Document Number: MQXGSRTOS
Rev 4.1.1, 08/2014

Getting Started with Freescale MQX™ RTOS

1 Read Me First

This document describes steps required to configure the Kinetis Design Studio, CodeWarrior, IAR Embedded Workbench®, DS-5, and KEIL™ development tools and use them to build, run, and debug applications of the Freescale MQX™ RTOS operating system. This document also provides board-specific information related to the MQX RTOS.

2 Building the MQX libraries

Subsequent sections show how to build the MQX libraries.

2.1 Compile-time configuration

Major compile-time configuration options are grouped in a single user configuration file located in:

```
<install_dir>/config/<board>/user_config.h
```

This user configuration file is included internally by private configuration files in MQX PSP and BSP and other core components such as RTCS, USB, and Shell. To share the

configuration settings between different boards, add header files to the user_config.h file with common settings. The header files are located either in the same <board> directory or in the “common” directory:

```
<install_dir>/config/common
```

NOTE

Prior to MQX RTOS 4.0, the configuration header files were added from different locations when compiling the BSP/PSP and other library projects. The BSP and PSP code included the user_config.h file from the original /config folder, while other libraries, such as RTCS or MFS, were using a “copied” version of the this file from the /lib folder. This was changed in MQX RTOS 4.0 such that all libraries are using the user_config.h file from the /config folder. This change removes the requirement to build the libraries in a certain order. Prior to MQX RTOS 4.0, the PSP and BSP libraries had to be built before the other libraries. This is no longer required.

2.2 Build Process

After either the compile-time user configuration file or MQX kernel source files are changed, re-build the MQX libraries.

The build process is similar for all core components:

- The output directory is: <install_dir>/lib/<board>.<compiler>/<target>/<component> (where <compiler> is a name of a build tool).

  For example, when CodeWarrior 10 tool is used to build MQX PSP and BSP libraries for TWR-K60D100M board in debug targets, the libraries are built in the /lib/twrk60d100m.cw10/debug/psp and /lib/twrk60d100m.cw10/debug/bsp directories.

  NOTE

  Since MQX RTOS 4.0, the library output paths were changed. The target names are now part of a library output path, while the name of the library file is the same for all targets. The Debug versions of the libraries are no longer created with the _d suffix which makes it easier to create custom build targets and change different versions of libraries easily in the application projects.

- All public header files, needed for the application, are automatically copied from internal include folders to the same output directory as the library itself.
- During PSP or BSP build process, the user_config.h file and other header files from the config/<board> and config/common directories are also copied into the lib/<board>.<compiler>/<target> output directory.

NOTE

After changing the /config/common/user_config.h file, recompile all MQX libraries.

No changes should be made to header files in the output build directory (/lib). The files are overwritten each time the libraries are built.

2.3 Library build targets

Each build project in Freescale MQX RTOS contains multiple compiler/linker configurations, so called, build “targets”.

Two different types of build targets exist for different compiler optimization settings:

Debug – the compiler optimizations are turned off or set to low. The compiled code is easy to debug but may be less effective and much larger than the release build. Libraries are compiled into /lib/<board>.<compiler>/debug/<component> directories.

Release – the compiler optimizations are set to maximum. The compiled code is very hard to debug and should be used for final applications only. Libraries are compiled into /lib/<board>.<compiler>/release/<component> directories.

NOTE

The library path name pattern was changed in MQX RTOS 4.0. The _d suffix for debug was removed from the library. This change simplifies creating custom build targets and also enables changing different versions of libraries easily in the application projects. In order for the application to use libraries compiled in a different build target, set the proper library search paths in the application project settings. The library names, referred by the application project, remain unchanged. Different ABI options are no longer supported by the MQX RTOS. The user may add ABI builds as custom targets.

2.4 Application build targets

Build target names of MQX application projects reference either Debug or Release builds of the core libraries. Additionally, the target names also specify the built board memory configuration. For example:

- Devices with internal Flash memory (e.g. K60D100M):
  - Int. Flash Release – This target is suitable for final application deployment. When programmed to Flash, the application starts immediately after reset. Variables are allocated in the internal SRAM memory.
  - Int. Flash Debug – Same as above, with the exception that only the Debug-compiled libraries are used. This target is suitable for debugging before deployment. This is the only target suitable for debugging larger applications for boards without external memory.
- Boards with external MRAM memory (TWRMCF52259 etc.):
  - Ext. MRAM Debug – Used solely for debugging purposes with code located in external MRAM memory (available e.g. on the TWRMCF52259).
  - Variables are located in the internal SRAM. A part of the external MRAM memory may also be used for additional RAM memory pools. An application executable is loaded to MRAM automatically by the debugger.
- Boards with external RAM memory
  - Ext. Ram Debug – Used solely for debugging purposes with code located in the external RAM memory (available as SDRAM e.g. on the M54455EVB). Both code and variables are located in the external memory. An application executable is loaded to RAM automatically by the debugger.
- Boards and devices with internal Flash memory and additional external RAM for data (TWR-K70F120M):
  - Int Flash <mem>Data Debug – The name of each target defines a memory used as the default data storage. For example, the application built with target named “Int Flash DDRData Debug” executes code out of internal Flash memory and uses the DDR memory for data storage.

2.5 Compiler and linker warning setting

Several compiler and linker warnings are suppressed in MQX release.

iar | kinetis + vybrid_a5 + vybrid_m4:

- Pa082 : undefined behavior: the order of volatile accesses is undefined in this statement
- Pe186 : pointless comparison of unsigned integer with zero
- Pe177 : variable ”[name]” was declared but never referenced
- Pe550 : variable ”[name]” was set but never used
- Pe174 : expression has no effect.

ds5 | vybrid_a5 + vybrid_m4:

- 1609 : instruction does not set bit zero, so does not create a return address
- 1296 : extended constant initializer used

uv4 | kinetis:

- 1609 : instruction does not set bit zero, so does not create a return address
- 186 : pointless comparison of unsigned integer with zero
- 1296 : extended constant initializer used
- 1581 : added <no_padbytes> bytes of padding at address <address>

gcc | kinetis + vybrid_a5 + vybrid_m4

- -Wno-missing-braces - bug in gcc
- -Wno-switch
- -Wno-unused-value
- -Wno-unused-variable
- -Wno-unused-but-set-variable
- -Wno-pointer-to-int-cast
- -Wno-unused-function
- -Wno-unused-label
- -Wno-char-subscripts
- -Wno-int-to-pointer-cast

2.6 Kinetis Design Studio

First, install the MQX KDS plugin using Help\Install New Software\ menu and select KDS update site.

To rebuild the MQX libraries, first import the build_libs.wsd working set description file by using File\Import\MQX\Import Working Sets menu. The MQX library projects will be imported to KDS working space together with build configurations settings.

The location of working set description (wsd) file is here:

```
<mqx_install_dir>/build/<board>/kds/build_libs.wsd
```

The output directory is here:

```
<mqx_install_dir>/lib/<board>.kds
```

For detailed information about the MQX support in ARM tools, see *Getting Started with Kinetis Design Studio IDE and Freescale MQX*™ RTOS (document MQXKDSUG). The document is included in the MQX installation in the `<mqx_install_dir>/doc/tools/kds/MQX_KDS_Getting_Started.pdf`.

2.7 Freescale CodeWarrior development studio version 10

To rebuild the MQX libraries, navigate to an appropriate folder in the Windows Explorer, drag the build_lib.wsd file and drop it into the CodeWarrior project view. The MQX library projects and build configurations settings are imported to CodeWarrior working space.

Location of working set description (wsd) file is here:

```
<mqx_install_dir>/build/<board>/cw10gcc/build_lib.wsd
```

The output directory is here:

```
<mqx_install_dir>/lib/<board>.cw10gcc/
```

For detailed information about importing and building MQX libraries and debugging MQX application in CodeWarrior 10, see the *CW for Microcontrollers v10 and MQX™ RTOS* (document MQXCWPP). This document is part of the MQX installation package.

NOTE

The wsd file import requires the installation of the MQX CodeWarrior plugins. The functionality is supported for CodeWarrior version 10.2 or newer and requires the MQX CW plugin installed. For installation description, see Chapter 7.3, CodeWarrior 10 Task Aware Debugger Plug-in.

2.8 IAR Embedded Workbench for ARM® and ColdFire

To rebuild the MQX libraries, open and batch-build this IAR EWARM workspace:

```
<mqx_install_dir>/build/<board>/iar/ build_libs.eww
```

The output directory is here:

```
<mqx_install_dir>/lib/<board>.iar/
```

For detailed information about the MQX support in IAR tools, see *Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench®* (document MQXGSIAR). The document is included in the MQX installation in the <mqx_install_dir>/doc/tools/iar/MQX_IAR_Getting_Started.pdf.

2.9 ARM Keil µVision4® development environment support

To rebuild the MQX libraries, open and batch-build this µVision4 Multi-Project Workspace:

```
<mqx_install_dir>/build/<board>/uv4/build_libs.uvmpw
```

The output directory is here:

```
<mqx_install_dir>/lib/<board>.uv4
```

For detailed information about the MQX support in ARM tools, see *Getting Started with Freescale MQX™ RTOS and MDK-ARM Keil™ µVision4®* (document MQXGSKEIL). The document is included in the MQX installation in the <mqx_install_dir>/doc/tools/uv4/MQX_uVision4_Getting_Started.pdf.

2.10 ARM Development Studio 5 (DS-5TM) environment support

First, install the MQX Eclipse plugin using Help\Install New Software\Add\Archive… menu. Then, select this archive:

```
<mqx_install_dir>/tools/ds5/ds5_update_site.zip
```

To rebuild the MQX libraries, first import the build_libs.wsd working set description file by using File\Import\MQX\Import Working Sets menu. The MQX library projects will be imported to DS-5 working space together with build configurations settings.

Location of working set description (wsd) file is here:

```
<mqx_install_dir>/build/<board>/ds5/build_libs.wsd
```

The output directory is here:

```
<mqx_install_dir>/lib/<board>.ds5
```

For detailed information about the MQX support in ARM tools, see *Getting Started with ARM® Design Studio 5 (DS-5™) with Freescale MQX™ RTOS* (document MQXGSDS5). The document is included in the MQX installation in the <mqx_install_dir>/doc/tools/uv4/MQX_DS5_Getting_Started.pdf.

2.11 Building the MQX RTOS using makefiles

MQX RTOS version 4.1 contains the Makefiles to build libraries and applications for selected platforms by using the command-line. The support for Makefile command line builds will be expanded to cover all boards and BSPs in the later MQX versions.

Please use the mingw32-make version 3.8.2 or higher. Download the latest version from sourceforge.net/projects/mingw/.

To rebuild the MQX libraries navigate to the <mqx_install_dir>/build/<board>/make directory and run this command (building MQX RTOS using the GCC from CW10.3 and later version of the toolchain):

```
C:\MinGW\bin\mingw32-make.exe build TOOL=cw10gcc CONFIG=debug
```

NOTE

Prior to the build, specify the path (TOOLCHAIN_ROOTDIR variable) for your build tools in the <mqx_install_dir>/build/common/make/global.mak

3 Porting User Applications to MQX RTOS version 4.1.0

Starting with the MQX RTOS version 4.1.0, the standard C99 integer and boolean types (from stdlib.h and stdbool.h) are used instead of the proprietary MQX types. When porting the application to the MQX RTOS version 4.1.0, compilation issues could occur. The psptypes_legacy.h header file is available and contains the definition of the MQX legacy types. This file can be included in the MQX application to overcome most of the compilation problems caused by the change of types. To avoid potential problems, adopt the C99 types in your application.

3.1 Changes in host-to-network and network-to-host macros

The htonl, htons, ntohl, and ntohs macros, which are used for converting values between the host and the network byte order, are modified to conform to the POSIX standard in the MQX RTOS version 4.1.0. The old implementation of these macros is moved from the pcb.h to mqx.h file and renamed with the “mqx_” prefix (mqx_htonl, mqx_htons, etc.).

The change is relevant for these functions:

uint32_t htonl(uint32_t); uint16_t htons(uint16_t); uint32_t ntohl(uint32_t); uint16_t ntohs(uint16_t);

3.2 MQX RTOS startup split

To avoid a crash risk if an interrupt occurs during the startup, the MQX RTOS startup is split in two parts.

- _bsp_enable_card() function has been replaced by the _bsp_pre_init() function that handles initialization of the OS vital functions, such as the timer (system tick), interrupt controller, memory management, etc. The _bsp_pre_init() function is called in the MQX initialization time, before any user task is created and the scheduler is not started.
- The second part of the startup is done in a separate _mqx_init_task that executes _bsp_init() function for I/O drivers or stacks initialization and _bsp_post_init() function for possible post-init operations. After the _bsp_post_init() function execution, the _mqx_init_task is destroyed.

All BSPs are adjusted to this concept. All I/O drivers are installed in the context of the _mqx_init_task after the MQX scheduler is started. This concept also allows a complex driver installation (handling ISRs during the driver initialization, drivers could use blocking functionality like _time_delay, etc.).

When porting your BSP to the MQX RTOS 4.1.0, split the _bsp_enable_card() function (init_bsp.c) into the_bsp_pre_init() and the _bsp_init() functions. All _bsp_enable_card() code up to the I/O subsystem initialization are part of the new _bsp_pre_init() function. The rest of the _bsp_enable_card() code is part of the new _bsp_init() function.

4 Creating a New MQX Project - Freescale CodeWarrior development studio version 10

The Freescale MQX RTOS setup installs the MQX “New Project Wizard” plug-in into CodeWarrior 10 installation directory. The Project Wizard helps to select one of the supported evaluation boards, include appropriate MQX libraries, and create initial application project. The project wizard is available in the File/New/MQX Projects menu in the CodeWarrior 10 IDE. Follow the steps displayed by the Wizard, specify application name, and select the target evaluation board. The wizard may create a new application or import one of the existing examples to your workspace. Start with the “New application” wizard:

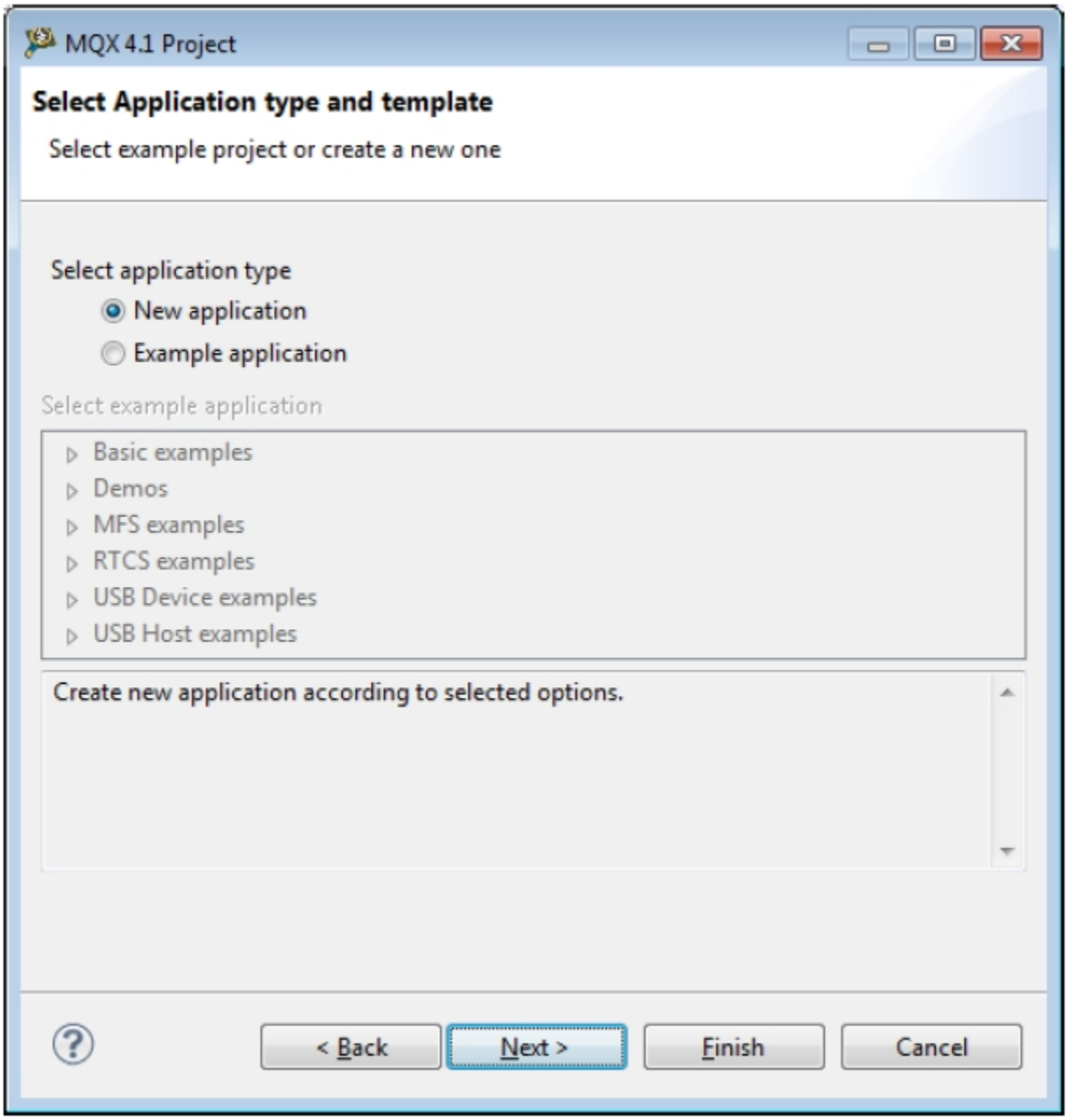

Figure 1. Select application type and template

Select the MQX libraries to be included in your project:

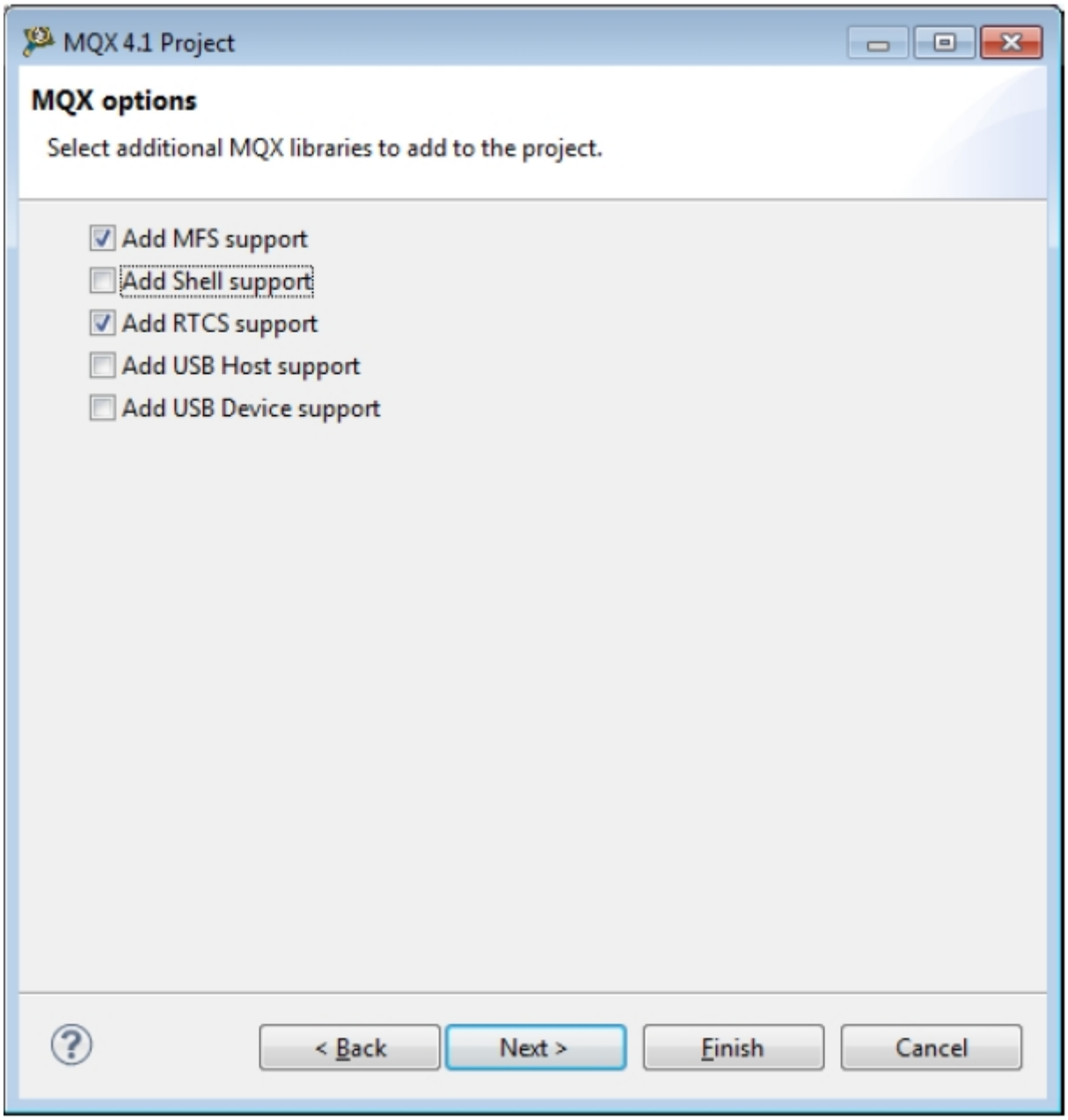

Figure 2. MQX options

The Wizard provides two types of projects:

- Empty application – a simple “Hello world” application with selected MQX libraries included
- Basic application – an application showing basic code to initialize selected MQX components

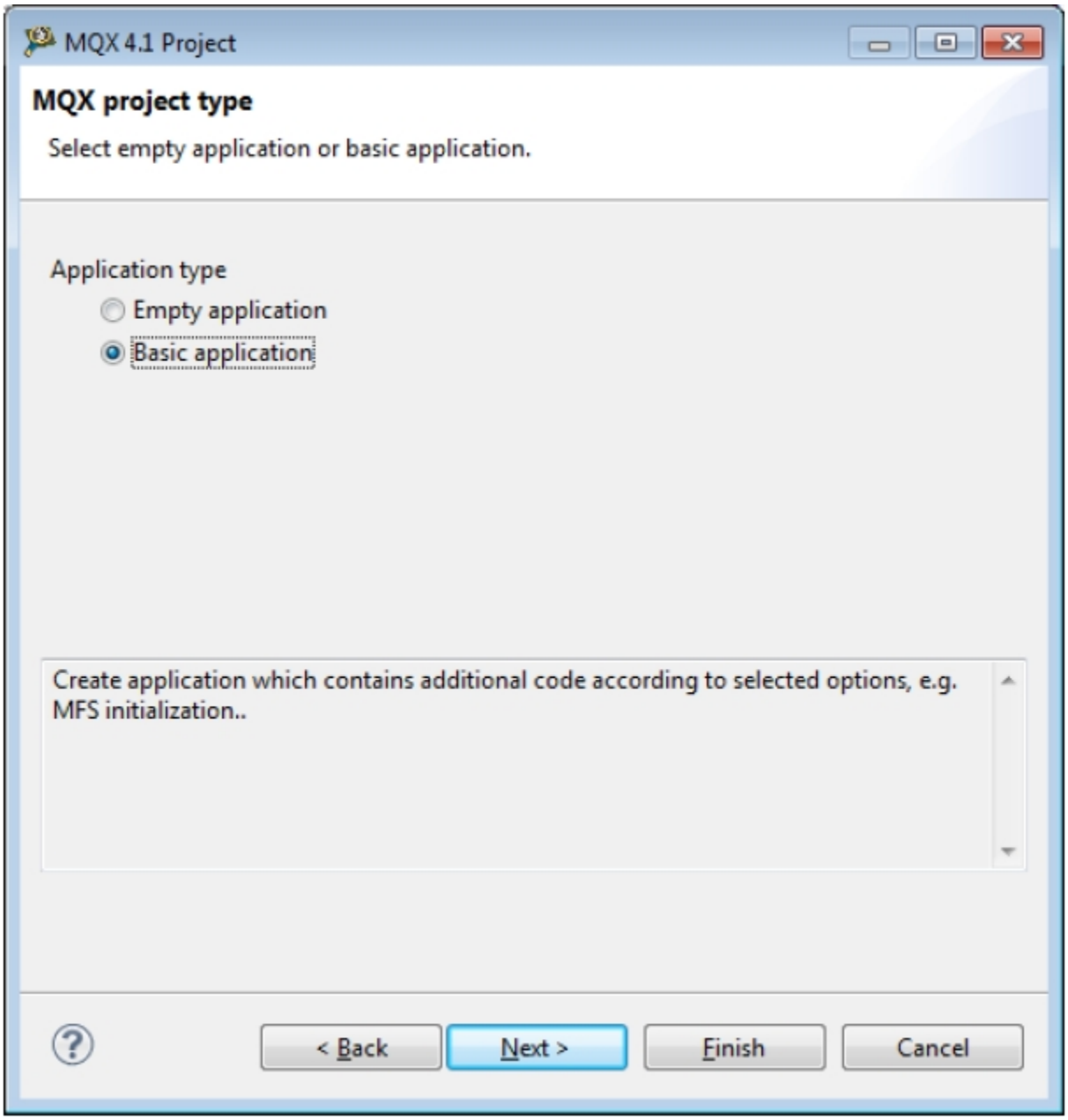

Figure 3. MQX project type

If the “Basic” application is selected, the Wizard continues by several other steps to gather information about what initialization code to generate:

- RTCS option – RTCS TCP IP stack is initialized and set to static or dynamic IP address based on user selection.
- USB Device or Host option – Stack and selected class drivers are initialized based on user selection.
- MFS option – The RAM-disk initialization code can be enabled.
- Shell option – The shell command structure can be enabled.

Finally, the debug connection can be selected before the new MQX project is created.

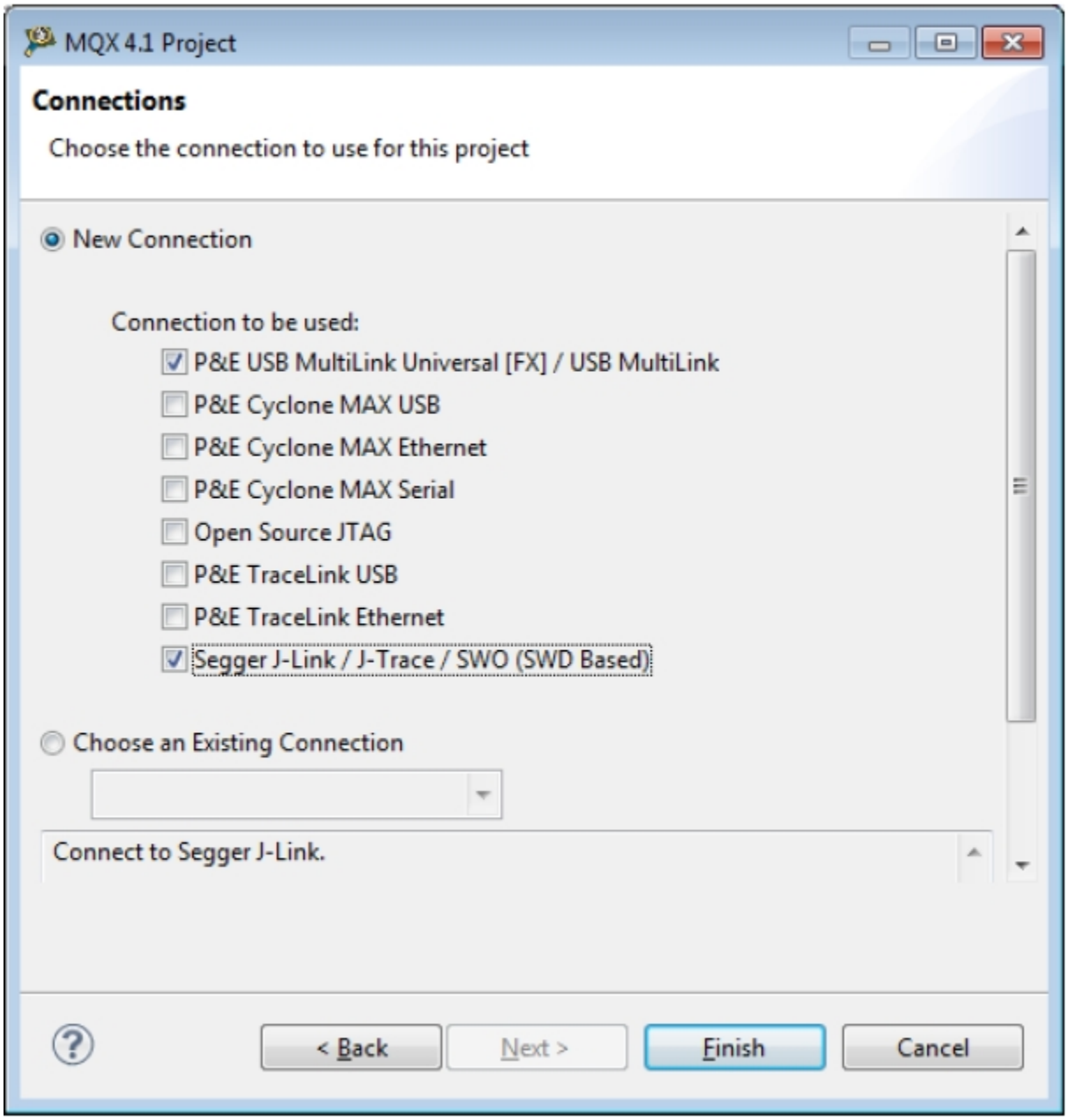

Figure 4. Connections

4.1 Other development tools

Typically, process of creating new projects mostly depends on the development environment being used. Describing this process for tools other than Freescale CodeWarrior is out of scope of this document. For more information, see the documentation provided by the tool vendor.

A general recommendation for starting a new MQX project in any IDE environment is to clone one of the existing example applications, save it under a custom name, and modify it to meet your specific needs. In this case, please be aware that there may be relative paths to support files referred to in the project. This may apply to search paths, linker command files, debugger, configuration files, etc. Make sure that you update the relative paths in the new “clone” of the project.

4.2 Standard Input and Output Channel Settings

An I/O communication device installed by MQX BSP can be used as the standard IO channel (e.g Serial line, IO Debug, telnet, USB CDC). The default console setting for each supported development board is specified in Chapter 9, Board-specific Information Related to the MQX RTOS. Freescale Tower evaluation kits offer several ways to connect the I/O console. The most common options are listed below.

- Serial I/O is a channel routed via TWR-SER or TWR-SER2 boards by using the RS232 connector. This port can be connected directly to the PC serial interface and used with a suitable terminal program (e.g. Hyper-Terminal).
- Serial I/O is a channel routed via combined OSBDM/OSJTAG debugging and communication port directly to the board. For microcontrollers, the communication interface is connected to a serial (SCI/UART) port. For a PC-Host, you can use either a virtual USB serial port driver or a special USB communication terminal application. See the development board documentation more details.
- DebugI/O is a channel which connects the PC and a target processor by using the debugging probe I/O functionality. For more information, see Chapter 5, Using debugI/O as the default I/O channel.

Default Serial I/O Channel Settings:

- Baud Rate 115200
- Data bits 8
- No parity
- 1 stop bit and no flow control

Pre-defined default I/O channel setting is specified for each board in the `mqx\source\bsp\<board_name>\board_name.h` header file. This setting can be overridden in the user_config.h file by adding this code and rebuilding the BSP library.

To set the default I/O channel to the UART2 serial interface (mapped to ttyc: device in the MQX RTOS), use this:

```
#define BSP_DEFAULT_IO_CHANNEL "ttyc:"
```

Ensure that the serial channel (ttyc: in this case) is in enabled in user_config.h file as follows:

```
#define BSPCFG_ENABLE_TTYC 1
```

5 Using debug I/O as a default I/O channel

The standard input and output channel can be redirected to the DebugI/O driver allowing the processor to communicate with the computer via the debugger probe. The MQX RTOS currently supports ARM® Cortex®-M Semihost and ITM technologies. On the PC host side, the communication has to be properly set up in the debugger. To set up the DebugI/O as the default I/O channel, add this code to the user_config.h file:

```
#define BSP_DEFAULT_IO_CHANNEL "iodebug:"
```

Enable the DebugI/O driver as follows:

```
#define BSPCFG_ENABLE_IODEBUG 1
```

See the *Freescale MQX™ RTOS I/O User's Guide* (document *MQXIOUG*) for more information on the Debug I/O driver setting (ITM vs. semihost mode, buffer setting…). The tool-related setting is described in the subsequent chapters.

NOTE

Avoid Using the I/O Debug as an input because reading from the Semihost port causes the microprocessor core to halt.

5.1 Freescale CodeWarrior development studio version 10

The CodeWarrior 10 IDE supports the Semihost communication channel for the output direction only (input is not supported by CW10.1). Use the internal DebugI/O buffer to speed up communication. See *Freescale MQX™ RTOS I/O User's Guide* (document MQXIOUG) for detailed description. Open the console from the Window/Show View/Console CW10 menu.

NOTE

Uncheck the Fixed width console option in the Run/Debug Console preferences in CW10.1.

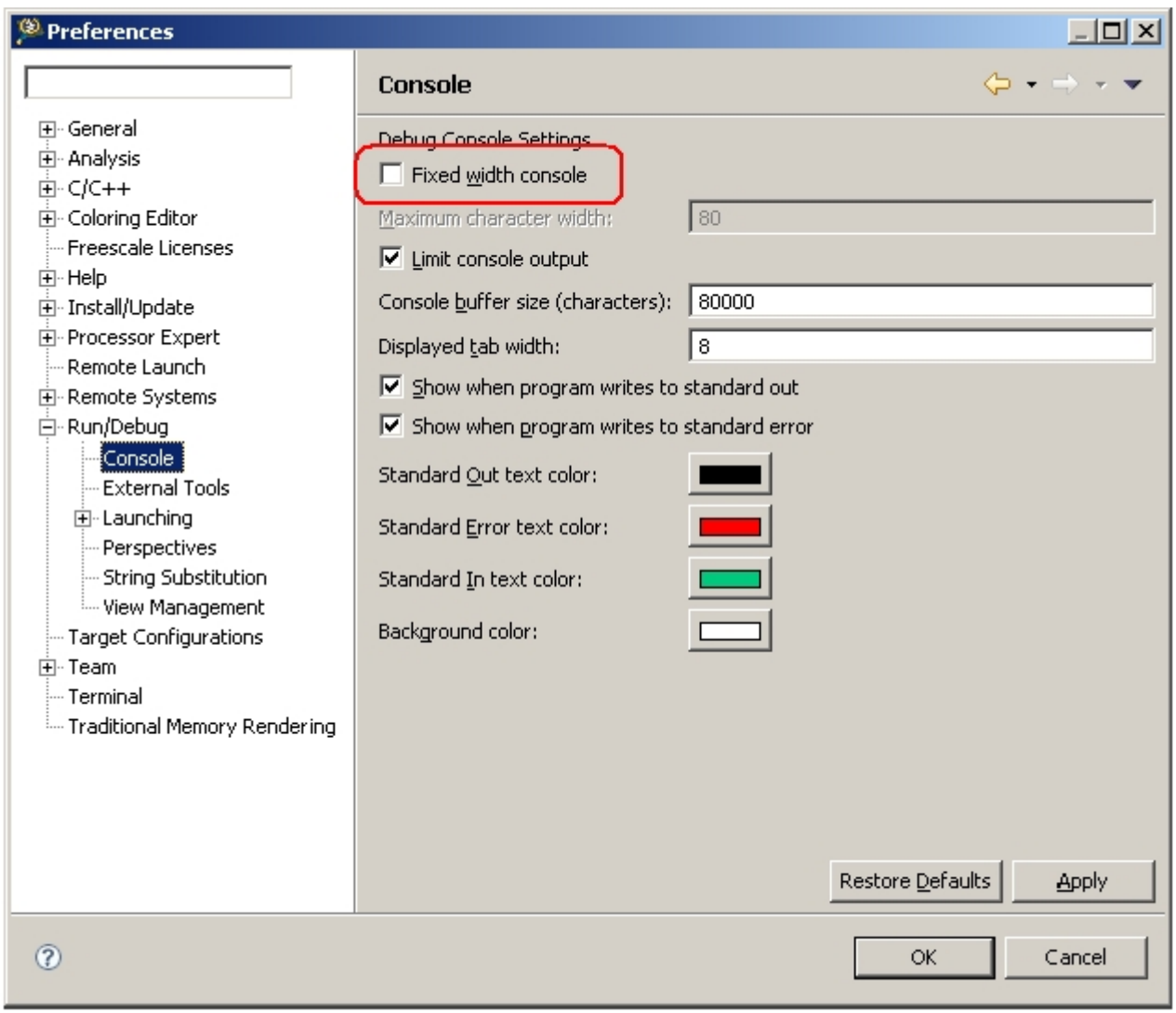

Figure 5. Console

5.2 IAR Embedded Workbench for ARM

The IAR EWARM IDE supports the Semihost communication channel for both input and output directions. Enable the Semihost options in project preferences.

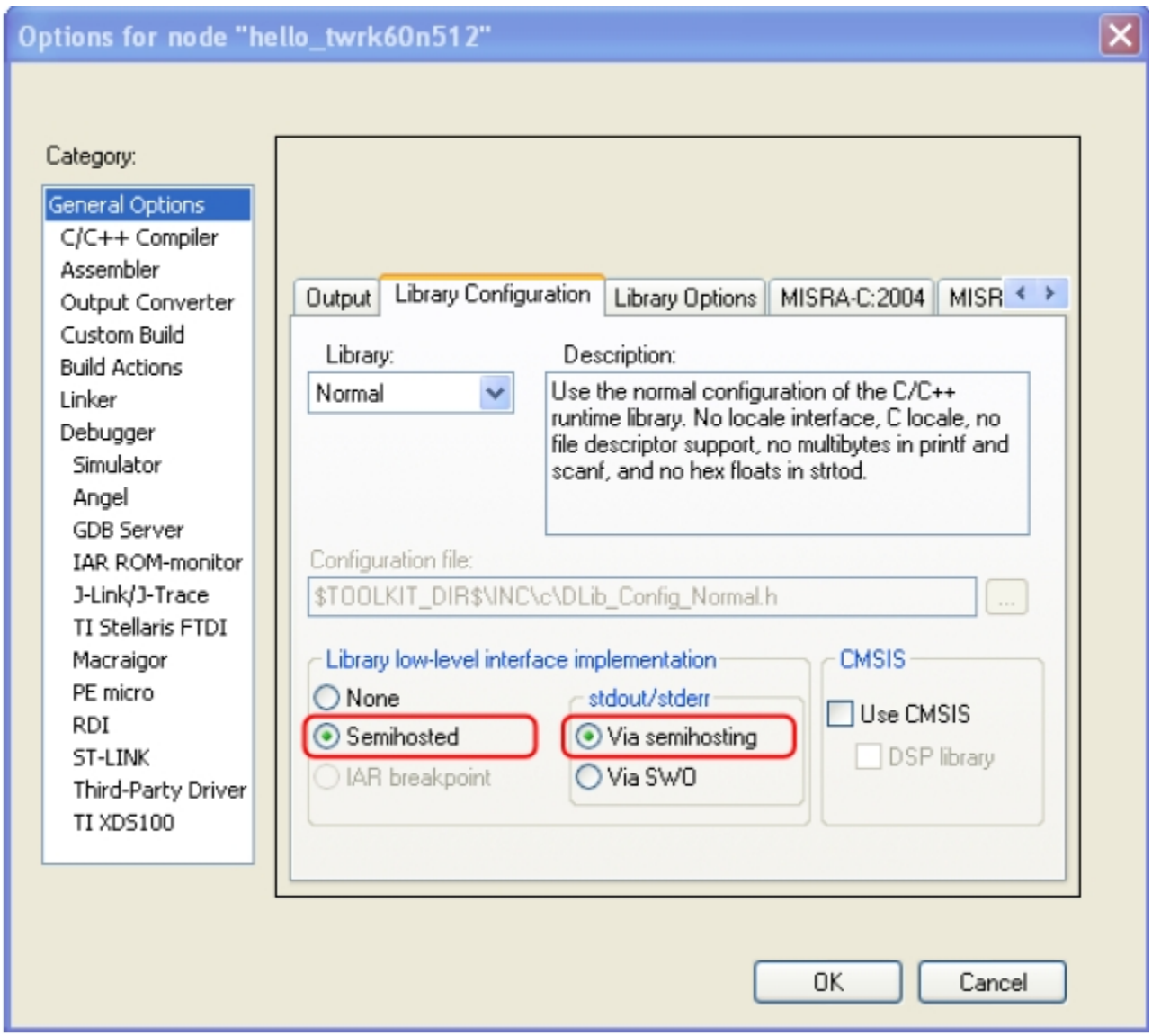

Figure 6. General Options

Open the console from the View - Terminal I/O IAR menu. For detailed information about setting up the debugger connection, see *Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench®* (document MQXGSIAR). The document is included in the MQX installation as the <mqx_install_dir>/doc/tools/iar/MQX_IAR_Getting_Started.pdf.

