX                 PinIO::PIN_FN_9
00561
00562 // GPIO_EMC_36
00563 #define PIN_72_SEMC_DATA14                 PinIO::PIN_FN_0
00564 #define PIN_72_XBAR1_IN22                  PinIO::PIN_FN_1
00565 #define PIN_72_GPT1_COMPARE2               PinIO::PIN_FN_2
00566 #define PIN_72_SAI3_TX_DATA                PinIO::PIN_FN_3
00567 #define PIN_72_CSI_DATA17                  PinIO::PIN_FN_4
00568 #define PIN_72_GPIO3_IO22                  PinIO::PIN_FN_5
00569 #define PIN_72_USDHC1_WP                   PinIO::PIN_FN_6
00570 #define PIN_72_ENET2_RDATA01               PinIO::PIN_FN_8
00571 #define PIN_72_FLEXCAN3_TX                 PinIO::PIN_FN_9
00572
00573 // GPIO_EMC_35
00574 #define PIN_73_SEMC_DATA13                 PinIO::PIN_FN_0
00575 #define PIN_73_XBAR1_INOUT18               PinIO::PIN_FN_1
00576 #define PIN_73_GPT1_COMPARE1               PinIO::PIN_FN_2
00577 #define PIN_73_SAI3_RX_BCLK                PinIO::PIN_FN_3
00578 #define PIN_73_CSI_DATA18                  PinIO::PIN_FN_4
00579 #define PIN_73_GPIO3_IO21                  PinIO::PIN_FN_5
00580 #define PIN_73_USDHC1_CD_B                 PinIO::PIN_FN_6
00581 #define PIN_73_ENET2_RDATA00               PinIO::PIN_FN_8
00582
00583 // GPIO_EMC_34
00584 #define PIN_74_SEMC_DATA12                 PinIO::PIN_FN_0
00585 #define PIN_74_FLEXPWM3_PWMB2              PinIO::PIN_FN_1
00586 #define PIN_74_USDHC1_VSELECT              PinIO::PIN_FN_2
00587 #define PIN_74_SAI3_RX_SYNC                PinIO::PIN_FN_3
00588 #define PIN_74_CSI_DATA19                  PinIO::PIN_FN_4
00589 #define PIN_74_GPIO3_IO20                  PinIO::PIN_FN_5
00590 #define PIN_74_ENET2_RX_ER                 PinIO::PIN_FN_8
00591
00592 // GPIO_EMC_33
00593 #define PIN_75_SEMC_DATA11                 PinIO::PIN_FN_0
00594 #define PIN_75_FLEXPWM3_PWMA2              PinIO::PIN_FN_1
00595 #define PIN_75_USDHC1_RESET_B              PinIO::PIN_FN_2
00596 #define PIN_75_SAI3_RX_DATA                PinIO::PIN_FN_3
00597 #define PIN_75_CSI_DATA20                  PinIO::PIN_FN_4
00598 #define PIN_75_GPIO3_IO19                  PinIO::PIN_FN_5
00599 #define PIN_75_ENET2_TX_CLK                PinIO::PIN_FN_8
00600 #define PIN_75_ENET2_REF_CLK2              PinIO::PIN_FN_9
00601
00602 // GPIO_EMC_32
```

```
00603 #define PIN_76_SEMC_DATA10                PinIO::PIN_FN_0
00604 #define PIN_76_FLEXPWM3_PWMB1             PinIO::PIN_FN_1
00605 #define PIN_76_LPUART7_RX                 PinIO::PIN_FN_2
00606 #define PIN_76_SER_PORT_5_RX              PinIO::PIN_FN_2
00607 #define PIN_76_CCM_PMIC_RDY               PinIO::PIN_FN_3
00608 #define PIN_76_CSI_DATA21                 PinIO::PIN_FN_4
00609 #define PIN_76_GPIO3_IO18                 PinIO::PIN_FN_5
00610 #define PIN_76_ENET2_TX_EN                PinIO::PIN_FN_8
00611
00612 // GPIO_EMC_31
00613 #define PIN_77_SEMC_DATA09                PinIO::PIN_FN_0
00614 #define PIN_77_FLEXPWM3_PWMA1             PinIO::PIN_FN_1
00615 #define PIN_77_LPUART7_TX                 PinIO::PIN_FN_2
00616 #define PIN_77_SER_PORT_5_TX              PinIO::PIN_FN_2
00617 #define PIN_77_LPSPI1_PCS1                PinIO::PIN_FN_3
00618 #define PIN_77_CSI_DATA22                 PinIO::PIN_FN_4
00619 #define PIN_77_GPIO4_IO31                 PinIO::PIN_FN_5
00620 #define PIN_77_ENET2_TDATA01              PinIO::PIN_FN_8
00621
00622 // GPIO_EMC_30
00623 #define PIN_78_SEMC_DATA08                PinIO::PIN_FN_0
00624 #define PIN_78_FLEXPWM3_PWMB0             PinIO::PIN_FN_1
00625 #define PIN_78_LPUART6_CTS_B              PinIO::PIN_FN_2
00626 #define PIN_78_SER_PORT_4_CTS             PinIO::PIN_FN_2
00627 #define PIN_78_LPSPI1_PCS0                PinIO::PIN_FN_3
00628 #define PIN_78_CSI_DATA23                 PinIO::PIN_FN_4
00629 #define PIN_78_GPIO4_IO30                 PinIO::PIN_FN_5
00630 #define PIN_78_ENET2_TDATA00              PinIO::PIN_FN_8
00631
00632 // GPIO_EMC_29
00633 #define PIN_80_SEMC_CS0                   PinIO::PIN_FN_0
00634 #define PIN_80_FLEXPWM3_PWMA0             PinIO::PIN_FN_1
00635 #define PIN_80_LPUART6_RTS_B              PinIO::PIN_FN_2
00636 #define PIN_80_SER_PORT_4_RTS             PinIO::PIN_FN_2
00637 #define PIN_80_LPSPI1_SDI                 PinIO::PIN_FN_3
00638 #define PIN_80_FLEXIO1_FLEXIO15           PinIO::PIN_FN_4
00639 #define PIN_80_GPIO4_IO29                 PinIO::PIN_FN_5
00640 #define PIN_80_FLEXSPI2_A_DATA03          PinIO::PIN_FN_8
00641
00642 // GPIO_EMC_28
00643 #define PIN_81_SEMC_WE                    PinIO::PIN_FN_0
00644 #define PIN_81_FLEXPWM1_PWMB2             PinIO::PIN_FN_1
00645 #define PIN_81_LPUART5_CTS_B              PinIO::PIN_FN_2
00646 #define PIN_81_SER_PORT_3_CTS             PinIO::PIN_FN_2
00647 #define PIN_81_LPSPI1_SDO                 PinIO::PIN_FN_3
00648 #define PIN_81_FLEXIO1_FLEXIO14           PinIO::PIN_FN_4
00649 #define PIN_81_GPIO4_IO28                 PinIO::PIN_FN_5
00650 #define PIN_81_FLEXSPI2_A_DATA02          PinIO::PIN_FN_8
00651
00652 // GPIO_EMC_27
00653 #define PIN_82_SEMC_CKE                   PinIO::PIN_FN_0
00654 #define PIN_82_FLEXPWM1_PWMA2             PinIO::PIN_FN_1
00655 #define PIN_82_LPUART5_RTS_B              PinIO::PIN_FN_2
00656 #define PIN_81_SER_PORT_3_RTS             PinIO::PIN_FN_2
00657 #define PIN_82_LPSPI1_SCK                 PinIO::PIN_FN_3
00658 #define PIN_82_FLEXIO1_FLEXIO13           PinIO::PIN_FN_4
00659 #define PIN_82_GPIO4_IO27                 PinIO::PIN_FN_5
00660 #define PIN_82_FLEXSPI2_A_DATA01          PinIO::PIN_FN_8
00661
00662 // GPIO_EMC_26
00663 #define PIN_83_SEMC_CLK                   PinIO::PIN_FN_0
00664 #define PIN_83_FLEXPWM1_PWMB1             PinIO::PIN_FN_1
00665 #define PIN_83_LPUART6_RX                 PinIO::PIN_FN_2
00666 #define PIN_83_SER_PORT_4_RX              PinIO::PIN_FN_2
00667 #define PIN_83_ENET_RX_ER                 PinIO::PIN_FN_3
00668 #define PIN_83_FLEXIO1_FLEXIO12           PinIO::PIN_FN_4
00669 #define PIN_83_GPIO4_IO26                 PinIO::PIN_FN_5
00670 #define PIN_83_FLEXSPI2_A_DATA00          PinIO::PIN_FN_8
00671
00672 // GPIO_EMC_25
00673 #define PIN_84_SEMC_RAS                   PinIO::PIN_FN_0
00674 #define PIN_84_FLEXPWM1_PWMA1             PinIO::PIN_FN_1
00675 #define PIN_84_LPUART6_TX                 PinIO::PIN_FN_2
00676 #define PIN_84_SER_PORT_4_TX              PinIO::PIN_FN_2
00677 #define PIN_84_ENET_TX_CLK                PinIO::PIN_FN_3
00678 #define PIN_84_ENET_REF_CLK               PinIO::PIN_FN_4
00679 #define PIN_84_GPIO4_IO25                 PinIO::PIN_FN_5
00680 #define PIN_84_FLEXSPI2_A_SCLK            PinIO::PIN_FN_8
00681
00682 // GPIO_EMC_24
00683 #define PIN_85_SEMC_CAS                   PinIO::PIN_FN_0
00684 #define PIN_85_FLEXPWM1_PWMB0             PinIO::PIN_FN_1
00685 #define PIN_85_LPUART5_RX                 PinIO::PIN_FN_2
00686 #define PIN_85_SER_PORT_3_RX              PinIO::PIN_FN_2
00687 #define PIN_85_ENET_TX_EN                 PinIO::PIN_FN_3
00688 #define PIN_85_GPT1_CAPTURE1              PinIO::PIN_FN_4
00689 #define PIN_85_GPIO4_IO24                 PinIO::PIN_FN_5
```