5.3 ARM Keil µVision4 IDE

Because ARM Keil µVision4 IDE does not support the Semihost output, use the ITM mode instead. The ITM communication channel for both input and output directions is supported. These images show how to set up ITM when using the ULINKpro programmer.

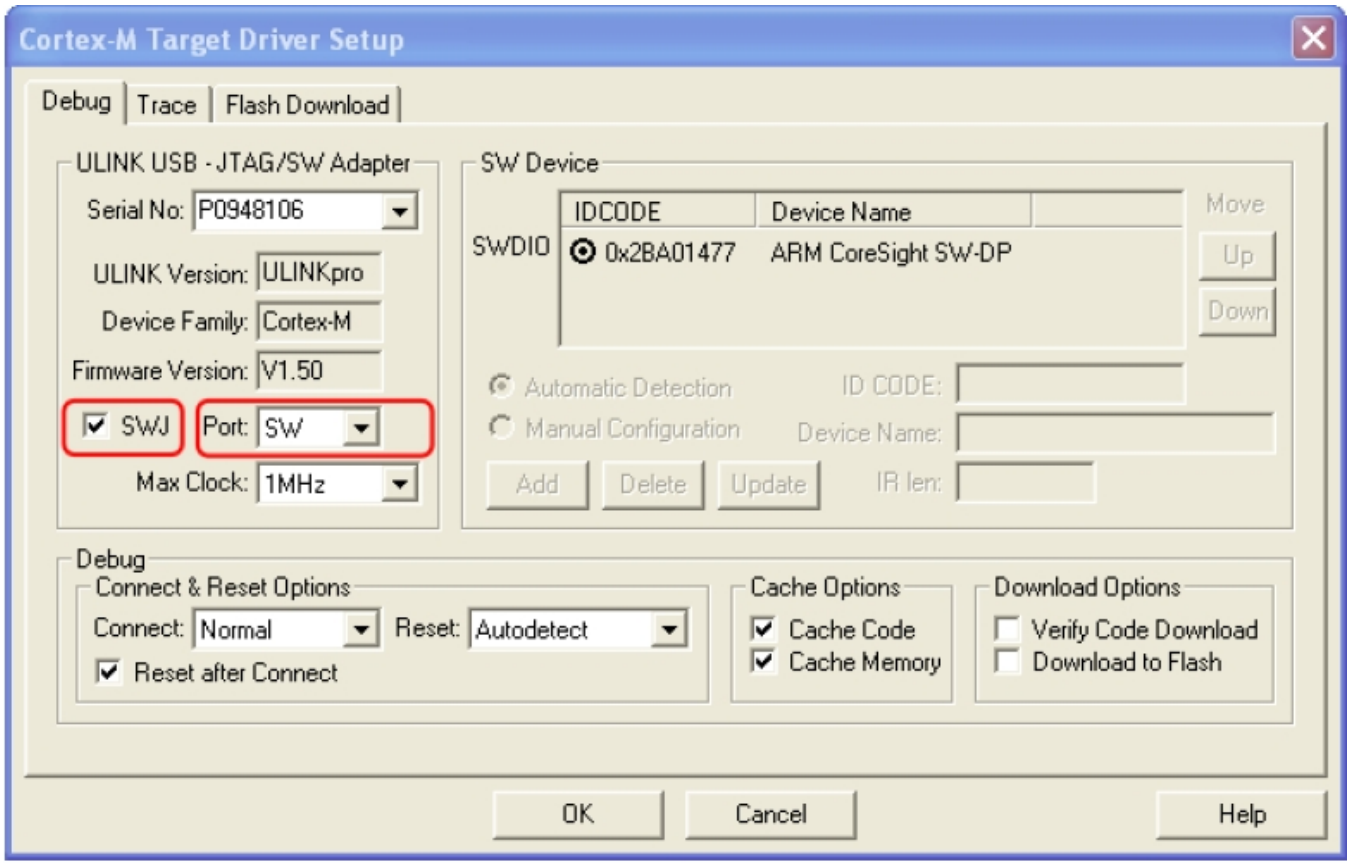

Figure 7. ARM Cortex-M target driver setup

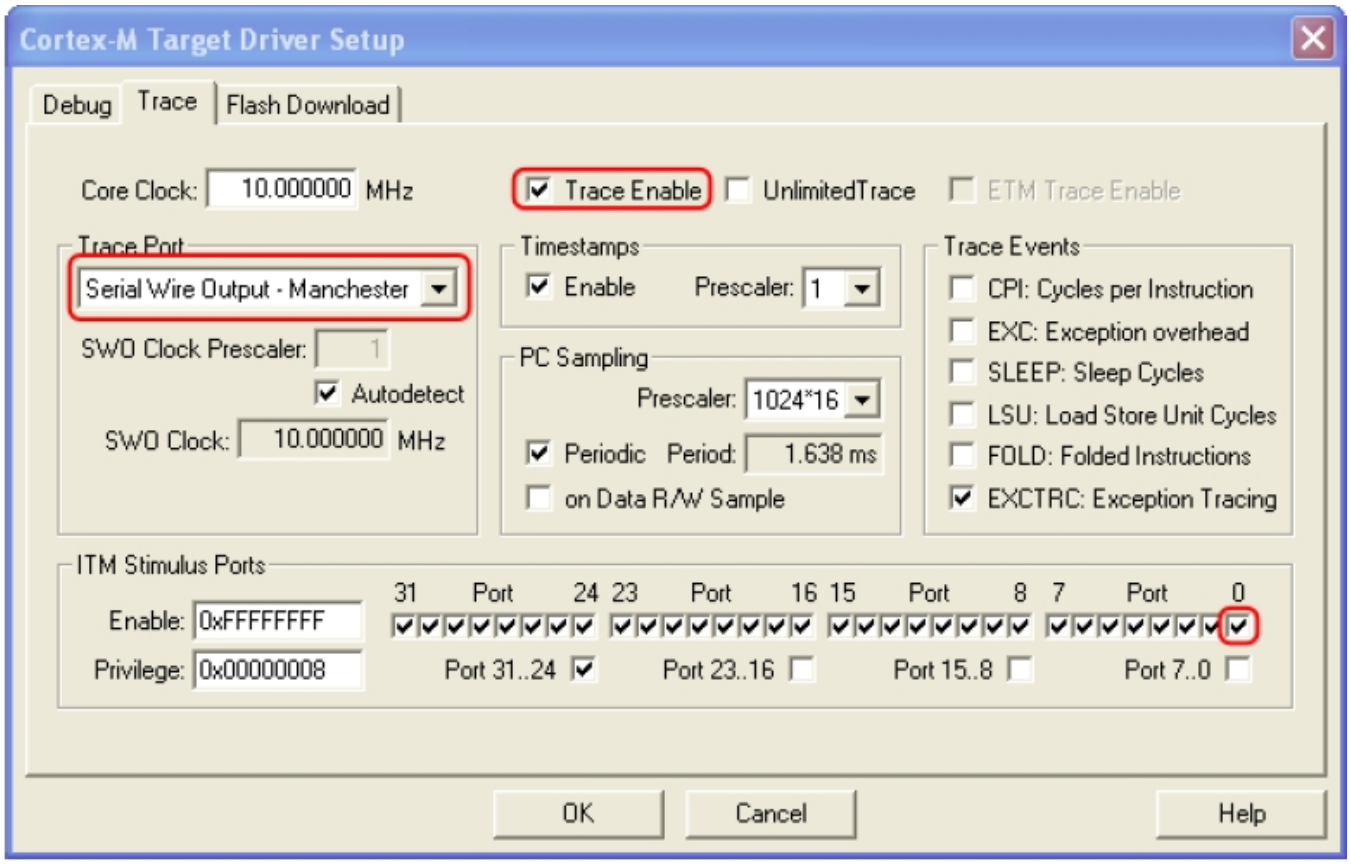

Figure 8. ARM Cortex-M target driver setup - serial wire output

To show the console view during the debug session, go to menu View / Serial Windows / Debug (printf) Viewer.

For more information about Keil and ULINK settings, navigate to this link: arm.com/files/pdf/Kinetis_LAB.pdf

For detailed information about setting up the debugger connection, see *Getting Started with Freescale MQX™ RTOS and MDK-ARM Keil™ µVision4®* (document MQXGSKEIL). The document is included in the MQX installation as the <mqx_install_dir>/doc/tools/uv4/MQX_uVision4_Getting_Started.pdf.

6 Setting default USB controller for host and device applications

Some microcontrollers have more USB controller units integrated on the chip. For MQX RTOS, the controllers are represented by structure instances exported from the BSP to the application. For example:

```
bsp_usb_host_ehci0_if, _bsp_usb_dev_ehci1_if
```

The structure names are further abstracted by the `USBCFG_DEFAULT_HOST_CONTROLLER` and `USBCFG_DEFAULT_DEVICE_CONTROLLER` macros. That allows the MQX RTOS example applications to use a generic approach for every processor and board setup.

To change the default USB controller in a BSP, modify the appropriate macro in the BSP header file. For example, the Vybrid microprocessor integrates two USB controllers which are both EHCI compatible. In the mqx\source\bsp \twrvf65gs10_a5\twrvf65gs10_a5.h file you can find:

```
#define USBCFG_DEFAULT_HOST_CONTROLLER (&_bsp_usb_host_ehci0_if)
#define USBCFG_DEFAULT_DEVICE_CONTROLLER (&_bsp_usb_dev_ehci0_if)
```

To change the MQX RTOS default controller used by applications, modify the commands as shown in this example:

```
#define USBCFG_DEFAULT_HOST_CONTROLLER (&_bsp_usb_host_ehci1_if)
#define USBCFG_DEFAULT_DEVICE_CONTROLLER (&_bsp_usb_dev_ehci1_if)
```

NOTE

After this change, fully rebuild the BSP package.

7 Task aware debugging plug-in

MQX Task Aware Debugging plug-in (TAD) is an optional extension to a debugger tool which enables easy debugging for multi-tasking applications. It helps to visualize internal MQX data structures, task-specific information, I/O device drivers, and other MQX context data.

The MQX TAD is available for these platforms:

- Kinetis Design Studio
- Eclipse-based CodeWarrior Development Studio version 10
- IAR Embedded Workbench for ARM versions 5 and newer
- ARM Keil µVision4 version 4.22 and newer
- ARM DS-5 version 5.14.1 and newer
- Lauterbach (TAD plug-in available directly from Lauterbach)

NOTE

TAD plug-in communicates with the debugger and works with MQX data obtained directly from the RAM memory. A typical debugger session is configured in a way that the code execution stops at the first breakpoint in function main(). In this function, MQX internal data structures are not yet fully initialized and the TAD functionality is significantly reduced. All TAD features are available only when execution stops at a breakpoint in the application task code.

7.1 Kinetis Design Studio IDE

For detailed information about setting up the TAD, see *Getting Started with Kinetis Design Studio IDE and Freescale MQX(tm) RTOS* (document MQXKDSUG). The document is included in the MQX installation as the `<mqx_install_dir>/doc/tools/iar/MQX_KDS_Getting_Started.pdf`.

7.2 Debugging in CodeWarrior development studio version 10

These steps describe how to debug an MQX application:

1. Import (open) the application project in to CodeWarrior workspace by using the File / Import / General / Existing Project menu.
2. Build project using the Project / Build Project menu.
3. Open “Debug Configurations” settings using the Run/Debug Configurations menu and select target you want to debug in the CodeWarrior Download tree.

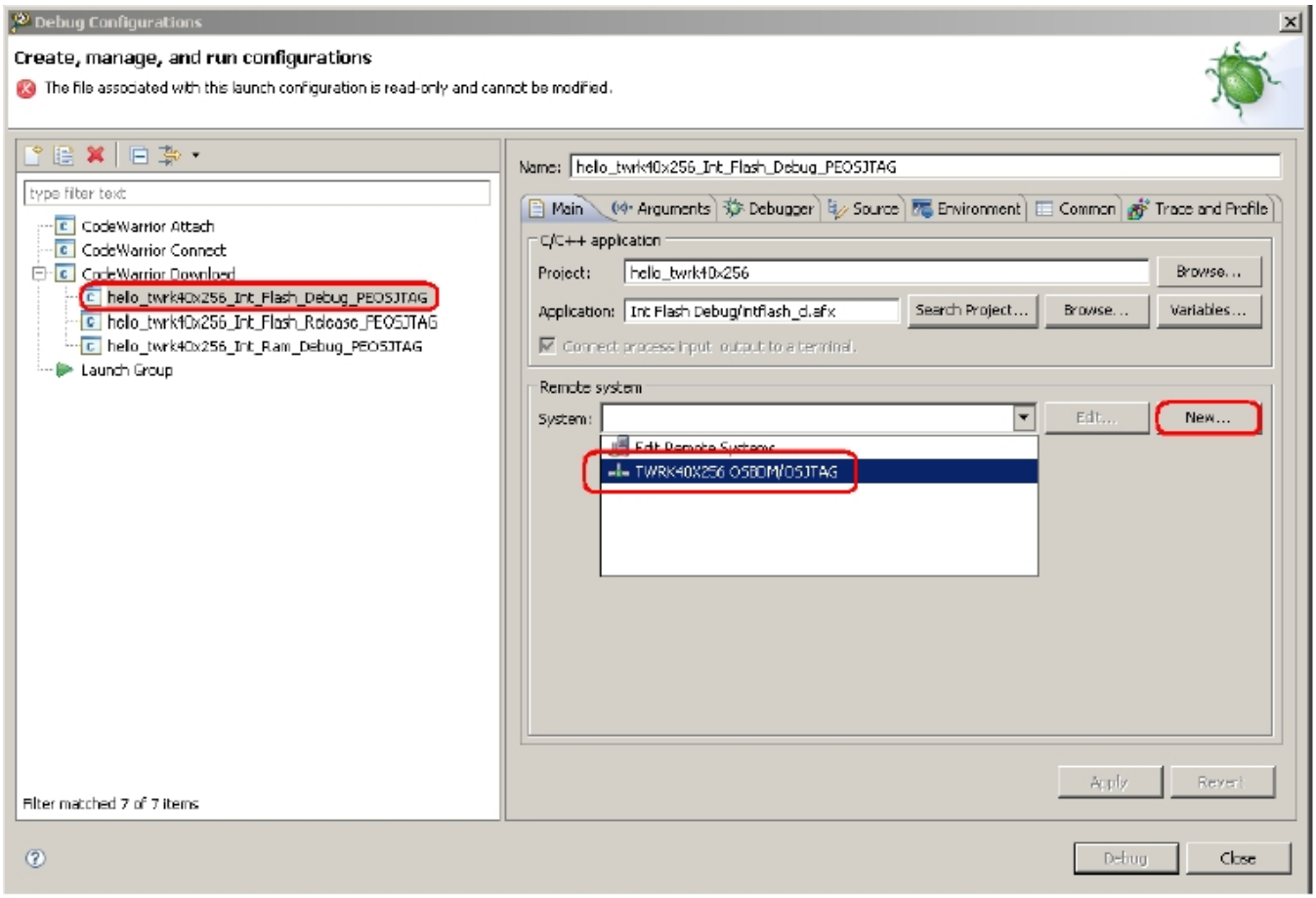

Figure 9. Debug configurations - CodeWarrior

4. Select connection type in the Remote system list. If your connection is not available in the list, define a new one by using New... menu
   - Select Hardware and Simulator, Connection name, and System type. In System tab specify Initialize target: as \lib\<board>.cw10\bsp\<target>\dbg\init_kinetis.tcl and Memory configuration: as \lib\<board>.cw10\bsp\<target>\dbg\<board>.mem
5. Click the Debug button in Debug Configurations Window. The CodeWarrior is switched to a debug perspective and stops the program in the main() function.

7.3 CodeWarrior 10 task aware debugger plug-in

Freescale MQX RTOS introduces a new version of Task Aware Debugger Plug-in (TAD) for CodeWarrior 10 Development Studio.

Installing CodeWarrior 10.3 (and newer) TAD and New Project Wizard Plug-ins

Both TAD and New Project Wizard plug-ins can be installed directly from CW10.3 IDE by using Help\Install New Software menu.

Installing CodeWarrior 10.1 & 10.2 TAD and New Project Wizard Plug-ins

TAD plug-in DLL is installed into the selected CodeWarrior tool automatically during Freescale MQX RTOS setup process. If the plug-in is not properly installed, for example, to a newly installed CodeWarrior studio, perform these steps to install TAD manually:

1. Close The CodeWarrior 10 IDE

2. Locate the tools\codewarrior_extensions\CW MCU v10.x directory in the Freescale MQX RTOS installation folder (by default C:\Program Files\Freescale\Freescale MQX x.y)
3. Navigate to MQX RTOS <install dir>\tools\codewarrior_extensions\CW MCU v10.x directory
4. Open the command-like console and execute the command: install_cw10_plugin.bat<CW10 install dir>
5. Note that the typical CodeWarrior 10 installation folder is C:\Freescale\CW MCU v10.x.
6. Re-start the CodeWarrior 10 IDE.

Enabling the TAD Functionality When Debugging Your MQX Application Project:

1. Open “Debug Configurations” settings of the application project by selecting the Run / Debug Configurations menu. In the Debugger Configuration panel, select proper Launch Configuration.
2. For selected Launch Configuration, go to the “Debugger” tab and activate the “OS Awareness” sub-tab.
3. In the “Target OS” drop-down list box, select the MQX OS for the target platform.

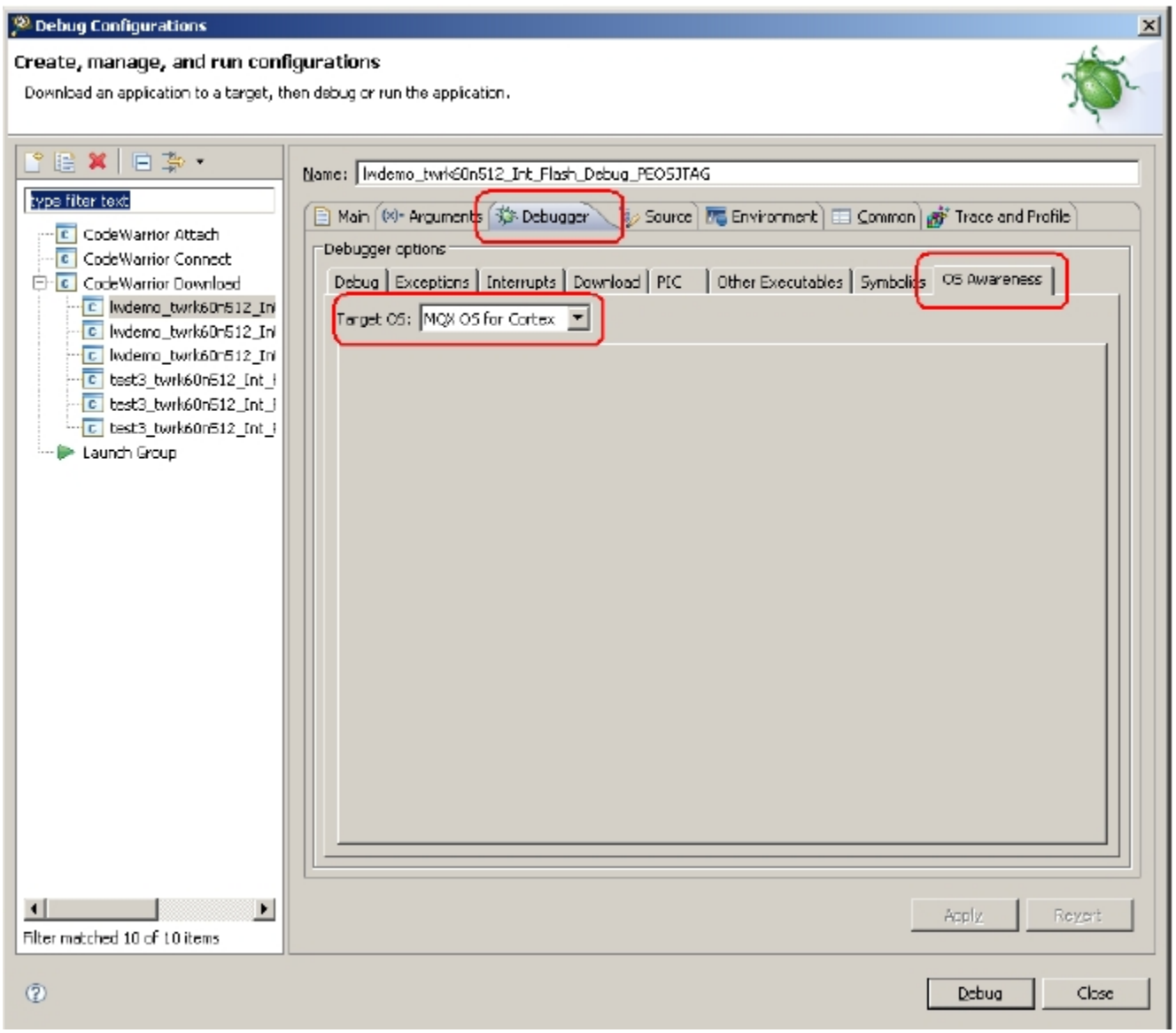

Figure 10. Debug configurations - TAD plugin

4. All example applications coming with Freescale MQX RTOS are already configured for the MQX OS Awareness.

CodeWarrior 10 TAD Features

The MQX plug-in implements the System Browser window showing all running MQX tasks. Open the “Show View” dialog by selecting the Window / Show View / Other… menu. In the “Show View” tree view select the “System Browser” item in the Debug tree. Double-click any task entry in this view to activate the task in the CodeWarrior 10 debugger.

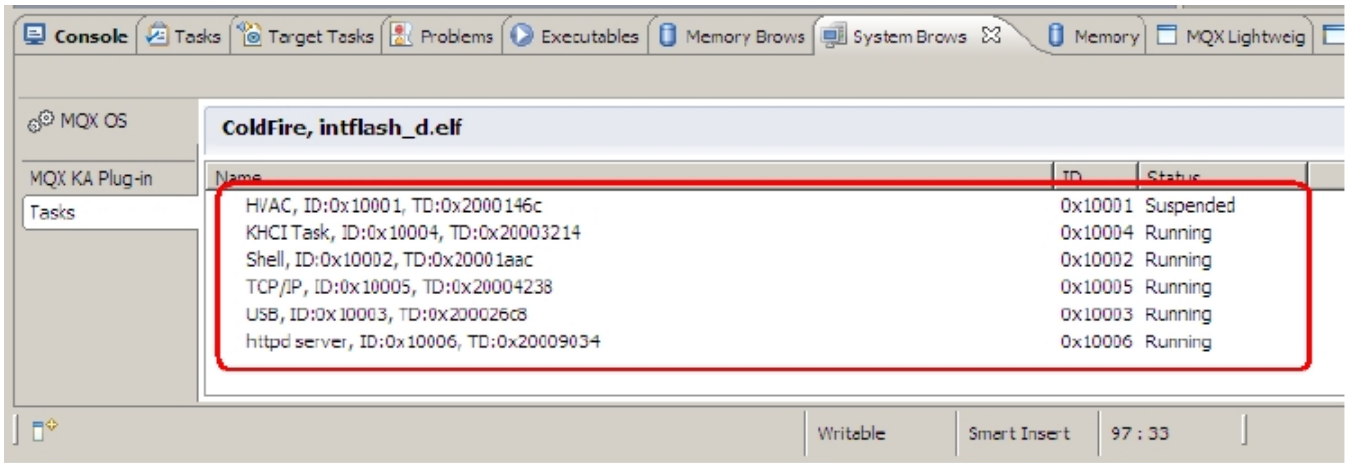

Figure 11. Show view window

The MQX Task Summary screen is available in the MQX / Task Summary menu.

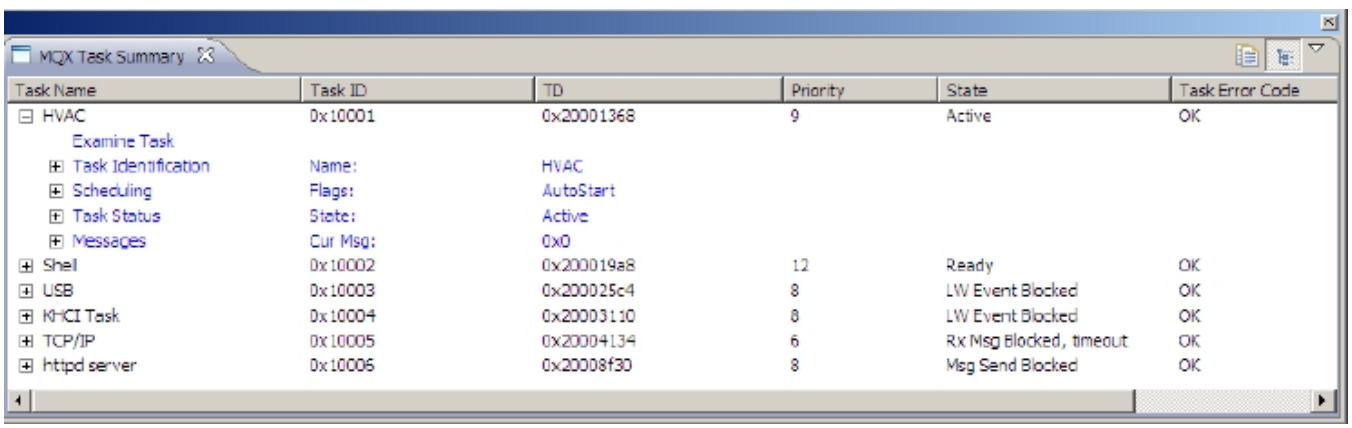

Figure 12. Task summary menu

The MQX Stack Usage screen is available in the MQX / Stack Usage menu.

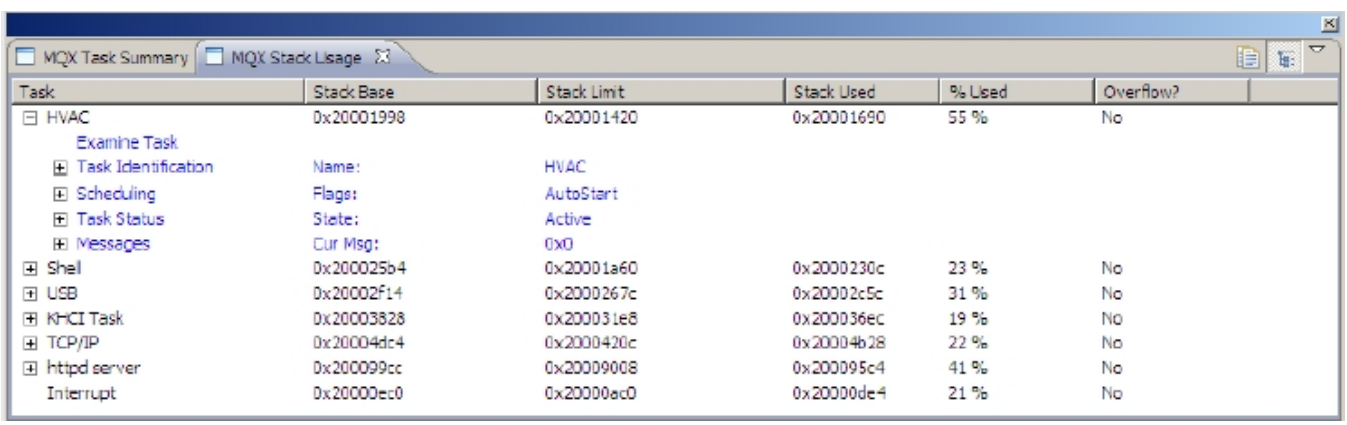

Figure 13. Task usage

The MQX Memory Block Summary screen is available in the MQX / Lightweight Memory Blocks menu.

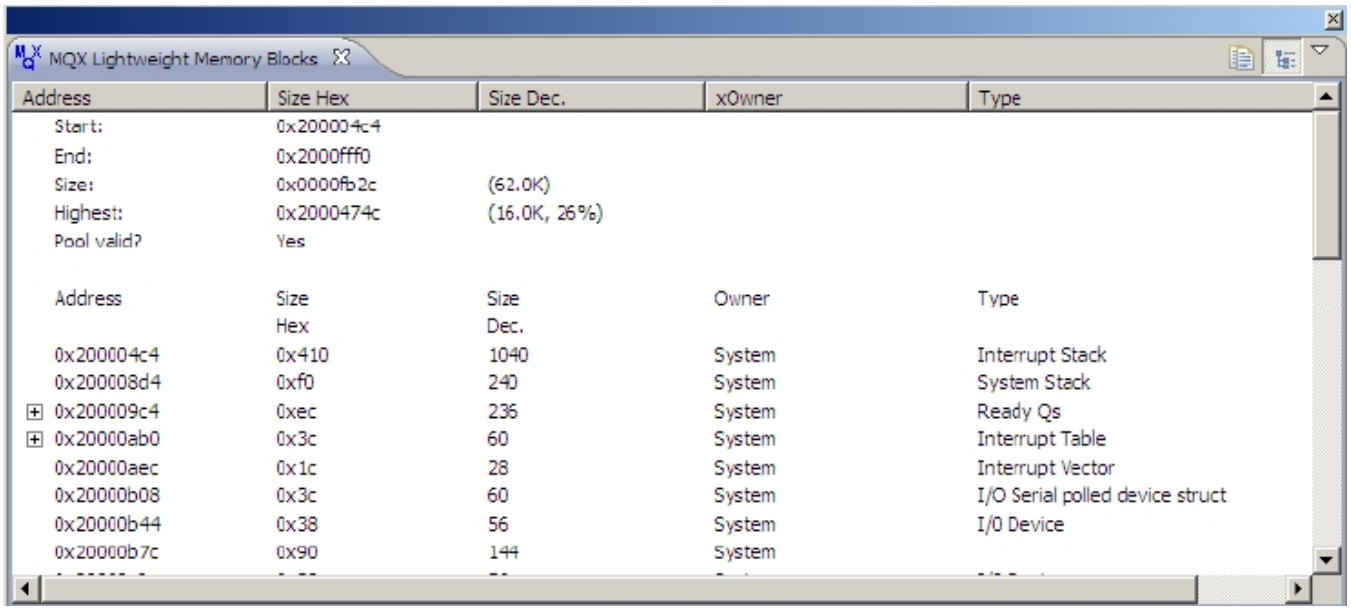

Figure 14. Lightweight memory blocks

7.4 IAR Embedded Workbench for ARM and ColdFire

TAD is currently available for these IAR Embedded Workbench CSpy Debugger versions:

- IAR EWARM version 6.x
- IAR EWCF version 5.3

For detailed information about setting up the TAD, see *Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench®* (document MQXGSIAR). The document is included in the MQX installation as the <mqx_install_dir>/doc/tools/iar/MQX_IAR_Getting_Started.pdf.

7.5 ARM Keil µVision4

TAD viewer is currently available for ARM Keil µVision4 Debugger. For detailed information about setting up TAD, see *Getting Started with Freescale MQX™ RTOS and MDK-ARM Keil µVision4®* (document MQXGSKEIL). The document is included in the MQX installation as the <mqx_install_dir>/doc/tools/uv4/MQX_uVision4_Getting_Started.pdf.

7.6 ARM DS-5

TAD plug-in is distributed separately from MQX release and is available directly from ARM Ltd. For detailed documentation, contact your ARM distributor. For more information about setting up the DS-5 IDE, see *Getting Started with Freescale MQX™ RTOS and Development Studio 5 (DS-5™)* (document MQXGSDS5). The document is included in the MQX installation as the <mqx_install_dir>/doc/tools/ds5/MQX_DS5_Getting_Started.pdf.

8 Integrating Processor Expert Drivers into MQX BSP

The Processor Expert tool, which is available in CodeWarrior 10, enables configuration and driver code generation by using a graphical user interface. The special RTOS Adapter component in Processor Expert and updated BSP code enable integration of the Processor Expert drivers into the BSP library. The Processor Expert drivers can be used by the MQX application in the same way as other BSP drivers. Currently, this integration is supported only for Freescale Kinetis platforms.

Processor Expert Logical Device Drivers (LDD), which is available for the Kinetis family, can be added into “PE ready” BSPs and used in the end user application.

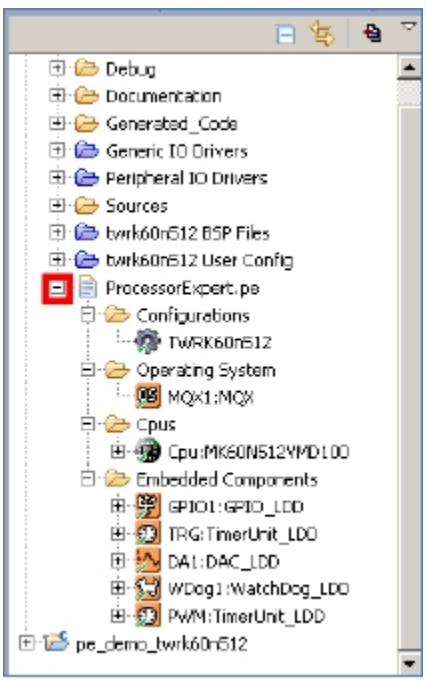

Figure 15. Processor Expert drivers

For more details about Processor Expert, see Processor Expert User Guide which can be found in:
<CW_MCU_10_Install_Directory>/MCU/Help/PDF/ProcessorExpertHelp.pdf

8.1 Processor Expert-ready BSPs

All Kinetis BSPs can host the Processor Expert components. For MQX RTOS version 3.8.1 and later, the special Processor Expert versions of the BSPs were removed and the functionality made available in the standard BSP projects.

The BSP projects have the ProcessorExpert.pe file which contains this:

- Pre-configured CPU component
- MQX RTOS Adapter component
- Set of peripheral components needed by the pe_demo example application

Other components may be added to the BSP project as needed.

8.2 Processor Expert MQX demo application

The Demo application which describes the integration of Processor Expert drivers is available here:

```
<install_dir>/demo/pe_demo/cw10/pe_demo_<board>/.project
```

The application executes these tasks:

1. The PWM signal is generated by using the FlexTimer FTM0 Channel 0. It can be observed by scope on PWM0 - A40 pin on TWR-ELEV FUNCTIONAL or TWR-PROTO board.
2. To signal that the application is running, it toggles LEDs (D9-D11) on the board by using the GPIO driver.

For detailed information about demo application, importing and building the MQX library projects and debugging MQX application in CodeWarrior 10, see the *CW for Microcontrollers v 10 and MQX™ RTOS* (document MQXCWPP) located: `doc\tools\cw\MQX_in_CW_10_x.pdf (Windows menu: Tools Documentation \CodeWarrior\Getting Started CW for Microcontrollers v10.x and MQX)`

8.3 Processor Expert and default clock settings in Kinetis BSPs

The MQX RTOS 3.8 introduces the low power management features on Kinetis platforms. The main features of this solution are the Low Power Manager and Clock Manager modules. The Clock Manager allows runtime switching between clock configurations statically defined at the BSP level.

Three predefined clock configurations are available for Kinetis-based platforms as follows:

- Default board frequency, processor is in normal run mode (MGG PEE mode)
- 12 MHz normal run mode (MCG PEE mode – used also for auto-trimming the internal oscillator)
- 2 MHz low power run mode (MCG BLPI mode)

The code to set the clock configuration in the Kinetis BSPs is generated by the Processor Expert tool and it is not easy to change manually (see bsp_cm.c and .h files). Use the Processor Expert to generate new or change existing clock configurations.

For more information, see *How-to Change Default Clock Settings in Kinetis BSPs* (document MQXGSCLKBSP). The document is included in the MQX RTOS installation as the <mqx_install_dir>/doc/tools/cw/MQX_Howto_SetupKinetisClock_UsingPE.pdf.

9 Board-specific Information Related to the MQX RTOS

This section provides more details about all boards and BSPs supported by the current MQX distribution.

All jumper and other hardware switches, not specifically described below, are assumed to be in factory-default positions. See board user guides for default settings.

9.1 AutoEVB Vybrid

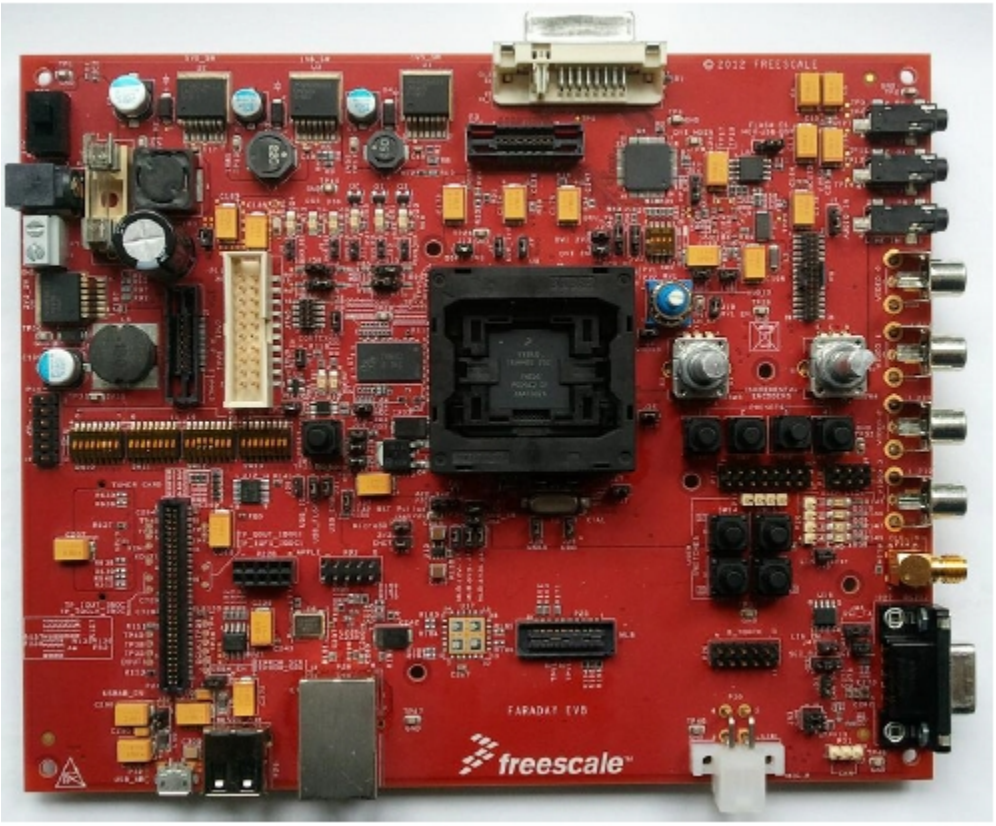

Figure 16. AutoEVB Vybrid

Table 1. ARM Cortex-A5 core - twrvf65gs10_a5 BSP

Core Clock	396 Mhz	-
Bus Clock	132 Mhz	-
Default Console	ttyb:	Default settings: USB Virtual serial port on OpenSDA debug interface. If the OpenSDA is programmed to CMSIS DAP functionality, change the default channel to ttyc: (the RS-232 port on TWR SER board or change J23 and J24 jumper settings according to this information.)
BSP Timer	Global Timer	-

Table 2. ARM Cortex-M4 core - twrvf65gs10_m4 BSP:

Core Clock	132 Mhz	-
Bus Clock	66 Mhz	-
Default Console	ttyc:	RS-232 port on TWR-SER or TWR-SER2 boards
BSP Timer	Systick	-

Important Jumper Settings (Rev. C):

- For boot from fuses: J15 1-2, J18 1-2
- For boot from RCON sw: J15 2-3, J18 1-2
- To use Ethernet on the AutoEVB board, these settings are required for DIP switches: SW12 switches 3, 4, and 5 in ON position.
- For Quad-SPI Flash boot SW10, SW11, SW12, SW13 all in OFF positions , J15 2-3,
- J18 1-2
- For SD card boot SW12, SW13 all in OFF positions, SW10 - switches 6 and 7 in ON position, SW11 - switch 4 in ON position, J15 2-3, J18 1-2

Known Issues:

- The board does not have the SPI connected out from the processor. The SPI driver is not supported.
- SAI driver example fails a few seconds after recording. The problem will be fixed in the next MQX RTOS release.
- SD Card detection is not supported on AutoEVB boards Rev.C. The board does not have a Hardware connection to detect the insertion of the SD card. To properly run the SD card demo, ensure that the card is in the slot before the program attempts to access it, for example, before power on reset.