```
00690 #define PIN_85_FLEXSPI2_A_SS0_B             PinIO::PIN_FN_8
00691
00692 // GPIO_EMC_23
00693 #define PIN_86_SEMC_ADDR10                  PinIO::PIN_FN_0
00694 #define PIN_86_FLEXPWM1_PWMA0               PinIO::PIN_FN_1
00695 #define PIN_86_LPUART5_TX                   PinIO::PIN_FN_2
00696 #define PIN_86_SER_PORT_3_TX                PinIO::PIN_FN_2
00697 #define PIN_86_ENET_RX_EN                   PinIO::PIN_FN_3
00698 #define PIN_86_GPT1_CAPTURE2                PinIO::PIN_FN_4
00699 #define PIN_86_GPIO4_IO23                   PinIO::PIN_FN_5
00700 #define PIN_86_FLEXSPI2_A_DQS               PinIO::PIN_FN_8
00701
00702 // GPIO_EMC_20
00703 #define PIN_87_SEMC_ADDR12                  PinIO::PIN_FN_0
00704 #define PIN_87_FLEXPWM2_PWMB3               PinIO::PIN_FN_1
00705 #define PIN_87_LPUART4_RX                   PinIO::PIN_FN_2
00706 #define PIN_87_SER_PORT_2_RX                PinIO::PIN_FN_2
00707 #define PIN_87_ENET_RDATA00                 PinIO::PIN_FN_3
00708 #define PIN_87_QTIMER2_TIMER1               PinIO::PIN_FN_4
00709 #define PIN_87_GPIO4_IO20                   PinIO::PIN_FN_5
00710
00711 // GPIO_EMC_19
00712 #define PIN_88_SEMC_ADDR11                  PinIO::PIN_FN_0
00713 #define PIN_88_FLEXPWM2_PWMA3               PinIO::PIN_FN_1
00714 #define PIN_88_LPUART4_TX                   PinIO::PIN_FN_2
00715 #define PIN_88_SER_PORT_2_TX                PinIO::PIN_FN_2
00716 #define PIN_88_ENET_RDATA01                 PinIO::PIN_FN_3
00717 #define PIN_88_QTIMER2_TIMER0               PinIO::PIN_FN_4
00718 #define PIN_88_GPIO4_IO19                   PinIO::PIN_FN_5
00719 #define PIN_88_SNVS_VIO_5                   PinIO::PIN_FN_6
00720
00721
00722
00723 // These pins are secondary I/O and are directly tied to the same trace as
00724 //  their corresponding pin, *USE WITH CARE*
00725
00726 // PIN_89 is electrically tied to PIN_53
00727 // GPIO_B0_02
00728 //#define PIN_89_LCD_HSYNC                    PinIO::PIN_FN_0
00729 #define PIN_89_QTIMER1_TIMER2               PinIO::PIN_FN_1
00730 #define PIN_89_FLEXCAN1_TX                  PinIO::PIN_FN_2
00731 #define PIN_89_LPSPI4_SOUT                  PinIO::PIN_FN_3
00732 #define PIN_89_FLEXIO2_D02                  PinIO::PIN_FN_4
00733 #define PIN_89_GPIO2_IO02                   PinIO::PIN_FN_5
00734 #define PIN_89_SEMC_CSX3                    PinIO::PIN_FN_6
00735 #define PIN_89_ENET2_1588_EVENT0_OUT        PinIO::PIN_FN_8
00736
00737 // PIN_90 is electrically tied to PIN_59
00738 // GPIO_B0_03
00739 //#define PIN_90_LCD_VSYNC                    PinIO::PIN_FN_0
00740 #define PIN_90_QTIMER2_TIMER0               PinIO::PIN_FN_1
00741 #define PIN_90_FLEXCAN1_RX                  PinIO::PIN_FN_2
00742 #define PIN_90_LPSPI4_SCK                   PinIO::PIN_FN_3
00743 #define PIN_90_FLEXIO2_D03                  PinIO::PIN_FN_4
00744 #define PIN_90_GPIO2_IO03                   PinIO::PIN_FN_5
00745 #define PIN_90_WDOG2_RESET_B_DEB            PinIO::PIN_FN_6
00746 #define PIN_90_ENET2_1588_EVENT0_IN         PinIO::PIN_FN_8
00747
00748 // PIN_91 is electrically tied to PIN_84
00749 // GPIO_EMC_21
00750 #define PIN_91_SEMC_BA0                     PinIO::PIN_FN_0
00751 #define PIN_91_FLEXPWM3_PWMA3               PinIO::PIN_FN_1
00752 #define PIN_91_LPI2C3_SDA                   PinIO::PIN_FN_2
00753 #define PIN_91_ENET_TDATA01                 PinIO::PIN_FN_3
00754 #define PIN_91_QTIMER2_TIMER2               PinIO::PIN_FN_4
00755 #define PIN_91_GPIO4_IO21                   PinIO::PIN_FN_5
00756
00757 // PIN_92 is electrically tied to PIN_83
00758 // GPIO_EMC_22
00759 #define PIN_92_SEMC_BA1                     PinIO::PIN_FN_0
00760 #define PIN_92_FLEXPWM3_PWMB3               PinIO::PIN_FN_1
00761 #define PIN_92_LPI2C3_SCL                   PinIO::PIN_FN_2
00762 #define PIN_92_ENET_TDATA00                 PinIO::PIN_FN_3
00763 #define PIN_92_QTIMER2_TIMER3               PinIO::PIN_FN_4
00764 #define PIN_92_GPIO4_IO22                   PinIO::PIN_FN_5
00765 #define PIN_92_FLEXSPI2_A_SS1_B             PinIO::PIN_FN_8
00766
```

24.535 MOD5441X/include/pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_PINS_H_
```

```
00006 #define _NB_PINS_H_
00007
00008 #include <pinconstant.h>
00009 #include <cpu_pins.h>
00010
00011 #define PINS
00012 //namespace PINS
00013 //{
00028 //class PinIO
00029 //{
00030 //    uint8_t connector, pinnum;
00031 //    PinIO(int j, int n)
00032 //    {
00033 //        connector = j;
00034 //        pinnum = n;
00035 //    };
00036 //
00037 //   public:
00038 //    void set(BOOL = TRUE);        // Set output high
00039 //    void clr() { set(FALSE); };   // Set output low
00040 //    BOOL toggle();                // Set output state to the opposite of current state
00041 //    BOOL read();                  // Read hi/low state of input pin
00042 //    BOOL readBack();              // Read pin state without affecting direction
00043 //    void hiz() { read(); };       // Set output to tristate
00044 //    void drive();                 // Turn output on (opposite of tristate)
00045 //    void function(int ft);        // Set pin to special function
00046 //    int getFunction();            // Get the special function the pin is set to
00047 //    PinIO &operator=(BOOL b)
00048 //    {
00049 //        set(b);
00050 //        return *this;
00051 //    };
00052 //    PinIO &operator=(int i)
00053 //    {
00054 //        set(i);
00055 //        return *this;
00056 //    };
00057 //    operator int() { return read(); };           // Read and return int value
00058 //    operator BOOL() { return read(); };          // Read and return BOOL value
00059 //    operator bool() { return (read() != 0); };   // Read and return boolean value
00060 //    friend class PinIOJ1Array;
00061 //    friend class PinIOJ2Array;
00062 //};
00063
00064 class PinIOJ1Array
00065 {
00066    public:
00067     PinIO operator[](int n);// { return PinIO(1, n); };
00068 };
00069
00070 class PinIOJ2Array
00071 {
00072    public:
00073     PinIO operator[](int n);// { return PinIO(2, n); };
00074 };
00075 //}   // namespace PINS
00076
00077 extern PinIOJ1Array J1;
00078 extern PinIOJ2Array J2;
00079 extern PinIOJ1Array P0;   // SB700EX implementation only
00080 extern PinIOJ2Array P1;   // SB700EX implementation only
00081
00082 //typedef PINS::PinIO PinIO;
00083
00084 #endif /* _NB_PINS_H_ */
```

24.536 MODM7AE70/include/pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_PINS_H_
00006 #define _NB_PINS_H_
00007
00008 #include <pinconstant.h>
00009 #include <cpu_pins.h>
00010
00025 // class PinIO
00026 // {
00027 // uint8_t connector, pinnum;
00028 // PinIO(int j, int n)
00029 // {
00030 // connector = j;
00031 // pinnum = n;
```

```
00032 // };
00033
00034 // public:
00035 // void set(BOOL = TRUE);        // Set output high
00036 // void clr() { set(FALSE); };   // Set output low
00037 // BOOL toggle();                // Set output state to the opposite of current state
00038 // BOOL read();                  // Read hi/low state of input pin
00039 // BOOL readBack();              // Read pin state without affecting direction
00040 // void hiz() { read(); };       // Set output to tristate
00041 // void drive();                 // Turn output on (opposite of tristate)
00042 // void function(int ft);        // Set pin to special function
00043 // int getFunction();            // Get the special function the pin is set to
00044 // PinIO &operator=(BOOL b)
00045 // {
00046 // set(b);
00047 // return *this;
00048 // };
00049 // PinIO &operator=(int i)
00050 // {
00051 // set(i);
00052 // return *this;
00053 // };
00054 // operator int() { return read(); };           // Read and return int value
00055 // operator BOOL() { return read(); };          // Read and return BOOL value
00056 // operator bool() { return (read() != 0); };   // Read and return boolean value
00057 // };
00058
00059 // Pin P2_12 is multiplexed between to CPU pins (PA5 and PB5).
00060 // The multiplexed pin used in the pinns class must be defined at the top of pinconstant.h
00061
00062 class PinIOArray2
00063 {
00064    public:
00065     PinIO operator[](int n)
00066     {
00067         switch (n)
00068         {
00069             case 3: return PinIO(1, 0);
00070             case 4: return PinIO(1, 1);
00071             case 6: return PinIO(2, 12);
00072             case 7: return PinIO(3, 30);
00073             case 8: return PinIO(0, 17);
00074             case 9: return PinIO(0, 2);
00075             case 10: return PinIO(3, 18);
00076             case 11: return PinIO(1, 13);
00077 #ifdef P2_12_USE_B5
00078             case 12: return PinIO(1, 5);
00079 #else
00080             case 12: return PinIO(0, 5);
00081 #endif
00082             case 13: return PinIO(0, 8);
00083             case 15: return PinIO(3, 24);
00084             case 16: return PinIO(0, 28);
00085             case 17: return PinIO(0, 26);
00086             case 18: return PinIO(0, 27);
00087             case 19: return PinIO(0, 1);
00088             case 20: return PinIO(0, 29);
00089             case 21: return PinIO(0, 21);
00090             case 22: return PinIO(1, 4);
00091             case 23: return PinIO(3, 28);
00092             case 24: return PinIO(3, 31);
00093             case 25: return PinIO(3, 22);
00094             case 26: return PinIO(3, 27);
00095             case 27: return PinIO(3, 20);
00096             case 28: return PinIO(3, 21);
00097             case 29: return PinIO(1, 2);
00098             case 30: return PinIO(3, 12);
00099             case 31: return PinIO(0, 23);
00100             case 32: return PinIO(0, 24);
00101             case 33: return PinIO(0, 25);
00102             case 34: return PinIO(0, 9);
00103             case 35: return PinIO(0, 10);
00104             case 36: return PinIO(0, 30);
00105             case 37: return PinIO(3, 11);
00106             case 38: return PinIO(1, 3);
00107             case 39: return PinIO(0, 3);
00108             case 40: return PinIO(0, 31);
00109             case 41: return PinIO(3, 25);
00110             case 42: return PinIO(0, 4);
00111             case 43: return PinIO(0, 13);
00112             case 44: return PinIO(3, 26);
00113             case 45: return PinIO(0, 14);
00114             case 47: return PinIO(0, 12);
00115             case 48: return PinIO(0, 11);
00116             default: return PinIO();
00117         }
00118     }
```

```
00119 };
00120
00121 class PinIOArray1
00122 {
00123    public:
00124     PinIO operator[](int n)
00125     {
00126         switch (n)
00127         {
00128             case 4: return PinIO(2, 8);      // PC8
00129             case 5: return PinIO(0, 22);     // PA22
00130             case 6: return PinIO(2, 14);     // PC14
00131             case 7: return PinIO(3, 19);     // PD19
00132             case 8: return PinIO(2, 11);     // PC11
00133             case 9: return PinIO(3, 15);     // PD15
00134             case 13: return PinIO(2, 13);    // PC13
00135             case 31: return PinIO(0, 6);     // PA6
00136             case 33: return PinIO(2, 19);    // PC19
00137             case 44: return PinIO(2, 30);    // PC30
00138             case 47: return PinIO(0, 19);    // PA19
00139             default: return PinIO();
00140         }
00141     }
00142 };
00143
00144 class PinIOArrayUSB
00145 {
00146    public:
00147     PinIO operator[](int n)
00148     {
00149         switch (n)
00150         {
00151             case 1: return PinIO(2, 16);    // PC16
00152             case 3: return PinIO(0, 7);     // PA7
00153             default: return PinIO();
00154         }
00155     }
00156 };
00157
00158 extern PinIOArray2 P2;
00159 extern PinIOArray1 P1;
00160 extern PinIOArrayUSB P3;
00161
00162 #endif /* _NB_PINS_H_ */
```

24.537 NANO54415/include/pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_PINS_H_
00006 #define _NB_PINS_H_
00007
00008 #include <pinconstant.h>
00009 #include <cpu_pins.h>
00010
00011 /*
00012  * The following class is created to assist in processor pin configuration and
00013  * operation for GPIO and special functions. Each pin is defined by the system
00014  * in the '\nburn\<Platform>\system' and '\nburn\<Platform>\include' directories for
00015  * the specific platform you are using.
00016  *
00017  * Examples:
00018  *  Pins[32] = 1;                    Set pin 32 high
00019  *  Pins[32] = 0;                    Set pin 32 low
00020  *  Pins[32].hiz();                  Set pin 32 to be a high impedance input
00021  *  BOOL bpinstate = Pins[32];       Set the pin to be an input and read
00022  *  if( !Pins[32] )
00023  *     iprintf( "The pin is low" );
00024  */
00025 //class PinIO
00026 //{
00027 //    uint8_t pinnum;
00028 //    PinIO(int n) { pinnum = n; };
00029 //
00030 //   public:
00031 //    void set(BOOL = TRUE);         // Set output high
00032 //    void clr() { set(FALSE); };    // Set output low
00033 //    BOOL toggle();                 // Set output state to the opposite of current state
00034 //    BOOL read();                   // Read hi/low state of input pin
00035 //    BOOL readBack();               // Read pin state without affecting direction
00036 //    void hiz() { read(); };        // Set output to tristate
00037 //    void drive();                  // Turn output on (opposite of tristate)
00038 //    void function(int ft);         // Set pin to special function
```

```
00039 //    int getFunction();            // Get the special function the pin is set to
00040 //    PinIO &operator=(BOOL b)
00041 //    {
00042 //        set(b);
00043 //        return *this;
00044 //    };
00045 //    PinIO &operator=(int i)
00046 //    {
00047 //        set(i);
00048 //        return *this;
00049 //    };
00050 //    operator int() { return (int)read(); };     // Read and return int value
00051 //    operator BOOL() { return (BOOL)read(); };   // Read and return BOOL value
00052 //    operator bool() { return (bool)read(); };   // Read and return boolean value
00053 //    friend class PinIOArray;
00054 //};
00055
00056 class PinIOArray
00057 {
00058    public:
00059     PinIO operator[](int n); // { return PinIO(n); };
00060 };
00061
00062 extern PinIOArray Pins;
00063
00064 #endif /* _NB_PINS_H_ */
```

24.538 SB800EX/include/pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_PINS_H_
00006 #define _NB_PINS_H_
00007
00008 #include <cpu_pins.h>
00009
00010 // JP3 Header
00011 class PinIOArray
00012 {
00013    public:
00014     PinIO operator[](int n)
00015     {
00016         switch (n)
00017         {
00018             case 1: return PinIO(3, 2);
00019             case 2: return PinIO(5, 5);
00020             case 3: return PinIO(5, 6);
00021             case 4: return PinIO(5, 4);
00022             case 5: return PinIO(5, 3);
00023             case 6: return PinIO(6, 4);
00024             case 7: return PinIO(2, 3);
00025             default: return PinIO();
00026         }
00027     }
00028 };
00029
00030 extern PinIOArray Pins;
00031
00032 #endif /* _NB_PINS_H_ */
```

24.539 SBE70LC/include/pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _NB_PINS_H_
00006 #define _NB_PINS_H_
00007
00008 #include <pinconstant.h>
00009 #include <cpu_pins.h>
00010
00025 // class PinIO
00026 // {
00027 // uint8_t connector, pinnum;
00028 // PinIO(int j, int n)
00029 // {
00030 // connector = j;
00031 // pinnum = n;
00032 // };
```

```
00033
00034 // public:
00035 // void set(BOOL = TRUE);         // Set output high
00036 // void clr() { set(FALSE); };    // Set output low
00037 // BOOL toggle();                 // Set output state to the opposite of current state
00038 // BOOL read();                   // Read hi/low state of input pin
00039 // BOOL readBack();               // Read pin state without affecting direction
00040 // void hiz() { read(); };        // Set output to tristate
00041 // void drive();                  // Turn output on (opposite of tristate)
00042 // void function(int ft);         // Set pin to special function
00043 // int getFunction();             // Get the special function the pin is set to
00044 // PinIO &operator=(BOOL b)
00045 // {
00046 // set(b);
00047 // return *this;
00048 // };
00049 // PinIO &operator=(int i)
00050 // {
00051 // set(i);
00052 // return *this;
00053 // };
00054 // operator int() { return read(); };           // Read and return int value
00055 // operator BOOL() { return read(); };          // Read and return BOOL value
00056 // operator bool() { return (read() != 0); };   // Read and return boolean value
00057 // };
00058
00059 class PinIOArrayJ1
00060 {
00061    public:
00062     PinIO operator[](int n)
00063     {
00064         switch (n)
00065         {
00066             case 3: return PinIO(3, 12);    // PD12  3,12
00067             case 4: return PinIO(3, 21);    // PD21  3,21
00068             case 5: return PinIO(1, 3);     // PB3   1,3
00069             case 6: return PinIO(3, 20);    // PD20  3,20
00070             case 7: return PinIO(3, 22);    // PD22  3,22
00071             case 8: return PinIO(0, 25);    // PA25  0,25
00072             case 9: return PinIO(0, 24);    // PA24  0,24
00073             case 10: return PinIO(1, 1);    // PB1   1,1
00074             case 11: return PinIO(1, 0);    // PB0   1,0
00075             case 12: return PinIO(1, 4);    // PB4   1,4
00076             case 13: return PinIO(0, 21);   // PA21  0,21
00077             case 14: return PinIO(1, 2);    // PB2   1,2
00078             case 15: return PinIO(3, 24);   // PD24  3,24
00079             case 16: return PinIO(0, 3);    // PA3   0,3
00080             case 17: return PinIO(0, 4);    // PA4   0,4
00081             default: return PinIO();
00082         }
00083     }
00084 };
00085
00086 class PinIOArrayJ3
00087 {
00088    public:
00089     PinIO operator[](int n)
00090     {
00091         switch (n)
00092         {
00093             case 2: return PinIO(1, 13);   // PB13 1,13
00094             case 3: return PinIO(2, 12);   // PC12 2,12
00095             case 4: return PinIO(1, 4);    // PB4  1,4
00096             case 5: return PinIO(3, 30);   // PD30 3,30
00097             case 6: return PinIO(1, 5);    // PB5  1,5
00098             case 7: return PinIO(0, 17);   // PA17 0,17
00099             default: return PinIO();
00100         }
00101     }
00102 };
00103
00104 class PinIOArrayJ4
00105 {
00106    public:
00107     PinIO operator[](int n)
00108     {
00109         switch (n)
00110         {
00111             case 1: return PinIO(2, 30);   // PC30   2,30
00112             case 2: return PinIO(3, 25);   // PD25   3,25
00113             case 3: return PinIO(3, 22);   // PD22   3,22
00114             case 4: return PinIO(3, 20);   // PD20   3,20
00115             case 5: return PinIO(3, 21);   // PD21   3,21
00116             case 6: return PinIO(0, 26);   // PA26   0,26
00117             case 7: return PinIO(0, 28);   // PA28   0,28
00118             default: return PinIO();
00119         }
```

```
00120     }
00121 };
00122
00123 extern PinIOArrayJ1 J1;
00124 extern PinIOArrayJ3 J3;
00125 extern PinIOArrayJ4 J4;
00126 #define Pins J1
00127
00128 #endif /* _NB_PINS_H_ */
```

24.540 SOMRT1061/include/pins.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005
00006 #ifndef _NB_PINS_H_
00007 #define _NB_PINS_H_
00008
00009 #include <cpu_pins.h>
00010
00011 #define PLAT_PIN_COUNT 92 // 4 pins are tied to *two* cpu pads
00012
00013 class PinIOArray
00014 {
00015     static const PinIO::def_t cpuPinMap[PLAT_PIN_COUNT];
00016 public:
00017     PinIO operator[] ( int n )
00018     {
00019         if ((n <= 0) || (n > PLAT_PIN_COUNT))
00020         {
00021             return PinIO(PinIO::def_t(PinDef_INVALID).port,
00022                          PinIO::def_t(PinDef_INVALID).idx);
00023         }
00024
00025         return PinIO( cpuPinMap[n-1].port, cpuPinMap[n-1].idx );
00026     };
00027     PinIO::def_t getDef( int n )
00028     {
00029         if ((n <= 0) || (n > PLAT_PIN_COUNT))
00030         {
00031             return PinIO::def_t(PinDef_INVALID);
00032         }
00033         return cpuPinMap[n-1];
00034     };
00035 };
00036
00037
00038 extern PinIOArray Pins;
00039
00040 #endif /* _NB_PINS_H_ */
```

24.541 MOD5441X/include/plat_cfg_types.h

```
00001 #ifndef __PLAT_CFG_TYPES_H
00002 #define __PLAT_CFG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 typedef int cfg_com_t;
00008
00009 extern const cfg_com_t plat_def_com;
00010
00011 typedef config_int config_uart;
00012
00013 #endif /* ----- #ifndef __PLAT_CFG_TYPES_H  ----- */
```

24.542 MODM7AE70/include/plat_cfg_types.h

```
00001 #ifndef __PLAT_CFG_TYPES_H
00002 #define __PLAT_CFG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 typedef int cfg_com_t;
00008
```

```
00009 extern const cfg_com_t plat_def_com;
00010
00011 typedef config_int config_uart;
00012
00013 #endif /* ----- #ifndef __PLAT_CFG_TYPES_H  ----- */
```

24.543 NANO54415/include/plat_cfg_types.h

```
00001 #ifndef __PLAT_CFG_TYPES_H
00002 #define __PLAT_CFG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 typedef int cfg_com_t;
00008
00009 extern const cfg_com_t plat_def_com;
00010
00011 typedef config_int config_uart;
00012
00013 #endif /* ----- #ifndef __PLAT_CFG_TYPES_H  ----- */
```

24.544 SB800EX/include/plat_cfg_types.h

```
00001 #ifndef __PLAT_CFG_TYPES_H
00002 #define __PLAT_CFG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 typedef int cfg_com_t;
00008
00009 extern const cfg_com_t plat_def_com;
00010
00011 typedef config_int config_uart;
00012
00013 #endif /* ----- #ifndef __PLAT_CFG_TYPES_H  ----- */
```

24.545 SBE70LC/include/plat_cfg_types.h

```
00001 #ifndef __PLAT_CFG_TYPES_H
00002 #define __PLAT_CFG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 typedef int cfg_com_t;
00008
00009 extern const cfg_com_t plat_def_com;
00010
00011 typedef config_int config_uart;
00012
00013 #endif /* ----- #ifndef __PLAT_CFG_TYPES_H  ----- */
```

24.546 SOMRT1061/include/plat_cfg_types.h

```
00001 #ifndef  __PLAT_CFG_TYPES_H
00002 #define  __PLAT_CFG_TYPES_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 struct com_io_t {
00008     int port;
00009     int tx;
00010     int rx;
00011 };
00012
00013 struct ser_port_pin_opts_t {
00014     uint8_t tx[3];
00015     uint8_t rx[3];
00016 };
00017
00018 typedef com_io_t cfg_com_t;
00019
00020 extern const cfg_com_t plat_def_com;
00021 extern const ser_port_pin_opts_t plat_ser_port_pin_opts[];
```

```
00022 extern const char plat_def_com_cfg_str[];
00023
00024 #define _CFG_BUART_XSTR(_x) _CFG_BUART_STR(_x)
00025 #define _CFG_BUART_STR(_x) #_x
00026 #define _BU_CFG_OPT(port,tx,rx) "Port: " _CFG_BUART_XSTR(DEF_PLAT_COM_PORT) "- Tx: "
      _CFG_BUART_XSTR(DEF_PLAT_COM_TX_PIN) "- Rx: " _CFG_BUART_XSTR(DEF_PLAT_COM_RX_PIN)
00027
00028 class config_uart : public config_obj
00029 {
00030     void initChoices();
00031 public:
00032     class pushback_int
00033     {
00034         int v;
00035         config_uart &parent;
00036
00037     public:
00038         pushback_int(config_uart &p) : v(0), parent(p) {};
00039         pushback_int(config_uart &p, int val) : v(val), parent(p) {};
00040         pushback_int(config_uart &p, const pushback_int &rhs) : v(rhs.v), parent(p) {};
00041         pushback_int(const pushback_int &rhs) : v(rhs.v), parent(rhs.parent) {};
00042
00043         operator int() { return v; }
00044
00045
00046         const pushback_int &operator=(int i) { v = i; return *this;}
00047         const pushback_int &operator=(pushback_int &rhs) { v = rhs.v; return *this;}
00048     };
00049     config_chooser portCfg{"Configuration", plat_def_com_cfg_str, ""};
00050     ConfigEndMarker;
00051     pushback_int portNum{*this};
00052     pushback_int pinTx{*this};
00053     pushback_int pinRx{*this};
00054
00055     config_uart(config_obj &owner, cfg_com_t &com_cfg, const char* name, const char *desc = NULL);
00056     config_uart(const cfg_com_t &com_cfg, const char* name, const char *desc = NULL);
00057
00058     virtual void CommitTestedValue(uint32_t permission_mask) override;
00059     operator uint32_t() { return portNum; }
00060     config_uart &operator=(const uint32_t i){portNum=i; return *this;};
00061 };
00062
00063 #endif   /* ----- #ifndef __PLAT_CFG_TYPES_H  ----- */
```

24.547 MOD5441X/include/wifi/nbWifiDefs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __NBWIFIDEFS_H
00006 #define __NBWIFIDEFS_H
00007
00008 #define NBWIFI_DEFAULT_RESETPIN 42   // Reset pin number
00009 #define NBWIFI_DEFAULT_IRQNUM 3      // IRQ number
00010 #define NBWIFI_DEFAULT_SPINUM 1      // SPI channel
00011 #define NBWIFI_DEFAULT_CSNUM 0       // SPI chip select number
00012 #define NBWIFI_DEFAULT_CONNUM 2      // Module physical connector number
00013 #define NBWIFI_DEFAULT_PINNUM -1     // GPIO pin used for SPI chip select
00014 #define NBWIFI_DEFAULT_UART 9        // UART number if using serial interface
00015 #define NBWIFI_DEFAULT_CHIPEN 34     // GPIO used for chip enable, only for the NBWIFIWILC
00016
00017 #define NBWIFI_DEFAULT_WIFICHANNEL 6
00018
00019 #endif /* ----- #ifndef __NBWIFIDEFS_H  ----- */
```