Debugging and Tool Chain Related Information:

IAR + J-Link/I-Jet probe debugging – Ensure that you have the J-Link or I-Jet probe updated to the latest firmware because the older versions of firmware do not fully support Vybrid multicore debugging. See *Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench®* (document MQXGSIAR) (<install_dir>/doc/tools/iar/MQX_IAR_Getting_Started.pdf) for details about Vybrid and multicore debugging.

Board-Specific Build Targets:

A5 Boot Loader target – target intended for application (boot loader) directly booted from BootROM. Code and data are located in the OCRAM (SRAM) memory so that it does not clash with BootROM. The target is used in the Vybrid dual-core boot loader application `mqx/examples/bootloader_vybrid`.

- A5 DDR target –target using DDR memory for both code and data. Load this target by using either U-Boot or MQX RTOS boot loader.
- A5 Int_Ram target – target using internal OCRAM (SRAM) memory for both code and data. Load this target by either U-Boot or MQX boot loader.
- M4 Int_Ram target – Code is located in the lower part of OCRAM (SRAM) and data is in the TCM memory. Load this target either by U-Boot or MQX boot loader.

Known Issues

- Both A5 and M4 Int_Ram target code space is limited and most of the RTCS-based applications are too large to fit to the memory.

9.2 TWR-VF65GS10

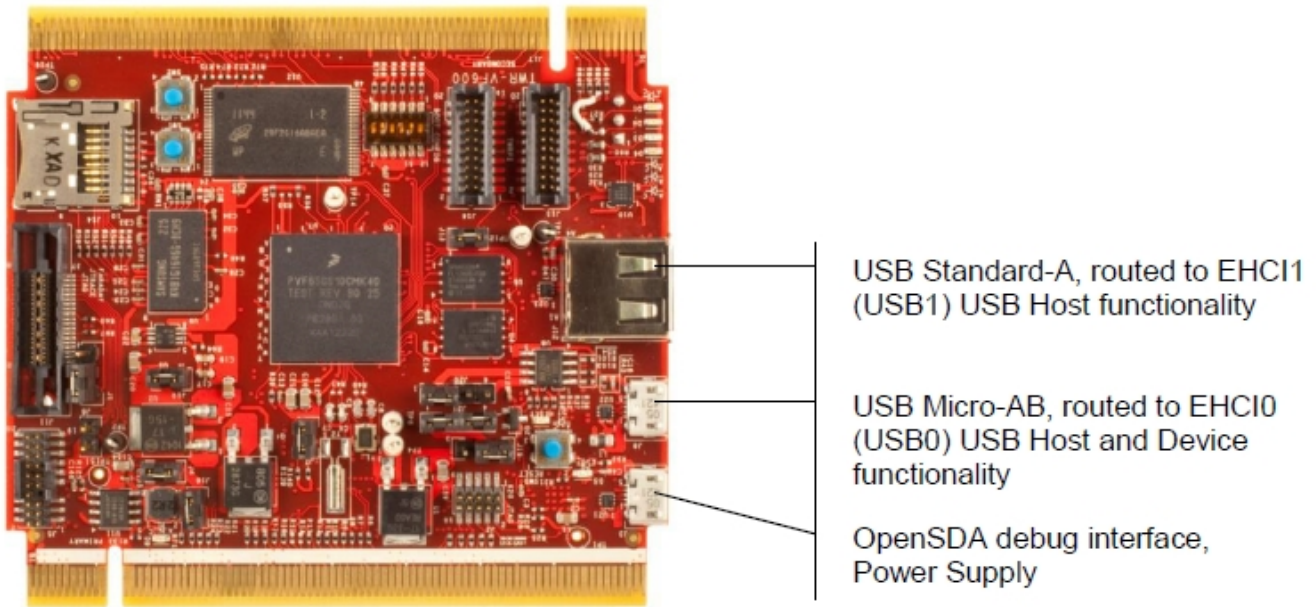

Figure 17. TWR-VF65GS10

Table 3. ARM Cortex-A5 core - twrvf65gs10_a5 BSP

Core Clock	396 Mhz	-
Bus Clock	132 Mhz	-
Default Console	ttyb:	Default settings: USB Virtual serial port on OpenSDA debug interface. If the OpenSDA is programmed to CMSIS DAP functionality, change the default channel to ttyc: (the RS-232 port on TWR SER board or change J23 and J24 jumper settings according to this information.)
BSP Timer	Global Timer	-

Table 4. ARM Cortex-M4 core - twrvf65gs10_m4 BSP:

Core Clock	132 Mhz	-
Bus Clock	66 Mhz	-
Default Console	ttyc:	RS-232 port on TWR-SER or TWR-SER2 boards
BSP Timer	Systick	-

Important Jumper Settings (Rev. G):

- Ensure that board is powered on the processor board by using the Dual USB power cable. Do not use power source from J5 of the primary elevator board.

Serial Port Settings

- Default: J23 (1-3 & 2-4) and J24 (1-3 & 2-4)
- Vybrid SCI1 (ttyb:) to OpenSDA (K20)
- Vybrid SCI2 (ttyc:) to Elev UART1 (TWR-SER)
- Alternative 1: J23 (1-2 & 3-4) and J24 (1-2 & 3-4)
- Vybrid SCI1 (ttyb:) to Elev UART1 (TWR-SER)
- Vybrid SCI2 (ttyc:) to OpenSDA (K20)
- Alternative 2: J23 (2-4) and J24 (2-4)
- Vybrid SCI1 (ttyb:) to Elev UART0 (0Rs connected)
- Vybrid SCI2 (ttyc:) to Elev UART1 (TWR-SER)

USB0 (EHCI0) Module

- Default BSP setting – use USB0 port `#define USBCFG_DEFAULT_DEVICE_CONTROLLER (&_bsp_usb_dev_ehci0_if)#define USBCFG_DEFAULT_HOST_CONTROLLER (&_bsp_usb_host_ehci0_if)` in the `<mqx_install_dir>\mqx\source\bsp\twrvf65gs10_a5\twrvf65gs10_a5.h`
- USB0 connector setting (TWR-SER USB Mini-AB vs. on-board USB Micro-AB connector)
- TWR-VF65GS10 board – J22 11-12 open for external TWR-SER USB Mini-AB port
- TWR-VF65GS10 board – J22 11-12 closed for on-board Micro-B (J8) USB connector
- USB0 host mode
- TWR-VF65GS10 board – J19 2-3
- TWR-VF65GS10 board – J20 1-2 and 3-4
- USB0 device mode
- TWR-VF65GS10 board – J20 2-3
- TWR-VF65GS10 board – J21 2-3

USB1 (EHCI1) Module

- To enable USB1 (EHCI1), change the BSP settings as follows and recompile BSP. The USB1 (EHCI1) is connected to USB Standard-A. `#define USBCFG_DEFAULT_DEVICE_CONTROLLER (&_bsp_usb_dev_ehci1_if)#define USBCFG_DEFAULT_HOST_CONTROLLER (&_bsp_usb_host_ehci1_if)` in the `<mqx_install_dir>\mqx\source\bsp\twrvf65gs10_a5\twrvf65gs10_a5.h`
- USB1 host mode
- TWR-VF65GS10 board – J19 2-3
- TWR-VF65GS10 board – J21 1-2 and 3-4

Known Issues

- A freeze bit in the MCR register allows the PIT timers to stop when the device enters debug mode. When either core is stopped by a debugger, the PIT timer stops too. For the multicore debugging, this process may cause confusion.
- Both ARM Cortex-A5 and ARM Cortex-M4 Int_Ram target code space is limited and most of the RTCS-based applications are too large to fit to the memory.

Debugging and Tool Chain Related Information:

IAR + J-Link/I-Jet probe debugging – Ensure that either the J-Link or I-Jet probe is updated to the latest firmware because older versions of firmware do not fully support the Vybrid multicore debugging). See *Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench®* (document MQXGSIAR) (<install_dir>/doc/tools/iar/MQX_IAR_Getting_Started.pdf) for details about Vybrid and multicore debugging.

- The OpenSDA circuit is programmed to MSD/CDC functionality by default. This provides the serial-to-USB conversion on UART1(ttyb) and other functionalities. The circuit can be also used as the debugging probe (CMSIS-DAP) for ARM DS5 IDE. Contact your ARM distributor for more details.
- This release was tested with TWR-VF65GS10 Rev. G. The older revisions of the boards have different default console setting. If using older board revisions, adjust the configuration according to the Hardware requirements.
- Note that CMSIS-DAP firmware does not support serial-to-USB conversion. Change the default serial channel to TWR-SER board (ttyc:) in Cortex-A5 BSP. Avoid using both cores simultaneously in this setting because both cores share the same serial channel.

Default MMU Settings

- MMU (A5 core only) is enabled with Level 1 translation after initialization. The entire memory area is set to read/write access policy with virtual addresses equal to physical addresses. Cache is enabled and turned on for the DDR region with the write back cache policy. After initialization, you can change or create new regions with 1MB or 4kB region granularity and with different types of access policy, cache policy, or memory type by using the _mmu_add_vregion() function.

Board-Specific Build Targets:

A5 Boot Loader target – target intended for application (boot loader) directly booted from BootROM. Both code and data are located in the OCRAM (SRAM) memory. The target is used in the Vybrid dual-core boot loader application mqx/examples/bootloader_vybrid.

- A5 DDR target –target using DDR memory for both code and data. Load this target by using either U-Boot or MQX boot loader.
- A5 Int_Ram target – target using internal OCRAM (SRAM) memory for both code and data. Load this target by using either U-Boot or MQX boot loader.
- M4 Int_Ram target – Code is located in the lower part of OCRAM (SRAM) and the data is in the TCM memory. Load this target by using either U-Boot or MQX boot loader.
- See Chapter 2.3 “Library Build Targets” for more details about the standard build targets.

Known Issues

- Both A5 and M4 Int_Ram target code space is limited and most of the RTCS-based applications are too large to fit to the memory.