24.548 MODM7AE70/include/wifi/nbWifiDefs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __NBWIFIDEFS_H
00006 #define __NBWIFIDEFS_H
00007
00008 #define NBWIFI_DEFAULT_RESETPIN 42   // Reset pin number
00009 #define NBWIFI_DEFAULT_IRQNUM 4      // IRQ number
00010 #define NBWIFI_DEFAULT_SPINUM 0      // SPI channel
00011 #define NBWIFI_DEFAULT_CSNUM 2       // SPI chip select number
00012 #define NBWIFI_DEFAULT_CONNUM 2      // Module physical connector number
00013 #define NBWIFI_DEFAULT_PINNUM -1     // GPIO pin used for SPI chip select
00014 #define NBWIFI_DEFAULT_UART 9        // UART number if using serial interface
```

```
00015 #define NBWIFI_DEFAULT_CHIPEN 34     // GPIO used for chip enable, only for the NBWIFIWILC
00016
00017 #define NBWIFI_DEFAULT_WIFICHANNEL 6
00018
00019 #endif /* ----- #ifndef __NBWIFIDEFS_H  ----- */
```

24.549 NANO54415/include/wifi/nbWifiDefs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __NBWIFIDEFS_H
00006 #define __NBWIFIDEFS_H
00007
00008 #define NBWIFI_DEFAULT_RESETPIN 25   // Reset pin number
00009 #define NBWIFI_DEFAULT_IRQNUM 3      // IRQ number
00010 #define NBWIFI_DEFAULT_SPINUM 1      // SPI channel
00011 #define NBWIFI_DEFAULT_CSNUM 0       // SPI chip select number
00012 #define NBWIFI_DEFAULT_CONNUM -1     // Module physical connector number
00013 #define NBWIFI_DEFAULT_PINNUM -1     // GPIO pin used for SPI chip select
00014 #define NBWIFI_DEFAULT_UART 9        // UART number if using serial interface
00015 #define NBWIFI_DEFAULT_CHIPEN 34     // GPIO used for chip enable, only for the NBWIFIWILC
00016
00017 #define NBWIFI_DEFAULT_WIFICHANNEL 6
00018
00019 #endif /* ----- #ifndef __NBWIFIDEFS_H  ----- */
```

24.550 SB800EX/include/wifi/nbWifiDefs.h

```
00001 #ifndef __NBWIFIDEFS_H
00002 #define __NBWIFIDEFS_H
00003
00004 #define NBWIFI_DEFAULT_RESETPIN 6    // Reset pin number
00005 #define NBWIFI_DEFAULT_RESETCONN 2   // Reset connector number
00006
00007 #define NBWIFI_DEFAULT_IRQNUM 3   // IRQ number
00008 #define NBWIFI_IRQ_PIN -1         // Not applicable since SB800EX has no header/connector
00009
00010 #define NBWIFI_DEFAULT_SPINUM 2    // SPI channel
00011 #define NBWIFI_DEFAULT_CSNUM 0     // SPI chip select number
00012 #define NBWIFI_DEFAULT_CONNUM 1    // Module physical connector number
00013 #define NBWIFI_DEFAULT_PINNUM -1   // GPIO pin used for SPI chip select
00014 #define NBWIFI_DEFAULT_UART 0      // UART number if using serial interface
00015 #define NBWIFI_DEFAULT_CHIPEN 1    // GPIO used for chip enable, only for the NBWIFIWILC
00016
00017 #define NBWIFI_DEFAULT_WIFICHANNEL 6
00018
00019 #endif /* ----- #ifndef __NBWIFIDEFS_H  ----- */
```

24.551 SBE70LC/include/wifi/nbWifiDefs.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef __NBWIFIDEFS_H
00006 #define __NBWIFIDEFS_H
00007
00008 #define NBWIFI_DEFAULT_RESETPIN 6   // Reset pin number
00009 #define NBWIFI_DEFAULT_IRQNUM 4     // IRQ number
00010 #define NBWIFI_DEFAULT_SPINUM 0     // SPI channel
00011 #define NBWIFI_DEFAULT_CSNUM 1      // SPI chip select number
00012 #define NBWIFI_DEFAULT_CONNUM 4     // Module physical connector number
00013 #define NBWIFI_DEFAULT_PINNUM -1    // GPIO pin used for SPI chip select
00014 #define NBWIFI_DEFAULT_UART 9       // UART number if using serial interface
00015 #define NBWIFI_DEFAULT_CHIPEN 34    // GPIO used for chip enable, only for the NBWIFIWILC
00016
00017 #define NBWIFI_DEFAULT_WIFICHANNEL 6
00018
00019 #endif /* ----- #ifndef __NBWIFIDEFS_H  ----- */
```

24.552 MODM7AE70/include/config_obj_platdefs.h

```
00001 #ifndef __CONFIG_OBJ_PLATDEFS_H
00002 #define __CONFIG_OBJ_PLATDEFS_H
00003
```

```
00004 #define DEF_PLAT_BAUD 115200
00005 #define DEF_PLAT_COM 0
00006 #define DEF_PLAT_DLY 2
00007 #define DEF_PLAT_QUIET false
00008 #define DEF_PLAT_WATCHDOG_ENABLED false
00009
00010 #endif /* ----- #ifndef __CONFIG_OBJ_PLATDEFS_H  ----- */
```

24.553 SBE70LC/include/config_obj_platdefs.h

```
00001 #ifndef __CONFIG_OBJ_PLATDEFS_H
00002 #define __CONFIG_OBJ_PLATDEFS_H
00003
00004 #define DEF_PLAT_BAUD 115200
00005 #define DEF_PLAT_COM 0
00006 #define DEF_PLAT_DLY 2
00007 #define DEF_PLAT_QUIET false
00008 #define DEF_PLAT_WATCHDOG_ENABLED false
00009
00010 #endif /* ----- #ifndef __CONFIG_OBJ_PLATDEFS_H  ----- */
```

24.554 SOMRT1061/include/config_obj_platdefs.h

```
00001 #ifndef  __CONFIG_OBJ_PLATDEFS_H
00002 #define  __CONFIG_OBJ_PLATDEFS_H
00003
00004 #define DEF_PLAT_BAUD   115200
00005 #define DEF_PLAT_COM_PORT 0
00006 #define DEF_PLAT_COM_TX_PIN 8
00007 #define DEF_PLAT_COM_RX_PIN 7
00008 #define DEF_PLAT_COM    com_io_t({DEF_PLAT_COM_PORT, DEF_PLAT_COM_TX_PIN, DEF_PLAT_COM_RX_PIN})
00009 #define DEF_PLAT_DLY    2
00010 #define DEF_PLAT_QUIET  false
00011
00012 #endif   /* ----- #ifndef __CONFIG_OBJ_PLATDEFS_H  ----- */
```

24.555 MODM7AE70/include/serial_platdefs.h

```
00001 #ifndef __SERIAL_PLATDEFS_H
00002 #define __SERIAL_PLATDEFS_H
00003 #include <same70_serial.h>
00004
00005 #define NUM_USARTS 2
00006 #define NUM_UARTS 5
00007 #define UART_PORT_OFFSET (NUM_USARTS)
00008 #define NUM_SER_PORTS (NUM_USARTS + NUM_UARTS)
00009
00010 #define USART_MAP       \
00011     {                   \
00012         USART0, USART1 \
00013     }
00014 #define UART_MAP                            \
00015     {                                       \
00016         UART0, UART1, UART2, UART3, UART4 \
00017     }
00018 #define SER_IRQ_ID_MAP                                                                          \
00019     {                                                                                           \
00020         USART0_IRQn, USART1_IRQn, UART0_IRQn, UART1_IRQn, UART2_IRQn, UART3_IRQn, UART4_IRQn \
00021     }
00022 #define SER_IO_CONF_MAP
    \
00023     {
    \
00024         {{1, 1, configMap::MODE_C, 1}, {1, 0, configMap::MODE_C, 0}}, {{1, 4, configMap::MODE_D, 1},
    {0, 21, configMap::MODE_A, 0}}, \
00025
    \
00026             {{0, 10, configMap::MODE_A, 1}, {0, 9, configMap::MODE_A, 0}},
    \
00027             {{0, 0, configMap::MODE_A, SER_IO_NO_CONF}, {0, 5, configMap::MODE_C, 0}},
    \
00028             {{3, 26, configMap::MODE_C, 1}, {3, 25, configMap::MODE_C, 0}},
    \
00029             {{0, 0, configMap::MODE_A, SER_IO_NO_CONF}, {3, 28, configMap::MODE_A, 0}},
    \
00030             {{3, 19, configMap::MODE_C, 1}, {3, 18, configMap::MODE_C, 0}},
    \
00031     }
00032
```

```
00033 #endif /* ----- #ifndef __SERIAL_PLATDEFS_H  ----- */
```

24.556 SBE70LC/include/serial_platdefs.h

```
00001 #ifndef __SERIAL_PLATDEFS_H
00002 #define __SERIAL_PLATDEFS_H
00003 #include <same70_serial.h>
00004
00005 #define NUM_USARTS 2
00006 #define NUM_UARTS 5
00007 #define UART_PORT_OFFSET (NUM_USARTS)
00008 #define NUM_SER_PORTS (NUM_USARTS + NUM_UARTS)
00009
00010 #define USART_MAP       \
00011     {                   \
00012         USART0, USART1 \
00013     }
00014 #define UART_MAP                            \
00015     {                                       \
00016         UART0, UART1, UART2, UART3, UART4 \
00017     }
00018 #define SER_IRQ_ID_MAP                                                                          \
00019     {                                                                                           \
00020         USART0_IRQn, USART1_IRQn, UART0_IRQn, UART1_IRQn, UART2_IRQn, UART3_IRQn, UART4_IRQn \
00021     }
00022 #define SER_IO_CONF_MAP
    \
00023     {
    \
00024         {{1, 1, configMap::MODE_C, 1}, {1, 0, configMap::MODE_C, 0}}, {{1, 4, configMap::MODE_D, 1},
    {0, 21, configMap::MODE_A, 0}}, \
00025
    \
00026             {{0, 10, configMap::MODE_A, 1}, {0, 9, configMap::MODE_A, 0}},
    \
00027             {{0, 0, configMap::MODE_A, SER_IO_NO_CONF}, {0, 5, configMap::MODE_C, 0}},
    \
00028             {{3, 26, configMap::MODE_C, 1}, {3, 25, configMap::MODE_C, 0}},
    \
00029             {{0, 0, configMap::MODE_A, SER_IO_NO_CONF}, {3, 28, configMap::MODE_A, 0}},
    \
00030             {{3, 19, configMap::MODE_C, 1}, {3, 18, configMap::MODE_C, 0}},
    \
00031     }
00032
00033 #endif /* ----- #ifndef __SERIAL_PLATDEFS_H  ----- */
```

24.557 SOMRT1061/include/serial_platdefs.h

```
00001 #ifndef  __SERIAL_PLATDEFS_H
00002 #define  __SERIAL_PLATDEFS_H
00003
00004 #define NUM_UARTS   7
00005
00006 #define NUM_SER_PORTS NUM_UARTS
00007
00008 // serial port macros for openserial portnum parameter
00009 #define SER_PORT_LPUART3   0
00010 #define SER_PORT_LPUART2   1
00011 #define SER_PORT_LPUART4   2
00012 #define SER_PORT_LPUART5   3
00013 #define SER_PORT_LPUART6   4
00014 #define SER_PORT_LPUART7   5
00015 #define SER_PORT_LPUART8   6
00016
00017 #define UART_MAP    {LPUART3, LPUART2, LPUART4, LPUART5, LPUART6, LPUART7, LPUART8}
00018
00019 #define SER_IRQ_ID_MAP  { LPUART3_IRQn, LPUART2_IRQn,\
00020     LPUART4_IRQn, LPUART5_IRQn, LPUART6_IRQn, LPUART7_IRQn, LPUART8_IRQn }
00021
00022 #define UART_CLOCKS \
00023 { \
00024     kCLOCK_Lpuart3, kCLOCK_Lpuart2, kCLOCK_Lpuart4, \
00025     kCLOCK_Lpuart5, kCLOCK_Lpuart6, kCLOCK_Lpuart7, kCLOCK_Lpuart8 \
00026 }
00027
00028 #define SER_PORT_PIN_OPTS \
00029 { \
00030     {{ 8,  46,0xFF},{ 7,  47,0xFF}}, \
00031     {{42,0xFF,0xFF},{43,0xFF,0xFF}}, \
00032     {{56,  88,0xFF},{58,  87,0xFF}}, \
00033     {{86,0xFF,0xFF},{85,0xFF,0xFF}}, \
```