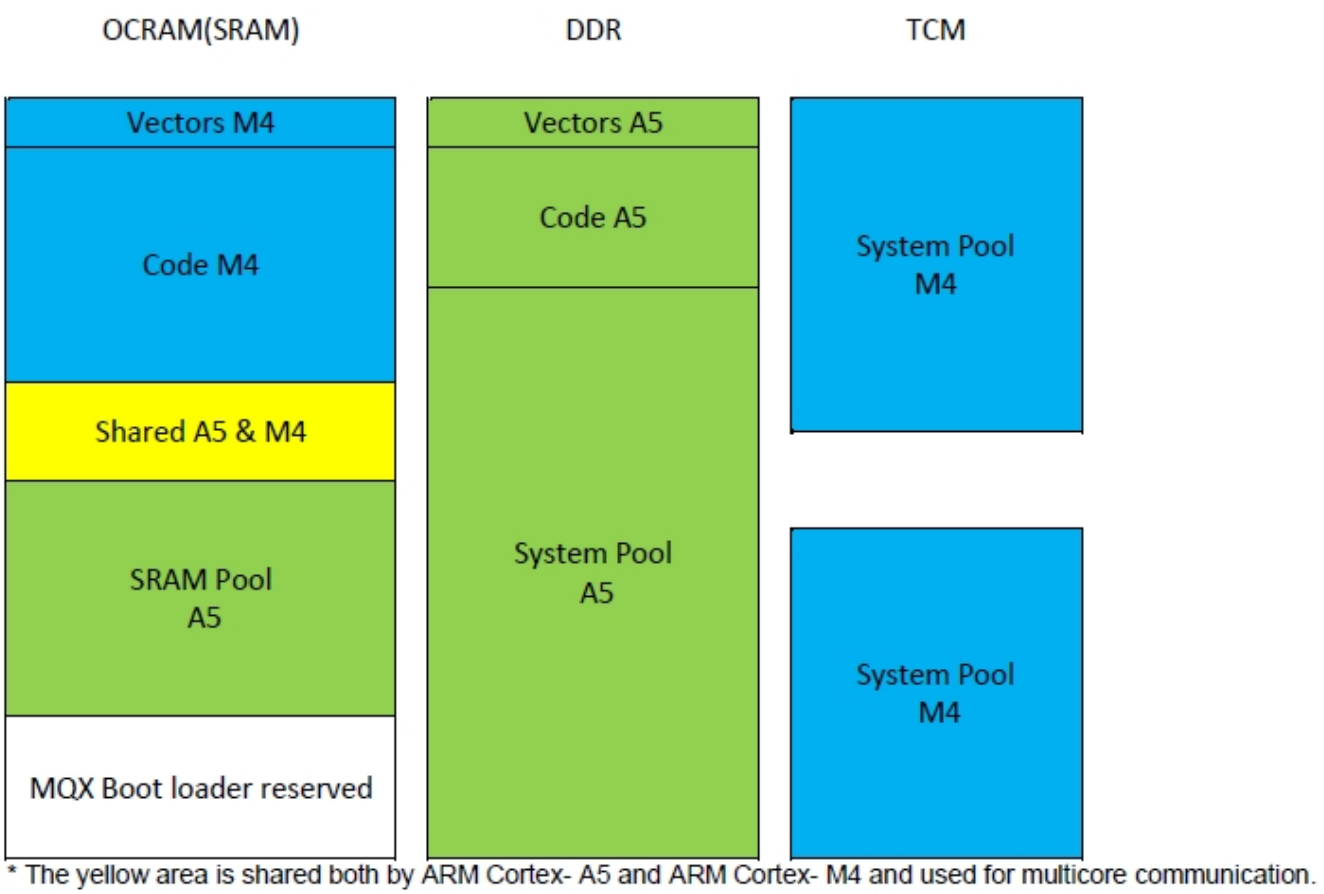

Figure 18. M4-Int_Ram (blue) + A5-DDR target (green) memory layout

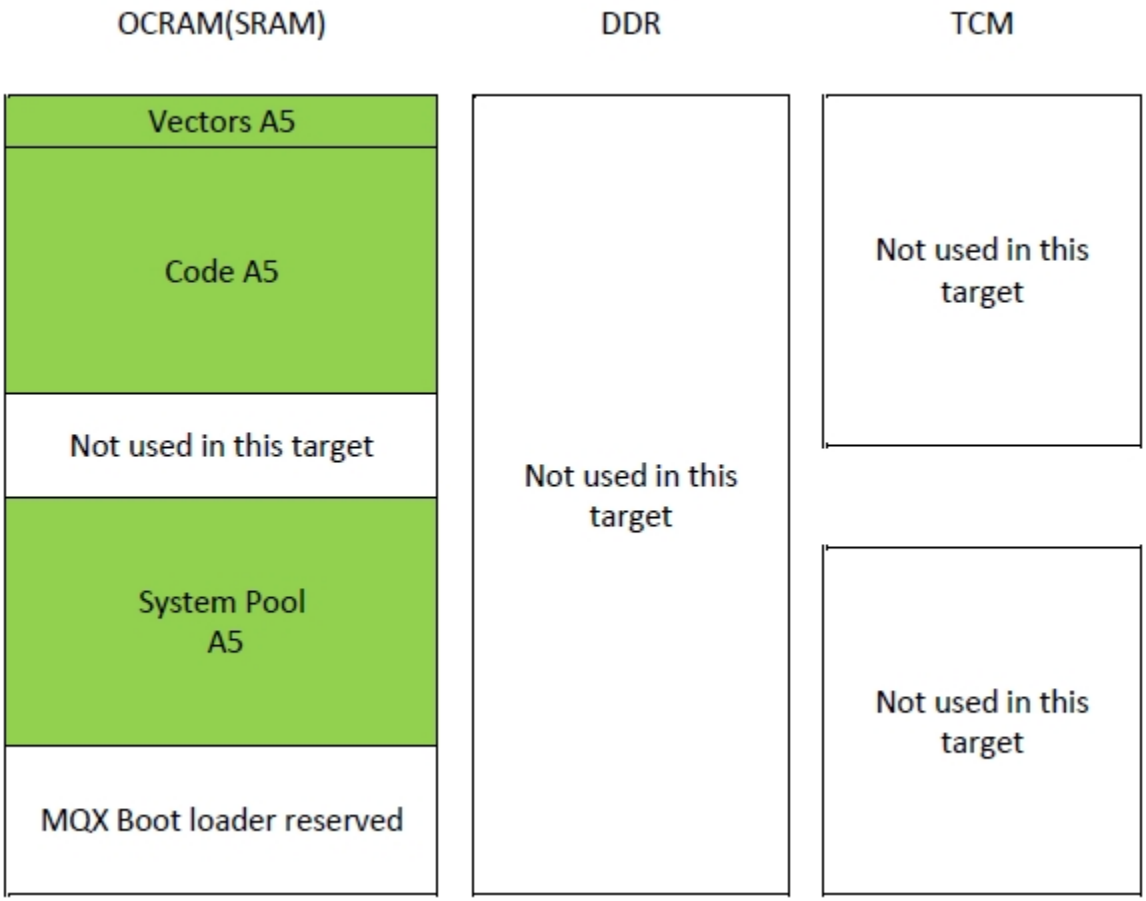

Figure 19. A5 Int_Ram target memory layout

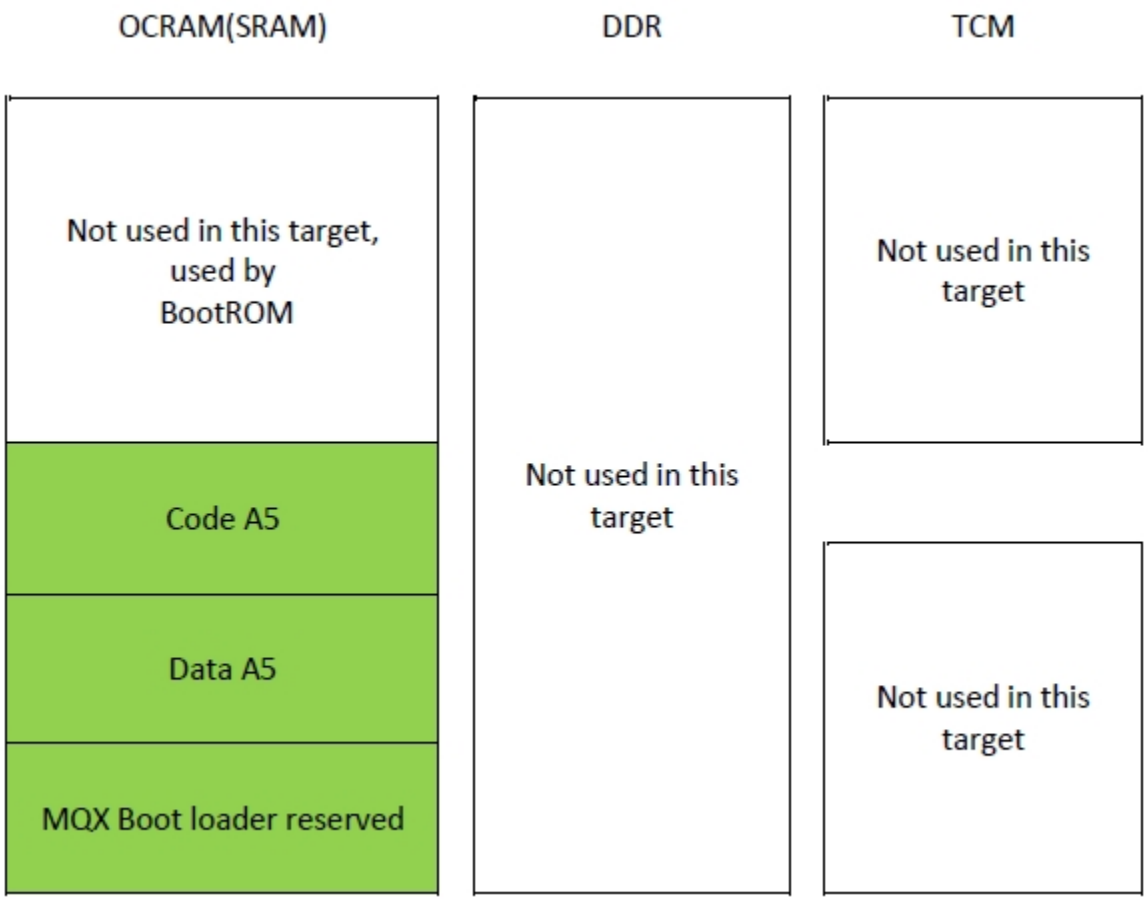

Figure 20. A5 Boot loader

9.3 FRDM-K64F Freescale Freedom platform

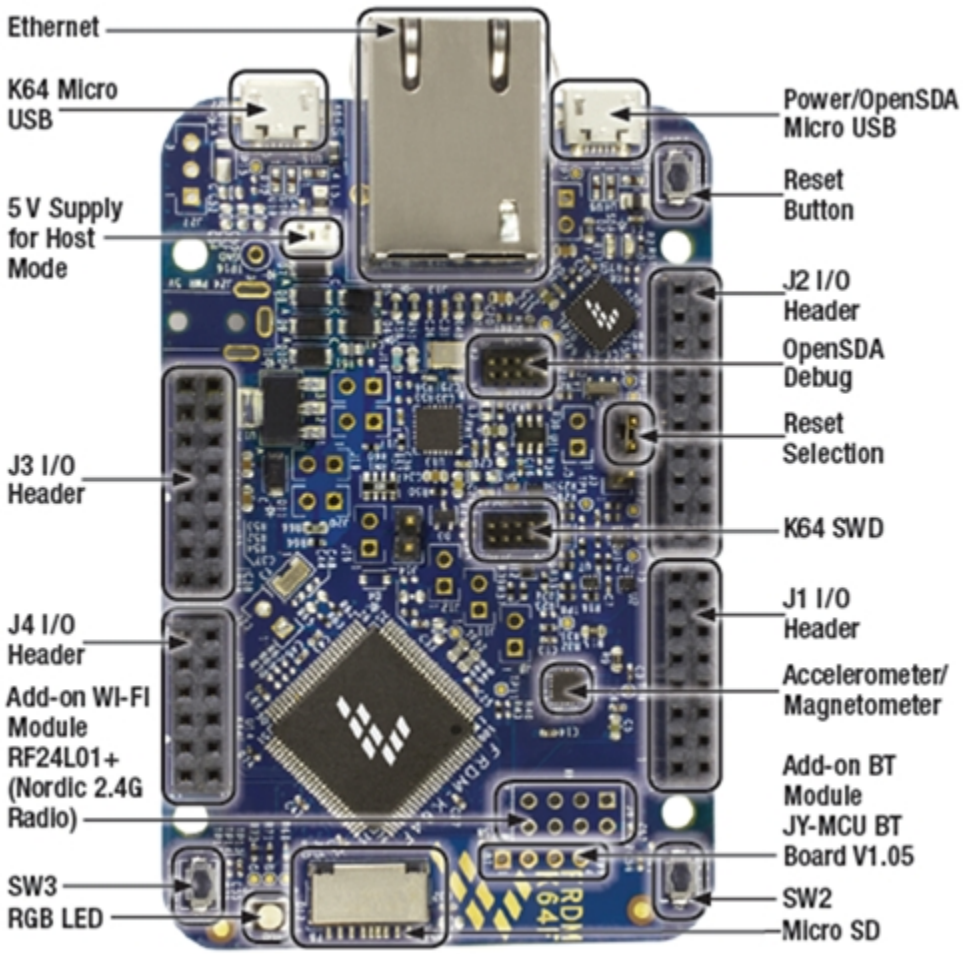

Figure 21. FRDM-K64F

Table 5. FRDM-K64F

Core Clock	120 Mhz	-
Bus Clock	60 Mhz	-
Default Console	ttya:	debugger USB_CDC port, J4 micro USB connector
BSP Timer	Systick	-

Important Jumper Settings (board Rev. C)

- Default jumper settings
  - J25 on position 1-2
  - J14, J17, J18, and J20 open
  - J5, J7, J8, J11, J12, J16, J19, J21, and J23 closed
- For USB Device and USB Host modes:
  - Use the on board J22 micro USB connector and the default jumper settings
- When using the J-Link debugger:
  - Populate the JTAG/SWD CONNECTOR J9, and cut off the J8 and J12 jumpers

Known Issues:

- When running a USB host-based application, such as mfs_usb or web_hvac, the application console (debugger USB_CDC) stops working when a USB device is attached to the J22 micro USB connector. The application itself still works properly. To avoid this board issue, plug in the USB device before the application starts.
- When using default onboard CMSIS-DAP Debugger with Keil, the debugger connection port must be changed from JTAG to SW. Otherwise, starting a debug session will return the RDDI-DAP Error. To change the debugger port setting select your project from WorkSpace and go to menu Project/Options or use the Alt+F7 shortcut, in the Debug click tab on CMSIS-DAP Settings button and then change JTAG to SW from Port menu.

9.4 TWR-K20D50M

Figure 22. TWR-K20D50M

Table 6. TWR-K20D50M

Core Clock	48 Mhz	-
Bus Clock	48 Mhz	-
Default Console	ttyb:	OSJTAG- USB mini connector
BSP Timer	Systick	-

Important Jumper Settings:

- For using USB Device mode, jumpers on position:
  - TWR-K20D50M board - J26 open
  - TWR-K20D50M board - J30 on position 5-6 (VREGIN)
- For using USB Host mode, jumpers on position:

  - TWR-K20D50M board - J26 on position 1-2
  - TWR-K20D50M board - J30 on position 5-6 (VREGIN)
- For using RTC example without a battery:
  - TWR-K20D50M board - J35 on position 1-2

Known Issues:

- The default console “ttyb:”, which is used by the OSJTAG, is also routed to the RS232 TWR-SER interface. This sometimes leads to conflicts.
- Timer interrupt wakeup from LLS power mode leads to the chip reset with the reset cause set to core lockup.
- The switch to VLPR power mode does not work. The chip does not acknowledge the power mode change in the PMSTAT register.

Other Notes:

- USB is only available on the TWR-K20D50M board. It is not routed to the port on the TWR-SER board.
- Example projects are provided only with configurations which have to be run from the Internal Flash Memory because of the small RAM size available on the K20D50 processor.

9.5 TWR-K20D72M

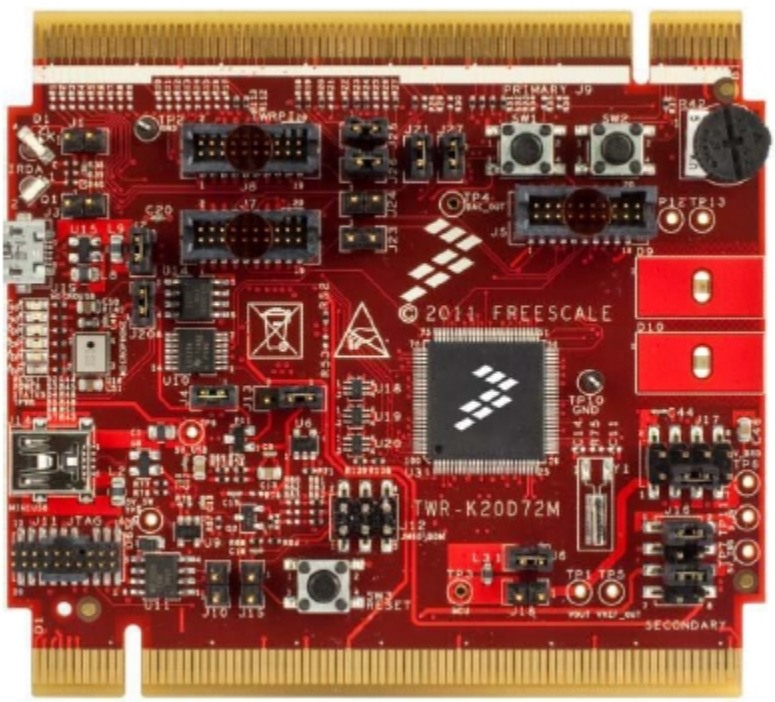

Figure 23. TWR-K20D72M

Table 7. TWR-K20D72M

Core Clock	72 Mhz	-
Bus Clock	36 Mhz	-

Table continues on the next page...

Table 7. TWR-K20D72M (continued)

Default Console	ttyb:	OSJTAG- USB mini connector
BSP Timer	Systick	-

Important Jumper Settings:

- On board connector does not work as Host by default. See known issues. For using USB Host mode with TWR-SER board set jumpers to this position:
  - TWR-SER board - J16 on position 1-2 (VB_HOST)
  - TWR-SER board - J10 on default position 1-2 (USB host)
- Use on board connector by default. To use the USB Device mode with TWR-SER board, set jumpers to this position:
  - TWR-SER board - J16 on position 3-4 (VB_DEV)
  - TWR-SER board - J10 on position 2-3 (USB device)
- For SD card operation:
  - TWR-MEM board - jumper J12 on position 3-4 (IRQA)

Known Issues:

- IAR IDE version 6.40 has issues when connecting to the board by using the J-Link probe. To resolve this issue, use OSBDM instead.
- Example projects contain build configurations for code execution which are different than the Flash or RAM memory. The RAM-based execution may be faster to debug. Not all examples, however, fit into RAM and may not link.
- USB host cannot be used with onboard USB connection when led (D9) is used. A shared pin, PTC9, is used to control the led and power supply for USB host on board Rev. B.
- The SCI1 module (mapped to ttyb) and SPI1 module, used as SD card communication channel, share a pin PTE1, which prevents the SD card from working properly. For correct SD card driver operation, the default console has to be changed to ttya. Please note that ttya is available at TWRPI and cannot be used out of the box.

Other Notes:

- The default console interface (ttyb:) is routed to OSBDM-COM (USB mini connector). Use the P&E Microcomputer Systems OSJTAG terminal to access the board serial line.
- Because SCI1 module and SPI1 module share a pin PTE1, useTTYA as a terminal for the SPI1 module.

9.6 TWR-K21D50M

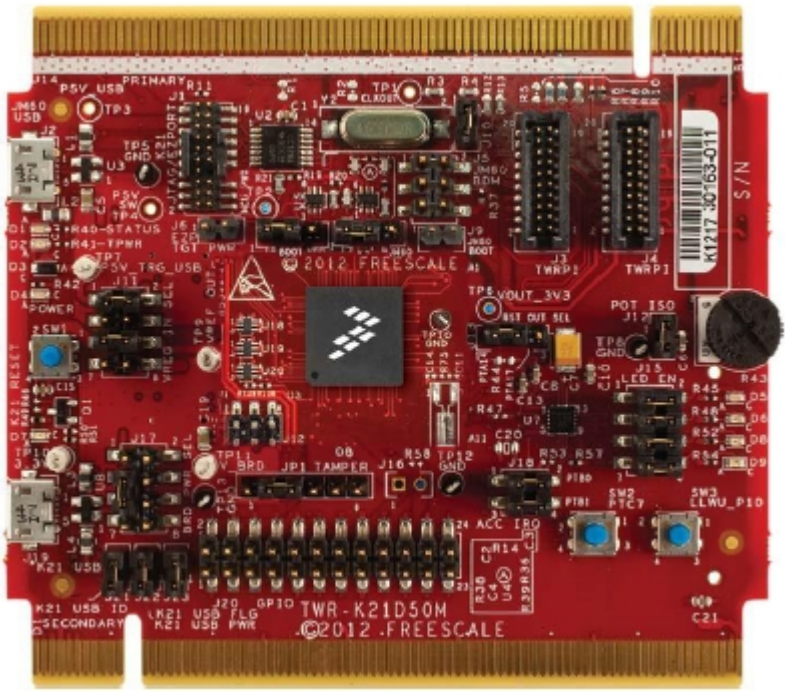

Figure 24. TWR-K21D50M

Table 8. TWR-K21D50M

Core Clock	48 Mhz	-
Bus Clock	48 Mhz	-
Default Console	ttyc:	OSJTAG- USB micro connector
BSP Timer	Systick	-

Important Jumper Settings:

- For using Tower System USB:
  - TWR_K21D50M board - shoot 6-8 on J11
  - Install R224, R226 on nets USB0_DP and USB0_DN and remove R225, R227 on K21_MICRO_USB_DP and K21_MICRO_USB_DN.
- For using Micro USB in K21 tower board:
  - TWR_K21D50M board - shoot 5-6 on J11
  - Leave R225 and R227 on nets K21_MICRO_USB_DP and K21_MICRO_USB_DN and do not populate R224 and R226 on USB0_DP and USB0_DN.
- For using USB Host mode, jumpers on position:
  - TWR-SER board - J16 on position 1-2 (VB_HOST)
  - TWR-SER board - J10 on default position 1-2 (USB host)
- For using USB Device mode, jumpers on position:
  - TWR-SER board - J16 on position 3-4 (VB_DEV)
  - TWR-SER board - J10 on position 2-3 (USB device)
- For using UART0 with Primary Elevator (PTA14 and PTA15), jumper J13 on position 2-3
- For SD card operation:
  - TWR-MEM board - J2 (SD_SEL2): set to position 1-2
  - TWR-MEM board - J3 (SD_CS): set to position 1-2

- TWR-MEM board - J12 (SD_SEL1) jumper on position 3-4 (IRQA)
- TWR-MEM board - J13 jumper installed

Known Issues:

- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.
- K21D50M has issues flashing by using J-Link in CW 10.2.
- ADC and LWADC examples are not working properly because of a hardware design. The ADC0_DPM3 is connected to a potentiometer and used for a different mode only. To run these demos, ADC0_DP3 needs to be connected to the GND.

Other Notes:

- For KEIL ARM compiler, the libraries are pre-compiled for “Release” target only. Compile the “Debug” target before the first use.

9.7 TWR-K21F120M

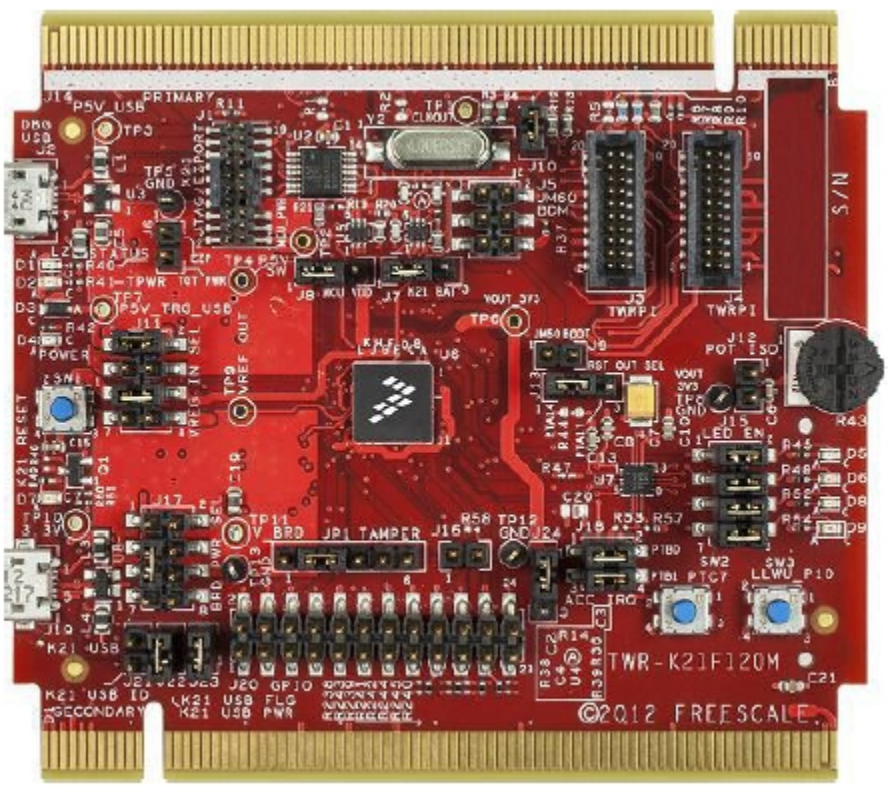

Figure 25. TWR-K21F120M

Table 9. TWR-K21F120M

Core Clock	120 Mhz	-
Bus Clock	60 Mhz	-

Table continues on the next page...

Table 9. TWR-K21F120M (continued)

Default Console	ttyf:	RS-232 port on TWR-SER or OSJTAG Micro-USB connector
BSP Timer	Systick	-

Important Jumper Settings:

For basic operations, ensure that these jumper settings are applied:

- For USB Device mode, use onboard J19 micro USB connector and jumpers on position:
  - WR-K21F120M System module, J24 on position 1-2
  - TWR-K21F120M System module, J22 open
  - TWR-K21F120M System module, J11 on position 5-6 (VREGIN)
- For USB Host mode, use onboard J19 micro USB connector and jumpers on position:
  - TWR-K21F120M System module, J24 on position 1-2
  - TWR-K21F120M System module, J22 on position 1-2 (Enable USB Power)
  - TWR-K21F120M System module, J11 on position 5-6 (VREGIN)
- For IPC example:
  - TWR-K21F120M board, J13 open
- For RAM Disc Example
  - TWR-MEM System module, J12 open 3-4

Known Issues:

- USB Host CDC Serial: USB no longer functions after the CDC device is re-attached.
- J-Link cannot be used to flash CW V10.4GCC to chip. Instead, OS-JTAG must be used.
- FlexBus shares pins with the SPI module. To run the demo by using FlexBus (Ramdisk), disable the SPI module in user_config.h.
- It is impossible to configure SPI1 and SPI2 for the DMA mode on this platform due to RX/TX sharing the same DMA channel. See the Reference Manual for the K21F120M.

Other Notes:

- Because the FlexBus pin conflicts with SPI0 and SAI when running the Ramdisk example, SPI0 and SAI must be disabled in user_config.h.

9.8 TWR-K40D100M

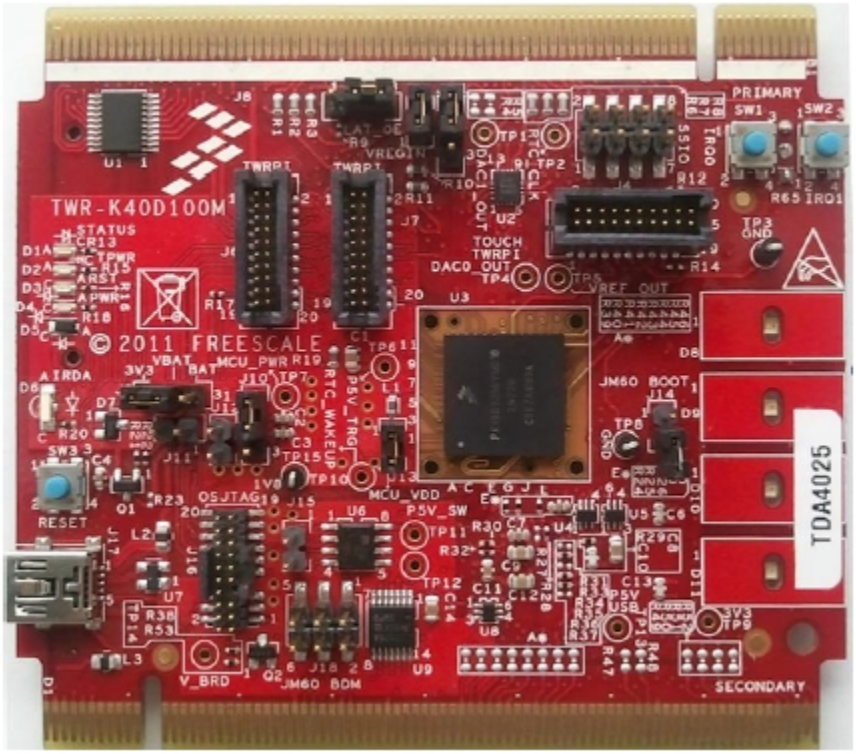

Figure 26. TWR-K40D100M

Table 10. TWR-K40D100M

Core Clock	96 Mhz	-
Bus Clock	48 Mhz	-
Default Console	ttya:	OSJTAG Micro-USB connector
BSP Timer	Systick	-

Important Jumper Settings:

- For using USB Host mode, jumpers on position:
  - TWR-SER board - J16 on position 1-2 (VB_HOST)
  - TWR-SER board - J10 on default position 1-2 (USB host)
- For using USB Device mode jumpers on position:
  - TWR-SER board - J16 on position 3-4 (VB_DEV)
  - TWR-SER board - J10 on position 2-3 (USB device)

Known Issues:

- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.