```
00034     {{84,0xFF,0xFF},{83,0xFF,0xFF}}, \
00035     {{77,0xFF,0xFF},{76,0xFF,0xFF}}, \
00036     {{29,  70,0xFF},{28,  69,0xFF}} \
00037 }
00038
00039 #endif   /* ----- #ifndef __SERIAL_PLATDEFS_H  ----- */
```

24.558 NANO54415/include/esdhc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _ESDHC_H
00006 #define _ESDHC_H
00007
00008 #include <predef.h>
00009 #include <stdio.h>
00010 #include <init.h>
00011 #include <sim.h> /*on-chip register definitions*/
00012 #include <pins.h>
00013 #include <nbrtos.h>
00014
00015 #include <sim5441x.h>
00016 #include "effs_fat/sdhc_mcf.h"
00017
00018 #ifndef __cplusplus
00019 #error eSDHC driver is a C++ only library
00020 #endif
00021
00022 /*
00023    ESDHC state
00024 */
00025 #define ESDHC_OK (0)
00026 #define ESDHC_BUSY (1)
00027 #define ESDHC_ERROR (2)
00028 #define ESDHC_TIMEOUT (3)
00029
00030 //#define __DEBUG_ESDHC // Enable debug messages
00031
00032 #define DMA_MODE_SIMPLE 1
00033 //#define DMA_MODE_ADMA2  1
00034 #define ESDHC_IRQ_MODE 1
00035
00036 /*
00037  *****************************************************************************-
00038  *
00039  *   dspiDriverStruct
00040  *
00041  *   This struct contains the major variables/configurations used for a eSDHC transfer
00042  *
00043  *
00044  *   OS_SEM* ESDHC_Sem - This is a pointer to an external semaphore provided by ()
00045  *
00046  *   uint8_t ESDHC_INT_STATUS - Status of the spi device
00047  *
00048  *
00049  *****************************************************************************-
00050  */
00051 typedef struct
00052 {
00053     void *pBlockData;
00054     uint16_t BlockSize;
00055     uint32_t BlocksCount;
00056     uint32_t CmdMask;
00057     uint32_t CmdArg;
00058     OS_SEM *finishedSem;
00059     void *pResp;
00060     volatile uint8_t ESDHC_INT_STATUS;
00061     volatile BOOL ESDHCfinished;
00062 } esdhcDriverStruct;
00063
00064 class ESDHCModule
00065 {
00066     OS_CRIT m_critSec;
00067     OS_SEM *m_finishedSem;
00068     uint32_t m_actualBaudrate;
00069
00070    public:
00071     // static  ESDHCModule *lastCxts;
00072     static esdhcDriverStruct tranCtx;
00073     static bool m_inProgress;
00074
00075    public:
00076     ESDHCModule();
```

```
00077     ~ESDHCModule();
00078
00079     uint8_t Init(uint32_t Baudrate = 0, bool hw_reset = false);
00080     void Reset(uint32_t Baudrate = INIT_BAUDRATE, bool hw_reset = true);
00081     bool SetBaudrate(uint32_t Baudrate);
00082     bool SetDataBusWidth(uint8_t width);
00083     uint8_t GetDataBusWidth(void);
00084     void ClearTransferStatus(bool abortTransfer = true);
00085
00086     void SetTimeouts(double rdTimeout_us, double wrTimeout_us);
00087
00088     bool RegisterSem(OS_SEM *finishedSem);
00089     inline bool ClrSem() { return RegisterSem(NULL); }
00090     inline OS_SEM *GetSem() { return m_finishedSem; }
00091
00092     uint8_t TransferCmd(uint8_t cmdIdx, uint32_t cmdArg = 0, void *cmdRsp = NULL);
00093     uint8_t TransferCmdData(uint8_t cmdIdx,
00094                             uint32_t cmdArg,
00095                             bool dataRead,
00096                             void *blkData,
00097                             uint16_t blk_size,
00098                             uint16_t blkCount = 1,
00099                             void *cmdRsp = NULL,
00100                             bool autoCMD12 = true);
00101     uint8_t GetTransferStatus();
00102     bool AbortTransfer(uint32_t timeout, bool force = false);
00103
00104     inline bool Done() { return !m_inProgress; }
00105
00106     inline uint32_t GetActualBaudrate() { return m_actualBaudrate; }
00107
00108     bool CardReady();
00109
00110 #ifndef ESDHC_IRQ_MODE
00111     static uint8_t send_cmd(unsigned long cmd_index, unsigned long cmd_arg = 0, void *resp_data =
      NULL);
00112     static uint8_t send_cmd_dt(unsigned long cmd_index,
00113                                unsigned long cmd_arg,
00114                                void *data,
00115                                unsigned short data_size,
00116                                void *resp_data = NULL);
00117     // static uint8_t read_block(int drv, unsigned long blk_addr, unsigned short blk_size, void
      *buff);
00118     // static uint8_t write_block(int drv, unsigned long blk_addr, unsigned short blk_size, void
      *buff);
00119 #endif
00120
00121   private:
00122     bool TransferDone();
00123
00124     uint32_t mRdDTOCV;
00125     uint32_t mWrDTOCV;
00126 };
00127
00128 #endif /* _ESDHC_H_ */
```

24.559 SB800EX/include/esdhc.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef _ESDHC_H
00006 #define _ESDHC_H
00007
00008 #include <predef.h>
00009 #include <stdio.h>
00010 #include <init.h>
00011 #include <sim.h> /*on-chip register definitions*/
00012 #include <pins.h>
00013 #include <nbrtos.h>
00014
00015 #include <sim5441x.h>
00016 #include "effs_fat/sdhc_mcf.h"
00017
00018 #ifndef __cplusplus
00019 #error eSDHC driver is a C++ only library
00020 #endif
00021
00022 /*
00023    ESDHC state
00024 */
00025 #define ESDHC_OK (0)
00026 #define ESDHC_BUSY (1)
00027 #define ESDHC_ERROR (2)
```

```
00028 #define ESDHC_TIMEOUT (3)
00029
00030 //#define __DEBUG_ESDHC // Enable debug messages
00031
00032 #define DMA_MODE_SIMPLE 1
00033 //#define DMA_MODE_ADMA2  1
00034 #define ESDHC_IRQ_MODE 1
00035
00036 /*
00037  *****************************************************************************-
00038  *
00039  *   dspiDriverStruct
00040  *
00041  *   This struct contains the major variables/configurations used for a eSDHC transfer
00042  *
00043  *
00044  *   OS_SEM* ESDHC_Sem - This is a pointer to an external semaphore provided by ()
00045  *
00046  *   uint8_t ESDHC_INT_STATUS - Status of the spi device
00047  *
00048  *
00049  *****************************************************************************-
00050  */
00051 typedef struct
00052 {
00053     void *pBlockData;
00054     uint16_t BlockSize;
00055     uint32_t BlocksCount;
00056     uint32_t CmdMask;
00057     uint32_t CmdArg;
00058     OS_SEM *finishedSem;
00059     void *pResp;
00060     volatile uint8_t ESDHC_INT_STATUS;
00061     volatile BOOL ESDHCfinished;
00062 } esdhcDriverStruct;
00063
00064 class ESDHCModule
00065 {
00066     OS_CRIT m_critSec;
00067     OS_SEM *m_finishedSem;
00068     uint32_t m_actualBaudrate;
00069
00070    public:
00071     // static  ESDHCModule *lastCxts;
00072     static esdhcDriverStruct tranCtx;
00073     static bool m_inProgress;
00074
00075    public:
00076     ESDHCModule();
00077     ~ESDHCModule();
00078
00079     uint8_t Init(uint32_t Baudrate = 0, bool hw_reset = false);
00080     void Reset(uint32_t Baudrate = INIT_BAUDRATE, bool hw_reset = true);
00081     bool SetBaudrate(uint32_t Baudrate);
00082     bool SetDataBusWidth(uint8_t width);
00083     uint8_t GetDataBusWidth(void);
00084     void ClearTransferStatus(bool abortTransfer = true);
00085
00086     void SetTimeouts(double rdTimeout_us, double wrTimeout_us);
00087
00088     bool RegisterSem(OS_SEM *finishedSem);
00089     inline bool ClrSem() { return RegisterSem(NULL); }
00090     inline OS_SEM *GetSem() { return m_finishedSem; }
00091
00092     uint8_t TransferCmd(uint8_t cmdIdx, uint32_t cmdArg = 0, void *cmdRsp = NULL);
00093     uint8_t TransferCmdData(uint8_t cmdIdx,
00094                             uint32_t cmdArg,
00095                             bool dataRead,
00096                             void *blkData,
00097                             uint16_t blk_size,
00098                             uint16_t blkCount = 1,
00099                             void *cmdRsp = NULL,
00100                             bool autoCMD12 = true);
00101     uint8_t GetTransferStatus();
00102     bool AbortTransfer(uint32_t timeout, bool force = false);
00103
00104     inline bool Done() { return !m_inProgress; }
00105
00106     inline uint32_t GetActualBaudrate() { return m_actualBaudrate; }
00107
00108     bool CardReady();
00109
00110 #ifndef ESDHC_IRQ_MODE
00111     static uint8_t send_cmd(unsigned long cmd_index, unsigned long cmd_arg = 0, void *resp_data =
      NULL);
00112     static uint8_t send_cmd_dt(unsigned long cmd_index,
00113                                unsigned long cmd_arg,
```

```
00114                                 void *data,
00115                                 unsigned short data_size,
00116                                 void *resp_data = NULL);
00117     // static uint8_t read_block(int drv, unsigned long blk_addr, unsigned short blk_size, void
      *buff);
00118     // static uint8_t write_block(int drv, unsigned long blk_addr, unsigned short blk_size, void
      *buff);
00119 #endif
00120
00121    private:
00122     bool TransferDone();
00123
00124     uint32_t mRdDTOCV;
00125     uint32_t mWrDTOCV;
00126 };
00127
00128 #endif /* _ESDHC_H_ */
```

24.560 NanoMemConstants.h

```
00001 #define JSON_CONFIG_SIZE (0x10000)
00002 extern uint8_t gConfigRam[JSON_CONFIG_SIZE];
```

24.561 NANO54415/include/NanoStdFileSupport.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <basictypes.h>
00006
00007 #ifdef __cplusplus
00008 extern "C"
00009 {
00010 #endif
00011
00012     int NanoRawEraseFlash(long block, int siz);
00013     int NanoRawWriteFlash(void *data, long pos, long size);
00014     void NanoStdFlashInit(puint8_t Buffer, int len, uint32_t FirstAddrInFlash);
00015
00016 #ifdef __cplusplus
00017 };
00018 #endif
```

24.562 SB800EX/include/NanoStdFileSupport.h

```
00001 #include <basictypes.h>
00002
00003 #ifdef __cplusplus
00004 extern "C"
00005 {
00006 #endif
00007
00008     int NanoRawEraseFlash(long block, int siz);
00009     int NanoRawWriteFlash(void *data, long pos, long size);
00010     void NanoStdFlashInit(puint8_t Buffer, int len, uint32_t FirstAddrInFlash);
00011
00012 #ifdef __cplusplus
00013 };
00014 #endif
```

24.563 NANO54415/include/SPIFlash.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 class SPIFlash
00006 {
00007    private:
00008     uint32_t PCS_MASK;
00009     void SendSPITX(uint8_t b, int end = 0);
00010     uint8_t DoSpiRx();
00011     void FlushSPIRX();
00012     void SpiStart();
00013     void SpiFrameWait();
00014     void SpiAscCmd(const char *cp);
```

```
00015     void QFlushSPIRX();
00016     uint8_t ReadSR();
00017     bool WriteSR(uint8_t bv);
00018
00019    public:
00020     SPIFlash(int csnum = 0);
00021     ~SPIFlash();
00022     void WriteEnable(bool enable);
00023     void ReadData(puint8_t dest, int len, int addr);
00024     bool WriteData(puint8_t dada, int len, int addr);
00025     bool WriteDataWErase(puint8_t data, int len, int addr);
00026     bool EraseChip();
00027     bool EraseSector(int addr);
00028     bool EraseRange(int addr, int len);
00029     bool WriteEnabled();
00030     void WriteProtect(int b);
00031     void ResetMe();
00032     uint32_t JedecId();
00033     uint32_t UniqueIdL();
00034     uint32_t UniqueIdM();
00035 };
```

24.564 SB800EX/include/SPIFlash.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 class SPIFlash
00006 {
00007    private:
00008     uint32_t PCS_MASK;
00009     void SendSPITX(uint8_t b, int end = 0);
00010     uint8_t DoSpiRx();
00011     void FlushSPIRX();
00012     void SpiStart();
00013     void SpiFrameWait();
00014     void SpiAscCmd(const char *cp);
00015     void QFlushSPIRX();
00016     uint8_t ReadSR();
00017     bool WriteSR(uint8_t bv);
00018
00019    public:
00020     SPIFlash(int csnum = 0);
00021     ~SPIFlash();
00022     void WriteEnable(bool enable);
00023     void ReadData(puint8_t dest, int len, int addr);
00024     bool WriteData(puint8_t dada, int len, int addr);
00025     bool WriteDataWErase(puint8_t data, int len, int addr);
00026     bool EraseChip();
00027     bool EraseSector(int addr);
00028     bool EraseRange(int addr, int len);
00029     bool WriteEnabled();
00030     void WriteProtect(int b);
00031     void ResetMe();
00032     uint32_t JedecId();
00033     uint32_t UniqueIdL();
00034     uint32_t UniqueIdM();
00035 };
```

24.565 LED_functions.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #include <cpu_pins.h>
00006 #include <pinconstant.h>
00007 #include <basictypes.h>
00008
00009 #define LED1_1 PortI[5]
00010 #define LED1_2 PortI[7]
00011
00012 #define LED2_1 PortH[0]
00013 #define LED2_2 PortI[4]
00014
00015 #define LED_OFF (0)
00016 #define LED_COLOR_RED (1)
00017 #define LED_COLOR_GREEN (2)
00018
00019 void setLEDColor(uint8_t number, uint8_t color);
00020 void turnLEDsOff();
```

24.566 SB800EXMemConstants.h

```
00001 #define JSON_CONFIG_SIZE (0x10000)
00002 extern uint8_t gConfigRam[JSON_CONFIG_SIZE];
```

24.567 serial_config.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004
00005 #ifndef SB800EX_SERIAL_CONFIG
00006 #define SB800EX_SERIAL_CONFIG
00007
00008 // Serial level shifter configuration options
00009 #define SERIAL_XCVR_LOOPBACK 0
00010 #define SERIAL_XCVR_RS232 2
00011 #define SERIAL_XCVR_RS485HALF 1
00012 #define SERIAL_XCVR_RS485FULL 3
00013
00014 int32_t SetSerialMode(uint32_t uartNum, uint8_t mode);
00015 int32_t GetSerialMode(uint32_t uartNum);
00016 void Set485TxEnable(uint32_t uartNumber, bool enable);
00017
00018 #endif
```

24.568 decomp.h

```
00001 #ifndef  __DECOMP_H
00002 #define  __DECOMP_H
00003
00004 //#include <startup.h>
00005
00006 void LoadApplication(AppHeader_t *AppHeader);
00007 #endif   /* ----- #ifndef __DECOMP_H  ----- */
```

24.569 evkmimxrt1060_flexspi_nor_config.h

```
00001 /*
00002  * Copyright 2018 NXP
00003  * All rights reserved.
00004  *
00005  * SPDX-License-Identifier: BSD-3-Clause
00006  */
00007
00008 #ifndef __EVKMIMXRT1060_FLEXSPI_NOR_CONFIG__
00009 #define __EVKMIMXRT1060_FLEXSPI_NOR_CONFIG__
00010
00011 #include <stdint.h>
00012 #include <stdbool.h>
00013 #include <sim.h>
00014
00018 #define FSL_XIP_BOARD_DRIVER_VERSION (MAKE_VERSION(2, 0, 0))
00021 /* FLEXSPI memory config block related defintions */
00022 #define FLEXSPI_CFG_BLK_TAG (0x42464346UL)     // ascii "FCFB" Big Endian
00023 #define FLEXSPI_CFG_BLK_VERSION (0x56010400UL) // V1.4.0
00024 #define FLEXSPI_CFG_BLK_SIZE (512)
00025
00026 /* FLEXSPI Feature related definitions */
00027 #define FLEXSPI_FEATURE_HAS_PARALLEL_MODE 1
00028
00029 /* Lookup table related defintions */
00030 #define CMD_INDEX_READ 0
00031 #define CMD_INDEX_READSTATUS 1
00032 #define CMD_INDEX_WRITEENABLE 2
00033 #define CMD_INDEX_WRITE 4
00034
00035 #define CMD_LUT_SEQ_IDX_READ 0
00036 #define CMD_LUT_SEQ_IDX_READSTATUS 1
00037 #define CMD_LUT_SEQ_IDX_WRITEENABLE 3
00038 #define CMD_LUT_SEQ_IDX_WRITE 9
00039
00040 #define CMD_SDR 0x01
00041 #define CMD_DDR 0x21
00042 #define RADDR_SDR 0x02
00043 #define RADDR_DDR 0x22
00044 #define CADDR_SDR 0x03
00045 #define CADDR_DDR 0x23
00046 #define MODE1_SDR 0x04
```

```
00047 #define MODE1_DDR 0x24
00048 #define MODE2_SDR 0x05
00049 #define MODE2_DDR 0x25
00050 #define MODE4_SDR 0x06
00051 #define MODE4_DDR 0x26
00052 #define MODE8_SDR 0x07
00053 #define MODE8_DDR 0x27
00054 #define WRITE_SDR 0x08
00055 #define WRITE_DDR 0x28
00056 #define READ_SDR 0x09
00057 #define READ_DDR 0x29
00058 #define LEARN_SDR 0x0A
00059 #define LEARN_DDR 0x2A
00060 #define DATSZ_SDR 0x0B
00061 #define DATSZ_DDR 0x2B
00062 #define DUMMY_SDR 0x0C
00063 #define DUMMY_DDR 0x2C
00064 #define DUMMY_RWDS_SDR 0x0D
00065 #define DUMMY_RWDS_DDR 0x2D
00066 #define JMP_ON_CS 0x1F
00067 #define STOP 0
00068
00069 #define FLEXSPI_1PAD 0
00070 #define FLEXSPI_2PAD 1
00071 #define FLEXSPI_4PAD 2
00072 #define FLEXSPI_8PAD 3
00073
00074 #define FLEXSPI_LUT_SEQ(cmd0, pad0, op0, cmd1, pad1, op1)
      \
00075     (FLEXSPI_LUT_OPERAND0(op0) | FLEXSPI_LUT_NUM_PADS0(pad0) | FLEXSPI_LUT_OPCODE0(cmd0) |
     FLEXSPI_LUT_OPERAND1(op1) | \
00076      FLEXSPI_LUT_NUM_PADS1(pad1) | FLEXSPI_LUT_OPCODE1(cmd1))
00077
00079 typedef enum _FlexSpiSerialClockFreq
00080 {
00081     kFlexSpiSerialClk_30MHz  = 1,
00082     kFlexSpiSerialClk_50MHz  = 2,
00083     kFlexSpiSerialClk_60MHz  = 3,
00084     kFlexSpiSerialClk_75MHz  = 4,
00085     kFlexSpiSerialClk_80MHz  = 5,
00086     kFlexSpiSerialClk_100MHz = 6,
00087     kFlexSpiSerialClk_120MHz = 7,
00088     kFlexSpiSerialClk_133MHz = 8,
00089     kFlexSpiSerialClk_166MHz = 9,
00090 } flexspi_serial_clk_freq_t;
00091
00093 enum
00094 {
00095     kFlexSpiClk_SDR,
00096     kFlexSpiClk_DDR,
00097 };
00098
00100 typedef enum _FlashReadSampleClkSource
00101 {
00102     kFlexSPIReadSampleClk_LoopbackInternally      = 0,
00103     kFlexSPIReadSampleClk_LoopbackFromDqsPad      = 1,
00104     kFlexSPIReadSampleClk_LoopbackFromSckPad      = 2,
00105     kFlexSPIReadSampleClk_ExternalInputFromDqsPad = 3,
00106 } flexspi_read_sample_clk_t;
00107
00109 enum
00110 {
00111     kFlexSpiMiscOffset_DiffClkEnable            = 0,
00112     kFlexSpiMiscOffset_Ck2Enable                = 1,
00113     kFlexSpiMiscOffset_ParallelEnable           = 2,
00114     kFlexSpiMiscOffset_WordAddressableEnable    = 3,
00115     kFlexSpiMiscOffset_SafeConfigFreqEnable     = 4,
00116     kFlexSpiMiscOffset_PadSettingOverrideEnable = 5,
00117     kFlexSpiMiscOffset_DdrModeEnable            = 6,
00118 };
00119
00121 enum
00122 {
00123     kFlexSpiDeviceType_SerialNOR    = 1,
00124     kFlexSpiDeviceType_SerialNAND   = 2,
00125     kFlexSpiDeviceType_SerialRAM    = 3,
00126     kFlexSpiDeviceType_MCP_NOR_NAND = 0x12,
00127     kFlexSpiDeviceType_MCP_NOR_RAM  = 0x13,
00128 };
00129
00131 enum
00132 {
00133     kSerialFlash_1Pad  = 1,
00134     kSerialFlash_2Pads = 2,
00135     kSerialFlash_4Pads = 4,
00136     kSerialFlash_8Pads = 8,
00137 };
```

```
00138
00140 typedef struct _lut_sequence
00141 {
00142     uint8_t seqNum;
00143     uint8_t seqId;
00144     uint16_t reserved;
00145 } flexspi_lut_seq_t;
00146
00148 enum
00149 {
00150     kDeviceConfigCmdType_Generic,
00151     kDeviceConfigCmdType_QuadEnable,
00152     kDeviceConfigCmdType_Spi2Xpi,
00153     kDeviceConfigCmdType_Xpi2Spi,
00154     kDeviceConfigCmdType_Spi2NoCmd,
00155     kDeviceConfigCmdType_Reset,
00156 };
00157
00159 typedef struct _FlexSPIConfig
00160 {
00161     uint32_t tag;
00162     uint32_t version;
00163     uint32_t reserved0;
00164     uint8_t readSampleClkSrc;
00165     uint8_t csHoldTime;
00166     uint8_t csSetupTime;
00167     uint8_t columnAddressWidth;
00169     uint8_t deviceModeCfgEnable;
00170     uint8_t deviceModeType;
00172     uint16_t waitTimeCfgCommands;
00174     flexspi_lut_seq_t deviceModeSeq;
00176     uint32_t deviceModeArg;
00177     uint8_t configCmdEnable;
00178     uint8_t configModeType[3];
00179     flexspi_lut_seq_t
00180         configCmdSeqs[3];
00181     uint32_t reserved1;
00182     uint32_t configCmdArgs[3];
00183     uint32_t reserved2;
00184     uint32_t controllerMiscOption;
00186     uint8_t deviceType;
00187     uint8_t sflashPadType;
00188     uint8_t serialClkFreq;
00190     uint8_t lutCustomSeqEnable;
00192     uint32_t reserved3[2];
00193     uint32_t sflashA1Size;
00194     uint32_t sflashA2Size;
00195     uint32_t sflashB1Size;
00196     uint32_t sflashB2Size;
00197     uint32_t csPadSettingOverride;
00198     uint32_t sclkPadSettingOverride;
00199     uint32_t dataPadSettingOverride;
00200     uint32_t dqsPadSettingOverride;
00201     uint32_t timeoutInMs;
00202     uint32_t commandInterval;
00203     uint16_t dataValidTime[2];
00204     uint16_t busyOffset;
00205     uint16_t busyBitPolarity;
00207     uint32_t lookupTable[64];
00208     flexspi_lut_seq_t lutCustomSeq[12];
00209     uint32_t reserved4[4];
00210 } flexspi_mem_config_t;
00211
00212 /*  */
00213 #define NOR_CMD_INDEX_READ CMD_INDEX_READ
00214 #define NOR_CMD_INDEX_READSTATUS CMD_INDEX_READSTATUS
00215 #define NOR_CMD_INDEX_WRITEENABLE CMD_INDEX_WRITEENABLE
00216 #define NOR_CMD_INDEX_ERASESECTOR 3
00217 #define NOR_CMD_INDEX_PAGEPROGRAM CMD_INDEX_WRITE
00218 #define NOR_CMD_INDEX_CHIPERASE 5
00219 #define NOR_CMD_INDEX_DUMMY 6
00220 #define NOR_CMD_INDEX_ERASEBLOCK 7
00221
00222 #define NOR_CMD_LUT_SEQ_IDX_READ CMD_LUT_SEQ_IDX_READ
00223 #define NOR_CMD_LUT_SEQ_IDX_READSTATUS \
00224     CMD_LUT_SEQ_IDX_READSTATUS
00225 #define NOR_CMD_LUT_SEQ_IDX_READSTATUS_XPI \
00226     2
00227 #define NOR_CMD_LUT_SEQ_IDX_WRITEENABLE \
00228     CMD_LUT_SEQ_IDX_WRITEENABLE
00229 #define NOR_CMD_LUT_SEQ_IDX_WRITEENABLE_XPI \
00230     4
00231 #define NOR_CMD_LUT_SEQ_IDX_ERASESECTOR 5
00232 #define NOR_CMD_LUT_SEQ_IDX_ERASEBLOCK 8
00233 #define NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM \
00234     CMD_LUT_SEQ_IDX_WRITE
00235 #define NOR_CMD_LUT_SEQ_IDX_CHIPERASE 11
```

```
00236 #define NOR_CMD_LUT_SEQ_IDX_READ_SFDP 13
00237 #define NOR_CMD_LUT_SEQ_IDX_RESTORE_NOCMD \
00238     14
00239 #define NOR_CMD_LUT_SEQ_IDX_EXIT_NOCMD \
00240     15
00241
00242 /*
00243  *  Serial NOR configuration block
00244  */
00245 typedef struct _flexspi_nor_config
00246 {
00247     flexspi_mem_config_t memConfig;
00248     uint32_t pageSize;
00249     uint32_t sectorSize;
00250     uint8_t ipcmdSerialClkFreq;
00251     uint8_t isUniformBlockSize;
00252     uint8_t reserved0[2];
00253     uint8_t serialNorType;
00254     uint8_t needExitNoCmdMode;
00255     uint8_t halfClkForNonReadCmd;
00256     uint8_t needRestoreNoCmdMode;
00257     uint32_t blockSize;
00258     uint32_t reserve2[11];
00259 } flexspi_nor_config_t;
00260
00261 #ifdef __cplusplus
00262 extern "C" {
00263 #endif
00264
00265 #ifdef __cplusplus
00266 }
00267 #endif
00268 #endif /* __EVKMIMXRT1060_FLEXSPI_NOR_CONFIG__ */
```