Other Notes:

- The default console interface (ttya:) is routed to OSBDM-COM (USB mini connector). Use the P&E Microcomputer Systems OSJTAG terminal to access the board serial line.

- To enable the TWR-SER RS232 interface, change the BSP_DEFAULT_IO_CHANNEL configuration option to "ttyd:" in the mqx\source\bsp\twrk40d100m\twrk40d100m.h file.
- To enable the audio example set J4 to 5-6 on the TWR-K40D100M System module (SAI channel 0 / TWR-AUDIO board used).

9.9 TWR-K40X256

Figure 27. TWR-K40X256

Table 11. TWR-K40X256

Core Clock	120 Mhz	-
Bus Clock	60 Mhz	-
Default Console	ttyf:	RS-232 port on TWR-SER or OSJTAG Micro-USB connector
BSP Timer	Systick	-

Important Jumper Settings:

- To use TWR-LCD board with eGUI:
  - TWR-LCD board SW5 - all switches to ON (enable touch screen)
  - TWR-LCD board - SW1 switches depending on usage either SPI (01111110) or 16 bits FlexBus (10111110)
  - TWR-K40X256 – open J7 3-4, 5-6, 7-8
  - TWR-K40X256 – open J14 15-16
  - TWR- K40X256 board to enable navigation buttons open J4 1-2

Known Issues:

- The FlexBus FB_OE_B signal is directly connected to OE pin of the address latch on the TWR-MEM card. This prevents using FlexBus for communication with MRAM and CF-CARD on TWR-MEM card.
- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.

Other Notes:

- The default console interface (ttya:) is routed to OSBDM-COM (USB mini connector). Use the P&E Microcomputer Systems OSJTAG terminal to access the board serial line.
- To enable the TWR-SER RS232 interface, change the BSP_DEFAULT_IO_CHANNEL configuration option to "ttyd:" in the `mqx\source\bsp\twrk40x256\twrk40x256.h file`.

9.10 TWR-K53N512

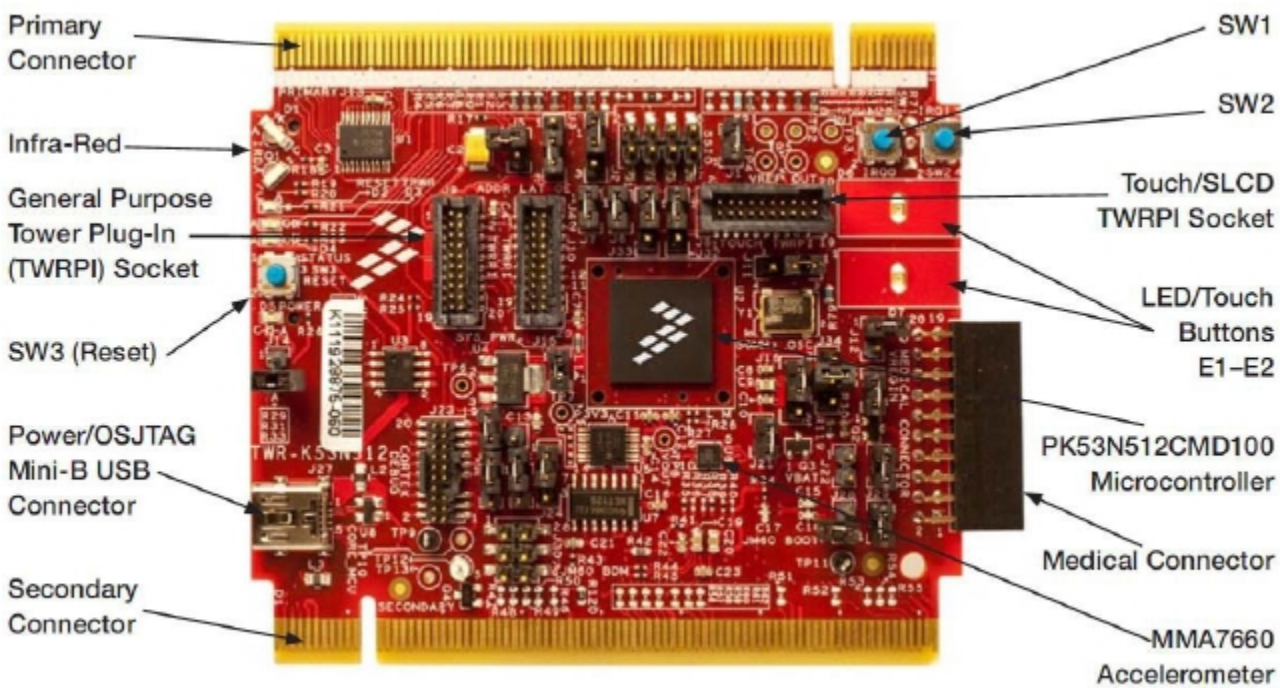

Figure 28. TWR-K53N512

Table 12. TWR-K53N512

Core Clock	96 Mhz	-
Bus Clock	48 Mhz	-
Default Console	ttya:	OSJTAG - USB mini connector
BSP Timer	Systick	-

Important Jumper Settings (board Rev. C)

- For standalone operation (using clock from TWR-K53N512 board):
  - To use 50MHz - Jumper J11 on position 1-2
- To enable Ethernet communication (use with TWR-SER):

    - TWR-K53N512 - Jumper 11 on position 2-3 – processor clock taken from the TWR-SER board. Jumper 32 on position 1-2, jumper 33 on position 1-2
    - TWR-SER CLK_SEL 3-4
    - TWR-SER CLKIN-SEL 2-3 (processor clock is taken from PHY)
    - TWR-SER – ETH-CONFIG J12 9-10 to select RMII communication mode
- Important: Plug both the processor and the serial board (TWR-SER) into the Tower. The processor is using the external clock from the Ethernet PHY on the serial card.
    - For using USB Host mode, jumpers on position:
    - TWR-SER board - J16 on position 1-2(VB_HOST)
    - TWR-SER board - J10 on default position 1-2(USB host)
- For using USB Device mode, jumpers on position:
    - WR-SER board - J16 on position 3-4(VB_DEV)
    - TWR-SER board - J10 on position 2-3(USB device)
- For using RAM disk, jumpers on position:
    - TWR-MEM board - J16 remove
    - TWR-MEM board - J11 remove (default)
- For using TWRPI-SLCD:
    - TWR-K53N512 jumper 32 on position 1-2; jumper 33 on position 1-2

Known Issues

- The TWR-K53N512 BSP has been created for the TWR-K53N512, Rev.C schematics (SCH-26994), Rev.X3 BOM (700-26694). This TWR-K53N512 board is populated by PK53N512CMD100. The BSP does not work for TWR-K53N512 boards populated by MK53DN512CMD10 with the 4N22D mask set. A workaround is to use the twrk60d100m BSP for this kind of TWR-K53N512 board.

9.11 TWR-K60N512

Figure 29. TWR-K60N512

Table 13. TWR-K60N512

Core Clock	96 Mhz	-
Bus Clock	48 Mhz	-
Default Console	ttyf:	OSJTAG - USB mini connector
BSP Timer	Systick	-

Important Jumper Settings (board Rev. C)

- For standalone operation:
  - TWR-K60N512 - Jumper J6 on position 1-2
- To enable Ethernet communication (use with TWR-SER):
  - TWR-K60N512 - Jumper J6 on position 2-3 - processor clock taken from the TWR-SER board
  - TWR-SER - CLK_SEL 3-4
  - TWR-SER - CLKIN-SEL 2-3 (processor clock is taken from PHY)
  - TWR-SER - ETH-CONFIG J12 9-10 to select RMII communication mode
  - Important: Plug both the processor and the serial board (TWR-SER) into the Tower. The processor is using the external clock from the Ethernet PHY on the serial card.
- To use TWR-LCD board with eGUI:
  - TWR-LCD board SW5 - all switches to ON (enable touch screen)
  - TWR-LCD board SW1 - switches depending on usage either SPI (01111110) or 16 bits FlexBus (10111110)
  - TWR-K60N512 – open J3 13-14

Known Issues:

- The FlexBus FB_AD9 (PTC6) signal on the TWR-K60N512 Rev. C board is directly connected to the IRDA sensor. This prevents using FlexBus to communicate with MRAM and CF-CARD on the TWR-MEM card.
- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.
- TWR-LCD board doesn't work correctly when the navigation buttons are used:
- Usage of center navigation button on TWR-LCD board conflicts with the LCD RESET signal. Both signals are shared on the main elevator A11 and A63.
- Low power does not work VLLSx mode:
  - An issue occurs with the VLLSx on Kinetis PK60N512VDM100 Mask 0M33Z. See the Mask Set Errata for Mask 0M33Z (document KINETIS_0M33Z), in this location: freescale.com/files/microcontrollers/doc/errata/KINETIS_0M33Z.pdf. At Errata ID 2613, MCU may not exit properly from VLLS3, VLLS2, or VLLS1 modes either via pin reset or other wakeup sources. Attempts to wake up or reset from VLLS modes may result in the MCU entering an unknown, non-responsive, state. In the non-responsive state, the MCU current will not match the RUN current prior to entering the low power mode.
  - The current workaround is to decrease voltage supplied to VDD to less than 2.0 V, by removing jumper J18 on the board and then connecting to the external bench supply.
- OSJTAG- USB mini connector issue:
  - In the MQX RTOS 4.1.0 version, the default console is configured to the ttyd: (TWR-SER RS232 connector). This issue will be fixed in the next MQX release.

Notes:

- The default console interface (ttyf:) is routed to OSBDM-COM (USB mini connector J13). Use the P&E Microcomputer Systems OSJTAG terminal to access the board serial line.
- To enable TWR-SER RS232 interface, change the BSP_DEFAULT_IO_CHANNEL configuration option to "ttyd:" in the mqx\source\bsp\twrk60n512\twrk60n512.h file.

9.12 TWR-K60D100M

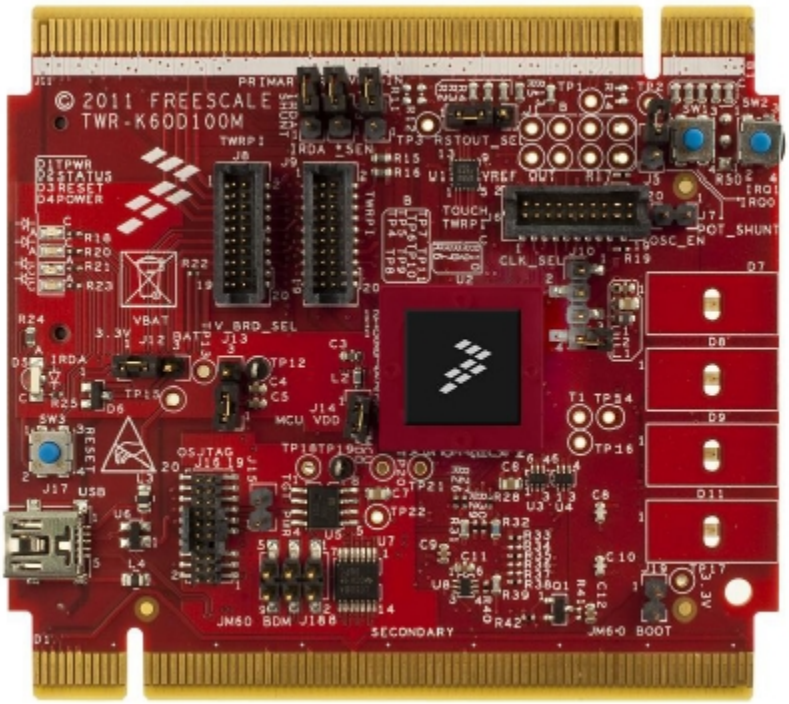

Figure 30. TWR-K60D100M

Table 14. TWR-K60D100M

Core Clock	96 Mhz	-
Bus Clock	48 Mhz	-
Default Console	ttyf:	OSJTAG - USB mini connector
BSP Timer	Systick	-

Important Jumper Settings

- For standalone operation:
  - TWR-K60D100M - Jumper J10 on position 1-2
- For using USB Host mode, jumpers on position:
  - TWR-SER board - J16 on position 1-2 (VB_HOST)
  - TWR-SER board - J10 on default position 1-2 (USB host)
- For using USB Device mode, jumpers on position:
  - TWR-SER board - J16 on position 3-4 (VB_DEV
  - TWR-SER board - J10 on position 2-3 (USB device)
- To enable Ethernet communication (use with TWR-SER):
  - TWR-K60D100M - Jumper J10 on position 2-3 - processor clock taken from the TWR-SER board
  - TWR-SER - CLK_SEL 3-4
  - TWR-SER - CLKIN-SEL 2-3 (processor clock is taken from PHY)

- TWR-SER - ETH-CONFIG J12 9-10 to select RMII communication mode
- Important: Plug both the processor and the serial board (TWR-SER) into the Tower. The processor is using the external clock from the Ethernet PHY on the serial card.

Known Issues:

- Some Compact Flash cards do not work correctly with TWR-MEM and MQX CF Card driver. An issue in the TWR-MEM CPLD code, Rev. A, causes incorrect communication with certain types of cards (e.g. Kingston). A fixed CPLD firmware is available in this folder: <install_dir>/mqx/source/io/pccard/twr_mem_pccard_cpld/. Load the firmware to the TWR-MEM CPLD by using the Altera Quartus II design tool and a BLASTER connection cable.
- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.

Other Notes:

- The default console interface (ttyf:) is routed to OSBDM-COM (USB mini connector J17). Use the P&E Microcomputer Systems OSJTAG terminal to access board serial line.
- To enable TWR-SER RS232 interface, change the BSP_DEFAULT_IO_CHANNEL configuration option to "ttyd:" in the mqx\source\bsp\twrk60d100m\twrk60d100m.h file.

9.13 TWR-K60F120M

Figure 31. TWR-K60F120M

Table 15. TWR-K60F120M

Core Clock	120 Mhz	-
Bus Clock	60 Mhz	-
Default Console	ttyf:	OSJTAG - USB mini connector

Table continues on the next page...

Table 15. TWR-K60F120M (continued)

BSP Timer	Systick	-

Important Jumper Settings (board Rev. B)

- For standalone operation
  - TWR-K60F120M - Jumper J9 on position 1-2
- To enable Ethernet communication (use with TWR-SER):
  - TWR-K60F120M - Jumper J6 on position 2-3 - processor clock taken from the TWR-SER board
  - TWR-SER - CLK_SEL 3-4
  - TWR-SER - CLKIN-SEL 2-3 (processor clock is taken from PHY)
  - TWR-SER - ETH-CONFIG J12 9-10 to select RMII communication mode
  - Important: Plug both the processor and the serial board (TWR-SER) into the Tower. Processor is using external clock from Ethernet PHY on the serial card.
- To enable USB communication:
  - TWR-SER2 – J21 (USB_VBUS_EN) shunt for USBHS

Known Issues:

- Some Compact Flash cards do not work correctly with the TWR-MEM and MQX CF Card driver. An issue in the TWR-MEM CPLD code, Rev. A, causes incorrect communication with certain types of cards (e.g. Kingston). A fixed CPLD firmware is available in <install_dir>/mqx/source/io/pccard/twr_mem_pccard_cpld/ folder. Load the firmware to the TWR-MEM CPLD by using the Altera Quartus II design tool and a BLASTER connection cable.
- The layout design of TWR_K60F120M Rev. B board does not allow NAND flash module to run at 24 MHz clock. To increase the working clock speed, use the drive strength attribute for NAND flash pins. In this package, NAND flash module on Rev. B board works properly at 20 MHz clock.
- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.

Board-Specific Build Targets:

- Int Flash SramData (Debug and Release) – this target enables to build applications that execute code from the internal flash and use the internal SRAM for application data.

Other Notes:

- The default console interface (ttyf:) for RevB board is routed to both OSBDM-COM (USB mini connector J13) and TWR-SER J8 RS232 Connectors. Use the P&E Microcomputer Systems OSJTAG terminal to access the board serial line.

9.14 TWR-K70F120M

Figure 32. TWR-K70F120M

Table 16. TWR-K70F120M

Core Clock	120 Mhz	-
Bus Clock	60 Mhz	-
Default Console	ttyc:	OSJTAG - USB mini connector
BSP Timer	Systick	-

Important Jumper Settings

- For standalone operation:
  - board Rev. B - TWR-K70F120M
    - Jumper J18 on position 1-2
    - Jumper J19 On
  - board Rev. C - TWR-K70F120M
    - Jumper J18 Off - Y1 Enabled
    - Jumper J19 On - Y1 Power
- To enable Ethernet communication (use with TWR-SER):
  - board Rev. B - TWR-K70F120M
    - Jumper J18 on position 2-3 - processor clock taken from TWR primary elevator pin B24
  - board Rev. C - TWR-K70F120M
    - Jumper J18 On - Y1 Output Disabled - processor clock taken from TWR primary elevator pin B24
  - TWR-SER - CLK_SEL 3-4
  - TWR-SER - CLKIN-SEL 2-3 (processor clock is taken from PHY)
  - TWR-SER - ETH-CONFIG J12 9-10 to select RMII communication mode
  - Important: Plug both the processor and the serial board (TWR-SER) into the Tower. The processor is using the external clock from the Ethernet PHY on the serial card.
- USB communication TWR-SER setup:

    - USBFS (KHCI0) module – The BSP is pre-configured to use USBFS module (KHCI0) by default. Try various USB Host and Device example applications and use USB MINIAB connector on TWR-SER board as the USB communication channel.
- USBHS (EHCI0) module – TWR-SER does not allow the use of the USBHS module.
- USB communication TWR-SER2 setup:
    - USBFS (KHCI0) module – The BSP is pre-configured to use USBFS module by default. The USBFS is connected to USB HOST (USB A connector) only. USB device functionality is not available as a result of Hardware limitations.
    - USBHS (EHCI0) module – To enable USBHS (EHCI0), change BSP settings as follows and recompile BSP. The USBHS (EHCI0) is connected to USB OTG (USB mini AB connector). `#define USBCFG_DEFAULT_DEVICE_CONTROLLER (&_bsp_usb_dev_ehci0_if) #define USBCFG_DEFAULT_HOST_CONTROLLER (&_bsp_usb_host_ehci0_if)` in the `<mqx_install_dir>\mqx\source\bsp\twrk70f120m\twrk70f120m.h`
        - i. Ensure that TWR-SER2 – J21 (USB_VBUS_EN) shunt is connected.
        - ii. Because of the Hardware limitation, using USB HS to debug applications may cause errors.
        - iii. Ensure that TWR kit is powered from the TWR-ELEV board (not from the processor board).
        - The keyboard2mouse example application demonstrates the usage of the KHCI (Host) and EHCI (Device) module in one demo. The example requires recompiling the BSP and this setup: `#define USBCFG_DEFAULT_DEVICE_CONTROLLER (&_bsp_usb_dev_ehci0_if) #define USBCFG_DEFAULT_HOST_CONTROLLER (&_bsp_usb_host_khci0_if)` in the `<mqx_install_dir>\mqx\source\bsp\twrk70f120m\twrk70f120m.h`
        - iv. Ensure that TWR-SER2 – J21 (USB_VBUS_EN) shunt is connected.
        - v. Because of the Hardware limitation, using USB HS to debug applications may cause errors.
        - vi. Ensure that the TWR kit is powered from the TWR-ELEV board and not from processor board.

Known Issues:

- FlexCAN1 pins PTC16 and PTC17 are shared with NAND flash memory pins on the TWR-K70FN1M Rev. A board. These pins are correctly set for the FlexCAN1 functionality. CAN1_RX / CAN1_TX signals are not correctly routed to the TJA1050 High speed CAN transceiver on the TWR-SER board. This prevents a correct FlexCAN example functionality.
- For a successful test of the FlexCAN example, insert R22 and R23 on TWR_K70F120M.
- Because RTC and CTS signals are not routed correctly from the TWR-K70FN1M Rev. A board to the RS485 connector of the TWR-SER board, the RS485 demo application does not function correctly.
- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.

Board-Specific Build Targets:

- Int Flash SramData (Debug and Release) – This target enables building applications that execute code from the internal flash and use internal SRAM for application data.
- Int Flash DDRData (Debug and Release) – This target enables building applications that execute code from the internal flash and use external DDR2 memory for application data.