24.570 fsl_flexspi_nor_boot.h

```
00001 /*
00002  * Copyright 2017 NXP
00003  * All rights reserved.
00004  *
00005  * SPDX-License-Identifier: BSD-3-Clause
00006  */
00007
00008 #ifndef __FLEXSPI_NOR_BOOT_H__
00009 #define __FLEXSPI_NOR_BOOT_H__
00010
00011 #include <stdint.h>
00012
00016 #define FSL_XIP_DEVICE_DRIVER_VERSION (MAKE_VERSION(2, 0, 0))
00019 /************************************
00020  *  IVT Data
00021  ************************************/
00022 typedef struct _ivt_ {
00026     uint32_t hdr;
00030     uint32_t entry;
00032     uint32_t reserved1;
00034     uint32_t dcd;
00038     uint32_t boot_data;
00040     uint32_t self;
00042     uint32_t csf;
00044     uint32_t reserved2;
00045 } ivt;
00046
00047 #define IVT_MAJOR_VERSION           0x4
00048 #define IVT_MAJOR_VERSION_SHIFT     0x4
00049 #define IVT_MAJOR_VERSION_MASK      0xF
00050 #define IVT_MINOR_VERSION           0x1
00051 #define IVT_MINOR_VERSION_SHIFT     0x0
00052 #define IVT_MINOR_VERSION_MASK      0xF
00053
00054 #define IVT_VERSION(major, minor)   \
00055   ((((major) & IVT_MAJOR_VERSION_MASK) << IVT_MAJOR_VERSION_SHIFT) |  \
00056   (((minor) & IVT_MINOR_VERSION_MASK) << IVT_MINOR_VERSION_SHIFT))
00057
00058 /* IVT header */
00059 #define IVT_TAG_HEADER        0xD1
00060 #define IVT_SIZE              0x2000
00061 #define IVT_PAR               IVT_VERSION(IVT_MAJOR_VERSION, IVT_MINOR_VERSION)
00062 #define IVT_HEADER           (IVT_TAG_HEADER | (IVT_SIZE << 8) | (IVT_PAR << 24))
00063
00064 /* Set resume entry */
00065 #if defined(__CC_ARM) || defined(__ARMCC_VERSION)
00066     extern uint32_t __Vectors[];
00067     extern uint32_t Image$$RW_m_config_text$$Base[];
00068 #define IMAGE_ENTRY_ADDRESS ((uint32_t)__Vectors)
```

```
00069 #define FLASH_BASE ((uint32_t)Image$$RW_m_config_text$$Base)
00070 #elif defined(__MCUXPRESSO)
00071     extern uint32_t __Vectors[];
00072     extern uint32_t __boot_hdr_start__[];
00073 #define IMAGE_ENTRY_ADDRESS ((uint32_t)__Vectors)
00074 #define FLASH_BASE          ((uint32_t)__boot_hdr_start__)
00075 #elif defined(__ICCARM__)
00076     extern uint32_t __VECTOR_TABLE[];
00077     extern uint32_t m_boot_hdr_conf_start[];
00078 #define IMAGE_ENTRY_ADDRESS ((uint32_t)__VECTOR_TABLE)
00079 #define FLASH_BASE ((uint32_t)m_boot_hdr_conf_start)
00080 #elif defined(__GNUC__)
00081     extern uint32_t __VECTOR_TABLE[];
00082     extern uint32_t __FLASH_BASE[];
00083 #define IMAGE_ENTRY_ADDRESS ((uint32_t)__VECTOR_TABLE)
00084 #define FLASH_BASE ((uint32_t)__FLASH_BASE)
00085 #endif
00086
00087 #define DCD_ADDRESS           dcd_data
00088 #define BOOT_DATA_ADDRESS     &boot_data
00089 #define CSF_ADDRESS           0
00090 #define IVT_RSVD             (uint32_t)(0x00000000)
00091
00092 /*************************************
00093  *  Boot Data
00094  *************************************/
00095 typedef struct _boot_data_ {
00096   uint32_t start;           /* boot start location */
00097   uint32_t size;            /* size */
00098   uint32_t plugin;          /* plugin flag - 1 if downloaded application is plugin */
00099   uint32_t placeholder;     /* placehoder to make even 0x10 size */
00100 }BOOT_DATA_T;
00101
00102 #define FLASH_SIZE            (uint32_t)(0x800000UL)
00103 //#if defined(BOARD_FLASH_SIZE)
00104 //#define FLASH_SIZE            BOARD_FLASH_SIZE
00105 //#else
00106 //#error "Please define macro BOARD_FLASH_SIZE"
00107 //#endif
00108 #define PLUGIN_FLAG           (uint32_t)0
00109
00110 /* External Variables */
00111 const BOOT_DATA_T boot_data;
00112 extern const uint8_t dcd_data[];
00113
00114 #endif /* __FLEXSPI_NOR_BOOT_H__ */
00115
```

24.571 hal_platdefs.h

```
00001 #ifndef  __HAL_PLATDEFS_H
00002 #define  __HAL_PLATDEFS_H
00003 /*NB_REVISION*/
00004
00005 /*NB_COPYRIGHT*/
00006
00007 #define PLAT_STORAGE_MAX_CONFIG_SIZE        0x8000
00008 #define PLAT_STORAGE_MAX_USERPARAMS_SIZE    0x4000
00009 #define PLAT_STORAGE_MAX_CERTSTORE_SIZE     0x200000
00010 #define PLAT_STORAGE_MAX_APP_SIZE           0x2000000
00011 #define PLAT_STORAGE_MAX_FILESYS_SIZE       0x1F00000
00012
00013 #endif   /* ----- #ifndef __HAL_PLATDEFS_H  ----- */
```

24.572 imx_boot.h

```
00001 /*NB_REVISION*/
00002
00003 /*NB_COPYRIGHT*/
00004 #ifndef  __IMX_BOOT_H
00005 #define  __IMX_BOOT_H
00006
00007 #include <predef.h>
00008 #include <basictypes.h>
00009 #include <sim.h>
00010 #include <hal.h>
00011
00012 #define IVT_TAG     (0xD1)
00013
00014 #define DCD_CMD_WR  (0xCC)
00015 #define DCD_CMD_CK  (0xCF)
00016 #define DCD_CMD_NOP (0xC0)
```

```
00017
00018 #define DCD_WR_WR   (0x00)
00019 #define DCD_WR_CLR  (0x40)
00020 #define DCD_WR_SET  (0xC0)
00021 #define DCD_WR_BYTE (0x01)
00022 #define DCD_WR_HALF (0x02)
00023 #define DCD_WR_WORD (0x04)
00024
00025 #define DCD_CK_ALL_CLR  (0x00)
00026 #define DCD_CK_ALL_SET  (0x80)
00027 #define DCD_CK_ANY_CLR  (0x40)
00028 #define DCD_CK_ANY_SET  (0xC0)
00029 #define DCD_CK_BYTE     (0x01)
00030 #define DCD_CK_HALF     (0x02)
00031 #define DCD_CK_WORD     (0x04)
00032
00033 struct FlexSpi_Cfg_Block_t {
00034     uint32_t    tag;
00035     uint32_t    version;
00036     uint8_t     _reserved0;
00037     uint8_t     readSampleClkSrc;
00038     uint8_t     csHoldTime;
00039     uint8_t     csSetupTime;
00040     uint8_t     colAddrWidth;
00041     uint8_t     devModeCfgEnable;
00042     uint8_t     _reserved1;
00043     uint16_t    waitTimeCfgCommands;
00044     uint32_t    devModeSeq;
00045     uint32_t    devModeArg;
00046
00047 } __attribute__((packed));
00048
00049 struct DCD_header_t {
00050     uint8_t tag;
00051     uint16_t beLength; // needs to be big endian
00052     uint8_t version;
00053 };
00054
00055 struct DCD_cmdhdr_t {
00056     uint8_t tag;
00057     uint16_t beLength; // needs to be big endian
00058     uint8_t parameter;
00059 };
00060
00061 struct DCD_cmd_wr_t {
00062     uint32_t addr;
00063     uint32_t val_mask;
00064 };
00065
00066 struct DCD_cmd_ck_t {
00067     uint32_t addr;
00068     uint32_t mask;
00069     uint32_t count;
00070 };
00071
00072 struct Boot_Data_t {
00073     uint32_t startAddr;
00074     uint32_t length;
00075     uint32_t isPlugin;
00076 };
00077
00078 struct IVT_header_t {
00079     uint8_t tag;
00080     uint16_t beLength; // needs to be big endian
00081     uint8_t version;
00082 };
00083
00084 struct IVT_t {
00085     IVT_header_t header;
00086     uint32_t entry;
00087     uint32_t reserved1;
00088     uint32_t devConfData;
00089     uint32_t bootData;
00090     IVT_t   *self;
00091     uint32_t comSeqFile;
00092     uint32_t reserved2;
00093 };
00094
00095 #endif   /* ----- #ifndef __IMX_BOOT_H  ----- */
```

24.573 xxhash.h

```
00001 /*
00002  * xxHash - Extremely Fast Hash algorithm
00003  * Header File
```

```
00004  * Copyright (c) 2012-2020, Yann Collet, Facebook, Inc.
00005  *
00006  * You can contact the author at :
00007  * - xxHash source repository : https://github.com/Cyan4973/xxHash
00008  *
00009  * This source code is licensed under both the BSD-style license (found in the
00010  * LICENSE file in the root directory of this source tree) and the GPLv2 (found
00011  * in the COPYING file in the root directory of this source tree).
00012  * You may select, at your option, one of the above-listed licenses.
00013 */
00014
00015 /* Notice extracted from xxHash homepage :
00016
00017 xxHash is an extremely fast Hash algorithm, running at RAM speed limits.
00018 It also successfully passes all tests from the SMHasher suite.
00019
00020 Comparison (single thread, Windows Seven 32 bits, using SMHasher on a Core 2 Duo @3GHz)
00021
00022 Name            Speed       Q.Score   Author
00023 xxHash          5.4 GB/s     10
00024 CrapWow         3.2 GB/s      2       Andrew
00025 MumurHash 3a    2.7 GB/s     10       Austin Appleby
00026 SpookyHash      2.0 GB/s     10       Bob Jenkins
00027 SBox            1.4 GB/s      9       Bret Mulvey
00028 Lookup3         1.2 GB/s      9       Bob Jenkins
00029 SuperFastHash   1.2 GB/s      1       Paul Hsieh
00030 CityHash64      1.05 GB/s    10       Pike & Alakuijala
00031 FNV             0.55 GB/s     5       Fowler, Noll, Vo
00032 CRC32           0.43 GB/s     9
00033 MD5-32          0.33 GB/s    10       Ronald L. Rivest
00034 SHA1-32         0.28 GB/s    10
00035
00036 Q.Score is a measure of quality of the hash function.
00037 It depends on successfully passing SMHasher test set.
00038 10 is a perfect score.
00039
00040 A 64-bits version, named XXH64, is available since r35.
00041 It offers much better speed, but for 64-bits applications only.
00042 Name     Speed on 64 bits    Speed on 32 bits
00043 XXH64       13.8 GB/s            1.9 GB/s
00044 XXH32        6.8 GB/s            6.0 GB/s
00045 */
00046
00047 #if defined (__cplusplus)
00048 extern "C" {
00049 #endif
00050
00051 #ifndef XXHASH_H_5627135585666179
00052 #define XXHASH_H_5627135585666179 1
00053
00054
00055 /* ****************************
00056 *  Definitions
00057 ******************************/
00058 #include <stddef.h>   /* size_t */
00059 #include <stdint.h>   /* uintxx_t */
00060 typedef enum { XXH_OK=0, XXH_ERROR } XXH_errorcode;
00061
00062
00063 /* ****************************
00064 *  API modifier
00065 ******************************/
00075 #ifdef XXH_PRIVATE_API
00076 #  ifndef XXH_STATIC_LINKING_ONLY
00077 #    define XXH_STATIC_LINKING_ONLY
00078 #  endif
00079 #  if defined(__GNUC__)
00080 #    define XXH_PUBLIC_API static __inline __attribute__((unused))
00081 #  elif defined (__cplusplus) || (defined (__STDC_VERSION__) && (__STDC_VERSION__ >= 199901L) /* C99
      */)
00082 #    define XXH_PUBLIC_API static inline
00083 #  elif defined(_MSC_VER)
00084 #    define XXH_PUBLIC_API static __inline
00085 #  else
00086 #    define XXH_PUBLIC_API static   /* this version may generate warnings for unused static functions;
      disable the relevant warning */
00087 #  endif
00088 #else
00089 #  define XXH_PUBLIC_API   /* do nothing */
00090 #endif /* XXH_PRIVATE_API */
00091
00103 #ifdef XXH_NAMESPACE
00104 #  define XXH_CAT(A,B) A##B
00105 #  define XXH_NAME2(A,B) XXH_CAT(A,B)
00106 #  define XXH32 XXH_NAME2(XXH_NAMESPACE, XXH32)
00107 #  define XXH64 XXH_NAME2(XXH_NAMESPACE, XXH64)
00108 #  define XXH_versionNumber XXH_NAME2(XXH_NAMESPACE, XXH_versionNumber)
```

```
00109 #  define XXH32_createState XXH_NAME2(XXH_NAMESPACE, XXH32_createState)
00110 #  define XXH64_createState XXH_NAME2(XXH_NAMESPACE, XXH64_createState)
00111 #  define XXH32_freeState XXH_NAME2(XXH_NAMESPACE, XXH32_freeState)
00112 #  define XXH64_freeState XXH_NAME2(XXH_NAMESPACE, XXH64_freeState)
00113 #  define XXH32_reset XXH_NAME2(XXH_NAMESPACE, XXH32_reset)
00114 #  define XXH64_reset XXH_NAME2(XXH_NAMESPACE, XXH64_reset)
00115 #  define XXH32_update XXH_NAME2(XXH_NAMESPACE, XXH32_update)
00116 #  define XXH64_update XXH_NAME2(XXH_NAMESPACE, XXH64_update)
00117 #  define XXH32_digest XXH_NAME2(XXH_NAMESPACE, XXH32_digest)
00118 #  define XXH64_digest XXH_NAME2(XXH_NAMESPACE, XXH64_digest)
00119 #  define XXH32_copyState XXH_NAME2(XXH_NAMESPACE, XXH32_copyState)
00120 #  define XXH64_copyState XXH_NAME2(XXH_NAMESPACE, XXH64_copyState)
00121 #  define XXH32_canonicalFromHash XXH_NAME2(XXH_NAMESPACE, XXH32_canonicalFromHash)
00122 #  define XXH64_canonicalFromHash XXH_NAME2(XXH_NAMESPACE, XXH64_canonicalFromHash)
00123 #  define XXH32_hashFromCanonical XXH_NAME2(XXH_NAMESPACE, XXH32_hashFromCanonical)
00124 #  define XXH64_hashFromCanonical XXH_NAME2(XXH_NAMESPACE, XXH64_hashFromCanonical)
00125 #endif
00126
00127
00128 /* *************************************
00129 *  Version
00130 ***************************************/
00131 #define XXH_VERSION_MAJOR    0
00132 #define XXH_VERSION_MINOR    6
00133 #define XXH_VERSION_RELEASE  2
00134 #define XXH_VERSION_NUMBER  (XXH_VERSION_MAJOR *100*100 + XXH_VERSION_MINOR *100 +
      XXH_VERSION_RELEASE)
00135 XXH_PUBLIC_API unsigned XXH_versionNumber (void);
00136
00137
00138 /* *****************************
00139 *  Simple Hash Functions
00140 *******************************/
00141 typedef unsigned int       XXH32_hash_t;
00142 typedef unsigned long long XXH64_hash_t;
00143
00144 XXH_PUBLIC_API XXH32_hash_t XXH32 (const void* input, size_t length, unsigned int seed);
00145 XXH_PUBLIC_API XXH64_hash_t XXH64 (const void* input, size_t length, unsigned long long seed);
00146
00160 /* *****************************
00161 *  Streaming Hash Functions
00162 *******************************/
00163 typedef struct XXH32_state_s XXH32_state_t;   /* incomplete type */
00164 typedef struct XXH64_state_s XXH64_state_t;   /* incomplete type */
00165
00168 XXH_PUBLIC_API XXH32_state_t* XXH32_createState(void);
00169 XXH_PUBLIC_API XXH_errorcode  XXH32_freeState(XXH32_state_t* statePtr);
00170
00171 XXH_PUBLIC_API XXH64_state_t* XXH64_createState(void);
00172 XXH_PUBLIC_API XXH_errorcode  XXH64_freeState(XXH64_state_t* statePtr);
00173
00174
00175 /* hash streaming */
00176
00177 XXH_PUBLIC_API XXH_errorcode XXH32_reset  (XXH32_state_t* statePtr, unsigned int seed);
00178 XXH_PUBLIC_API XXH_errorcode XXH32_update (XXH32_state_t* statePtr, const void* input, size_t length);
00179 XXH_PUBLIC_API XXH32_hash_t  XXH32_digest (const XXH32_state_t* statePtr);
00180
00181 XXH_PUBLIC_API XXH_errorcode XXH64_reset  (XXH64_state_t* statePtr, unsigned long long seed);
00182 XXH_PUBLIC_API XXH_errorcode XXH64_update (XXH64_state_t* statePtr, const void* input, size_t length);
00183 XXH_PUBLIC_API XXH64_hash_t  XXH64_digest (const XXH64_state_t* statePtr);
00184
00185 /*
00186 These functions generate the xxHash of an input provided in multiple segments.
00187 Note that, for small input, they are slower than single-call functions, due to state management.
00188 For small input, prefer `XXH32()` and `XXH64()` .
00189
00190 XXH state must first be allocated, using XXH*_createState() .
00191
00192 Start a new hash by initializing state with a seed, using XXH*_reset().
00193
00194 Then, feed the hash state by calling XXH*_update() as many times as necessary.
00195 Obviously, input must be allocated and read accessible.
00196 The function returns an error code, with 0 meaning OK, and any other value meaning there is an error.
00197
00198 Finally, a hash value can be produced anytime, by using XXH*_digest().
00199 This function returns the nn-bits hash as an int or long long.
00200
00201 It's still possible to continue inserting input into the hash state after a digest,
00202 and generate some new hashes later on, by calling again XXH*_digest().
00203
00204 When done, free XXH state space if it was allocated dynamically.
00205 */
00206
00207
00208 /* *************************
00209 *  Utils
```

```
00210 ***************************/
00211 #if !(defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 199901L))   /* ! C99 */
00212 #  define restrict   /* disable restrict */
00213 #endif
00214
00215 XXH_PUBLIC_API void XXH32_copyState(XXH32_state_t* restrict dst_state, const XXH32_state_t* restrict
      src_state);
00216 XXH_PUBLIC_API void XXH64_copyState(XXH64_state_t* restrict dst_state, const XXH64_state_t* restrict
      src_state);
00217
00218
00219 /* **************************
00220 *  Canonical representation
00221 ****************************/
00222 /* Default result type for XXH functions are primitive unsigned 32 and 64 bits.
00223 *  The canonical representation uses human-readable write convention, aka big-endian (large digits
      first).
00224 *  These functions allow transformation of hash result into and from its canonical format.
00225 *  This way, hash values can be written into a file / memory, and remain comparable on different
      systems and programs.
00226 */
00227 typedef struct { unsigned char digest[4]; } XXH32_canonical_t;
00228 typedef struct { unsigned char digest[8]; } XXH64_canonical_t;
00229
00230 XXH_PUBLIC_API void XXH32_canonicalFromHash(XXH32_canonical_t* dst, XXH32_hash_t hash);
00231 XXH_PUBLIC_API void XXH64_canonicalFromHash(XXH64_canonical_t* dst, XXH64_hash_t hash);
00232
00233 XXH_PUBLIC_API XXH32_hash_t XXH32_hashFromCanonical(const XXH32_canonical_t* src);
00234 XXH_PUBLIC_API XXH64_hash_t XXH64_hashFromCanonical(const XXH64_canonical_t* src);
00235
00236 #endif /* XXHASH_H_5627135585666179 */
00237
00238
00239
00240 /* ================================================================================================
00241    This section contains definitions which are not guaranteed to remain stable.
00242    They may change in future versions, becoming incompatible with a different version of the library.
00243    They shall only be used with static linking.
00244    Never use these definitions in association with dynamic linking !
00245 =================================================================================================== */
00246 #if defined(XXH_STATIC_LINKING_ONLY) && !defined(XXH_STATIC_H_3543687687345)
00247 #define XXH_STATIC_H_3543687687345
00248
00249 /* These definitions are only meant to allow allocation of XXH state
00250    statically, on stack, or in a struct for example.
00251    Do not use members directly. */
00252
00253    struct XXH32_state_s {
00254        uint32_t total_len_32;
00255        uint32_t large_len;
00256        uint32_t v1;
00257        uint32_t v2;
00258        uint32_t v3;
00259        uint32_t v4;
00260        uint32_t mem32[4];   /* buffer defined as U32 for alignment */
00261        uint32_t memsize;
00262        uint32_t reserved;   /* never read nor write, will be removed in a future version */
00263    };   /* typedef'd to XXH32_state_t */
00264
00265    struct XXH64_state_s {
00266        uint64_t total_len;
00267        uint64_t v1;
00268        uint64_t v2;
00269        uint64_t v3;
00270        uint64_t v4;
00271        uint64_t mem64[4];   /* buffer defined as U64 for alignment */
00272        unsigned memsize;
00273        unsigned reserved[2];          /* never read nor write, will be removed in a future version */
00274    };   /* typedef'd to XXH64_state_t */
00275
00276
00277 #  ifdef XXH_PRIVATE_API
00278 #    include "xxhash.c"   /* include xxhash functions as `static`, for inlining */
00279 #  endif
00280
00281 #endif /* XXH_STATIC_LINKING_ONLY && XXH_STATIC_H_3543687687345 */
00282
00283
00284 #if defined (__cplusplus)
00285 }
00286 #endif
```

Index