After changing the /config/common/user_config.h file, recompile all MQX libraries. Important: Do not change header files in the output build directory (/lib). The files are overwritten each time the libraries are built.

9.15 KwikStik (MK40X256)

Figure 33. KwikStik (MK40X256)

Table 17. KwikStik (MK40X256)

Core Clock	96 Mhz	-
Bus Clock	48 Mhz	-
Default Console	iodebug: (ttyf):	Use the I/O Debug as an output only. Avoid using it as an input because reading from the Semihost port causes the microprocessor core to halt. If the console input is required, use the serial line (channel ttyf:) n RS232, DB9 on the TWR-SER board. Power the board from the USB connected to the TWR-ELEV card.
BSP Timer	Systick	-

Important Jumper Settings:

- For using RAM disk, jumpers on position:
  - TWR-MEM board - J16 remove
  - TWR-MEM board - J11 remove (default)Board-specific build targets

Known Issues:

- RTC clock - The VBAT pin is not powered (R117 is DNP in the schematics) on the KwikStik Rev. 1 – 4 boards. The issue is resolved on Rev. 5 boards. Enabling RTC in older revision boards prevents using the card (a board is automatically rebooted during startup).
- SD Card driver - The data pins of the SD card connector (DATA0~DATA3) are connected to the incorrect processor pins (SDHC0_D4~SDHC0_D7). The issue is present on KwikStik Rev. 1 – Rev. 4 boards and resolved on Rev. 5 boards.

9.16 TWR-K64F120M

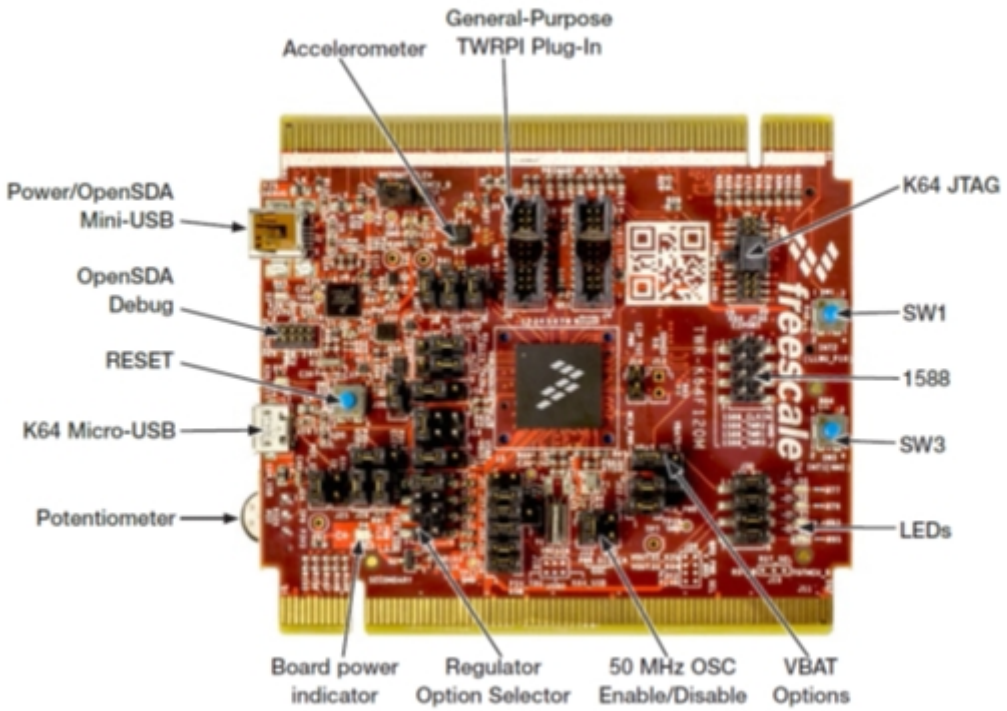

Figure 34. TWR-K64F120M

Table 18. TWR-K64F120M

Core Clock	120 Mhz	-
Bus Clock	60 Mhz	-
Default Console	ttyb:	OpenSDA - USB mini connector
BSP Timer	Systick	-

Important Jumper Settings (board Rev. B)

- The default jumper settings for TWR-K64F120M standalone operation:
  - J1, J16, J18, J19, J20, J31, J38 on position 1-2
  - J6, J7,J8, J22, J23, J24, J25, J26, J27, J28, J33, J31, J34, J35, J36 and J39 closed
  - J32 open
  - J29 on position 5-6
  - J10 and J15 on position 2-3
- To enable Ethernet communication (use with TWR-SER):
  - TWR-K64F120M - Jumper J32 on position 1-2 - processor clock taken from the TWR-SER board
  - TWR-SER - CLK_SEL 3-4
  - TWR-SER - CLKIN-SEL 2-3 (processor clock is taken from PHY)
  - TWR-SER - ETH-CONFIG J12 9-10 to select RMII communication mode
  - Important: Plug both the processor and the serial board (TWR-SER) into the Tower. Processor is using external clock from Ethernet PHY on the serial card.
- For USB Device mode, use onboard J17 Micro USB connector and jumpers set to default settings.

- For USB Host mode, use onboard J17 Micro USB connector and jumpers set to default settings. Connect the J2 (K20 DEBUG) Mini USB to the PC to enable power for the USB Host.
- For the SD card example:
  - TWR-K64F120M System module, place the resistor 4.7 K into R521.
- For the RAMDisk example:
  - TWR-SER System module, J5 5-6 open.
- For the USB port on TWR-SER:
  - Remove R63, R65 and place R522 and R523.
  - J19 change position to 2-3.

Known Issues:

- Some Compact Flash cards do not work correctly with the TWR-MEM and MQX CF Card driver. An issue in the TWR-MEM CPLD code, Rev. A, causes incorrect communication with certain types of cards (e.g. Kingston). A fixed CPLD firmware is available in <install_dir>/mqx/source/io/pccard/twr_mem_pccard_cpld/ folder. Load the firmware to the TWR-MEM CPLD by using the Altera Quartus II design tool and a BLASTER connection cable.
- Example projects contain build configurations for code execution different than the Flash or RAM memory. The RAM-based execution may be faster to debug but not all examples fit into RAM and may not link.

Known Issues:

- USB Host CDC Serial: USB does not work after the CDC device is re-attached.
- Because the DMA TX channel and DMA RX channel for SPI1 and SPI2 are the same, SPI1 and SPI2 do not use the DMA driver.
- Because the UART5 TX and RX conflict with LED3 and LED4, some demos using LED3 and LED4 do not run properly, such as HVAC, HVAC_Error, and Web_hvac examples. The workaround uses ttyb instead of ttyf and connects A76 (UART3_RX) with RXD_SEL in TWR-SER and A77 (UART3_TX) with TX_SEL in TWR-SER.
- FlexBus shares pins with the SAI, CAN, and TTYB modules. To run the demo by using FlexBus (Ramdisk), disable the SAI module in user_config.h, change the IO port to TTYF, and remove jumper J5 (CAN_SEL) from TWR-SER.
- Because the DMA TX channel and the DMA RX channel of SCI4 and SCI5 are the same, SCI4 and SCI5 do not use a DMA driver.

9.17 TWR-MCF51JF128

The MCF51JF128 BSP was tested for these hardware configurations:

- TWR-MCF51JF128 Rev. A processor board
- TWR-SER Rev. C serial board
- TWR-ELEV Primary and Secondary - four-story elevator boards
- TWR-MEM Rev. B memory extension board. [optional]

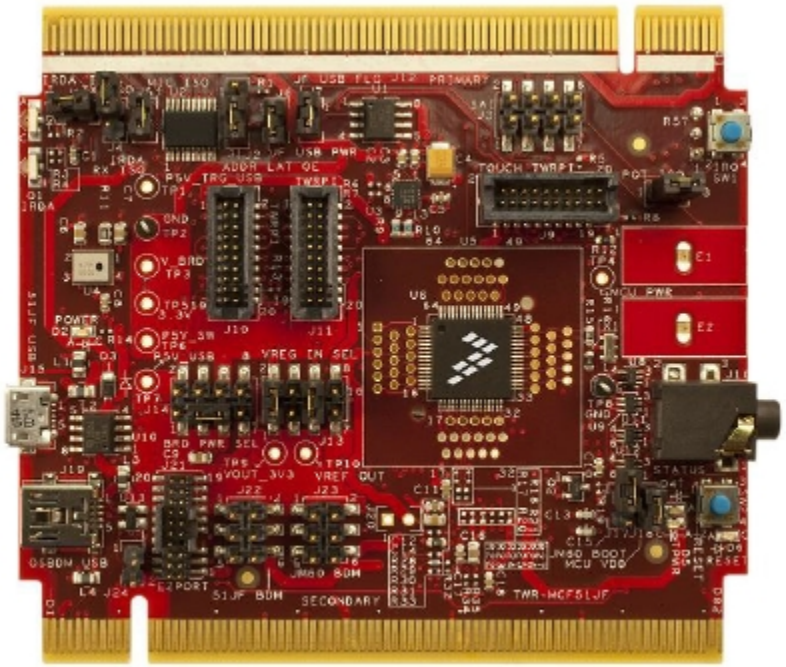

Figure 35. TWR-MCF51JF128

Table 19. TWR-MCF51JF128

Core Clock	48 Mhz	-
Bus Clock	24 Mhz	-
Default Console	ttya:	RS232 on TWR-SER board
BSP Timer	MTIM1	-

9.18 TWR-MCF52259

TWR-MCF52259-KIT (Rev. A) consists of this:

- MCF52259 microcontroller module board
- TWR-ELEV four-story elevator boards
- TWR-SER serial board
- [optional] TWR-MEM memory extension board
- [optional] TWR-LCD display board

Figure 36. TWR-MCF52259

Table 20. TWR-MCF52259

Core Clock	80 Mhz	-
Bus Clock	40 Mhz	-
Default Console	ttyb:	RS232
BSP Timer	PIT0	-

Important Jumper Settings:

- For basic operations, ensure that these jumper settings are used:
  - TWR-SER board - J2 on default position 1-2 (PHY CLK_SEL 25MHz)
  - TWR-SER board - J3 shunt removed (CLKIN_SEL)
  - TWR-SER board - J15 on default position 1-2 (SER_SEL enabling RS232 operation)
  - TWR-SER board - J17 on default position 1-2 (RXD_SEL enabling RS232 operation)
  - TWR-SER board - J18 shunt removed (RTS_SEL no RS232 flow control)
  - TWR-SER board - J19 on default position 1-2 (TXD_SEL enabling RS232 operation)
- To enable external MRAM memory (available on Memory Storyboard):
  - TWR-MEM board - J10 on position 1-2
  - TWR-MEM board - J11 shunt removed
- To enable Ethernet (in addition to basic operation jumper setting above):
  - TWR-SER board - J12 shunt on pins 15-16 for duplex operation
- To enable USB operation in HOST mode:
  - TWR-SER board - J16 shunt on pins 1-2
  - TWR-SER board - J10 shunt on pins 1-2
- To enable USB operation in DEVICE mode:

    - TWR-SER board - J16 shunt on pins 3-4
    - TWR-SER board - J10 shunt on pins 2-3
- To enable SD Card operation:
    - TWR-MEM Board - J3 on position 1-2 to route QSPI_PCS0 to SD Card Chip Select
    - TWR-MEM Board - J4 remove shunt on pins 1-2 to disable QSPI_PCS0 routing to serial Flash
    - TWR-MEM board- J12 all jumpers except 3-4 populated
- To enable CompactFlash Card operation (available on Memory Storyboard):
    - TWR-MEM board - J16 on position 2-3
- To use write protect detection signals with SD Card on the Memory Storyboard:
    - TWR-MCF52259 board - turn off switch 3 on SW2 dip-switch.
- To select either CS0 or CS1 for SPI Flash:
    - TWR-MEM board - J14 on position 1-2 (CS0)
- To use TWR-LCD board with eGUI:
    - TWR-LCD board - SW5 all switches to ON (enable touch screen)
    - TWR-LCD board - SW1 switches depending on usage either SPI (01111110) or 16 bits FlexBus (10111110)
    - TWR-MCF52259 board - to enable navigation buttons set SW2 dip 2 and SW2 dip 3 to OFF

Specific Build Targets:

- None. Recompile all MQX libraries after changing /config/common/user_config.h file.
- Important: Do not change header files in the output build directory (/lib). The files are overwritten each time the libraries are built. After changing the /config/common/user_config.h file, recompile all MQX libraries.
- Important: No changes should be made to header files in the output build directory (/lib). The files are overwritten each time the libraries are built.
- See Chapter 2.3 “Library Build Targets” for more details about the standard build targets. The Ext. MRAM Debug target can be used only with Memory Storyboard.

Other Notes:

- The OSBDM Firmware compatibility issue may affect application debugging. See *Freescale MQX ™ RTOS Release Notes* (document MQXRN) for more details about the OSBDM Firmware Compatibility.

9.19 TWR-MCF54418

TWR-MCF54418-KIT supports this hardware configuration:

- TWR-MCF54418 Rev. D processor board
- TWR-SER2 Rev. C serial board
- TWR-ELEV Primary and Secondary - four-story elevator boards
- TWR-MEM Rev. B memory extension board

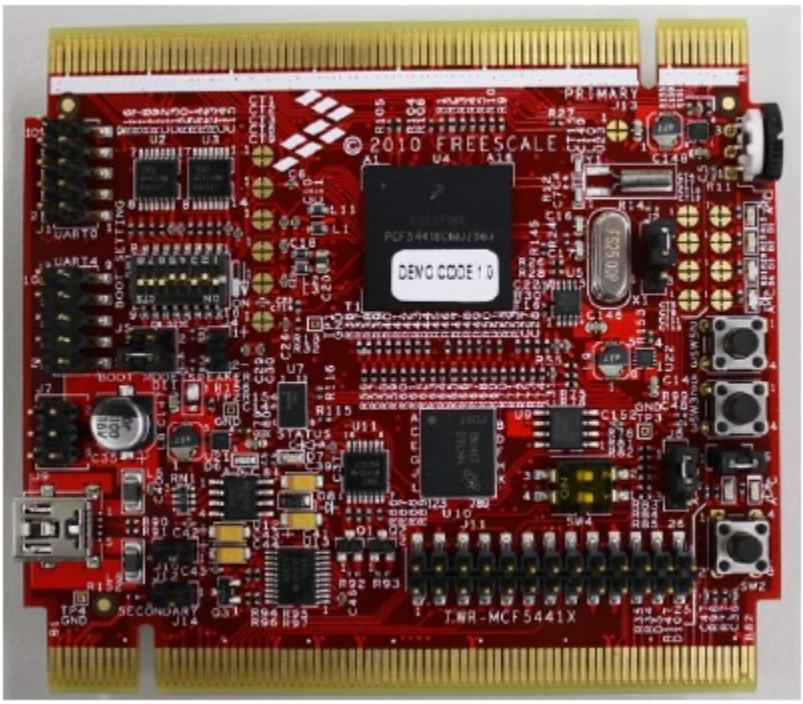

Figure 37. TWR-MCF54418

Table 21. TWR-MCF54418

Core Clock	250 Mhz	-
Bus Clock	125 Mhz	-
Default Console	ttyd:	RS232
BSP Timer	PIT0	-

NOTE

Plug both the processor and the serial board (TWR-SER2) into the Elevator bus. The MCF54418 processor is using the external clock generated by the Ethernet PHY on the Serial card.

Important Jumper Settings:

TWR-MCF54418 Rev. D board (use the highlighted setting for a basic MQX operation)

- J2 on position 1-2 Input Clock Selection
  - 1-2 external clock (RMII clock from TWR-SER2 board)
  - 2-3 onboard 25MHz clock
- J8 on position 1-2 TCK/PSTCLK Routing:
  - 1-2 PSCCLK routed to pin 24 of BDM header J11
  - 2-3 PSTCLK routed to pin 6 of BDM header J11
- J6 on position 1-2 To enable PE micro debugger
- J5 on position 3-4 Boot Mode Selection
  - 1-2 and 3-4 Internal RCON
  - 3-4 External RCON
  - No Jumper Serial Boot

- J4 no jumper 8 Ohm speaker connector
- J10 no jumper IRQ active at high level
- J12 no jumper MCU Reset In
- SW4: Both off
- SW1: 1-on, 2-off, 3-on, 4-off, 5-off, 6-on, 7-on, 8-on (booting from NAND)

TWR-SER2 Rev. C Board: (use the highlighted setting for a basic MQX operation)

- J1 on position 2-3 RS232/485 RX Select (UART1)
  - 1-2 RS485 Mode (connects RX to RO)
  - 2-3 RS232 Mode (connects RX to R1OUT)
- J2 on position 2-3 RS232/485 TX Select (UART1)
  - 1-2 RS485 Mode (connects TX to DI)
  - 2-3 RS232 Mode (connects TX to T1IN)
- J4 no jumper Can Isolation
  - 1-2 Connects CAN_S to S
  - 3-4 Connects CAN_TX to TXD
  - 5-6 Connects CAN_RX to RXD
- J7 on positions 1-2, 3-4 JS16 RS232 Isolation (UART0)
  - 1-2 Connects RX to S08JS16 RXD
  - 3-4 Connects TX to S08JS16 TXD
- J8 no jumper Power Down Port B
  - 1-2 Disables Ethernet PHY B
- J9 no jumper Power Down Port A
  - 1-2 Disables Ethernet PHY A
- J11 no jumper RS485 Config (UART1)
  - 1-2 Loopback Mode (connects RE to DE)
  - 3-4 Loopback Mode (connects TX0_P to RX0_P)
  - 5-6 Loopback Mode (connects TX0_N to RX0_N)
  - 7-8 NC
  - 9-10 5V Supply to DB9
- J13 on position 1-2 RS232/485 Disable (UART 1)
  - 1-2 Disables RS485
  - 2-3 Disables RS232
- J16 no jumper VBUS OC Isolation
  - 1-2 Connects USB VBUS OC to Elevator
- J19 no jumper UART2 Connector
- J20 no jumper UART3 Connector
- J21 no jumper VBUS EN Isolation
  - 1-2 Connects USB VBUS EN to Elevator
- J22 no jumper RS232 (UART2) Isolation
  - 1-2 Connects TX to T1IN
  - 3-4 Connects RX to R1OUT
  - 5-6 Connects RTS to T2IN
  - 7-8 Connect CTS to R2OUT
- J23 no jumper RS232 (UART3) Isolation
  - 1-2 Connects TX to T1IN
  - 3-4 Connects RX to R1OUT
  - 5-6 Connects RTS to T2IN
  - 7-8 Connect CTS to R2OUT
- J24 no jumper USB Device Mode
  - 1-2 Device Mode (capable of powering Tower System)
- SW1 1-on ,2-on (MII MODE pull-up, RXDV) 3,4,5,6,7,8 off
- SW2 1-on, 3-on (MII MODE pull-up, 50MHz) 2,4,5,6,7,8 off

TWR-MEM Rev. A – TWR MEM can operate only with TWR-SER2 card in a default setting. Use TWR–SER board for the SDHC operation.

For eSDHC operation:

- J12: (SD-SEL1) on position 1-2 to enable SD Card detect signal
- J12: (SD-SEL1) on position 5-6 to enable SD Card data[1] signal
- J12: (SD-SEL1) on position 7-8 to enable SD Card data[2] signal
- J12: (SD-SEL1) on position 9-10 to enable SD Card cmd pull up
- J12: (SD-SEL1) on position 11-12 to enable SD Card data[0] pull up
- J2: (SD-SEL2) on position 2-3 to enable SD Card data[3] pull down
- J3: (SD-CS) on position 1-2 to enable SD Card data[3] signal
- J13: on position 1-2 to enable SD Card write protect signal

Board-specific build targets:

- Ext Flash (Debug and Release) - These targets enable building applications to boot the system from the external NAND Flash memory. After the reset, the initialization code of the application (bootstrap) is loaded from NAND Flash to SRAM. This initialization code copies the rest of the application to the DDR RAM memory and executes the application there.
- Note that this could take a while especially if a large application is started.
- See NAND Flashing procedure below. There are two variants of Ext Flash target in the application projects: one for external P&E Microcomputer Systems BDM interface (PEBDM Ext Flash) and the other for on-board OSBDM interface (OSBDM Ext Flash). The OSBDM debugging connection does not work correctly with the old version of the OSBDM firmware. Please update the OSBDM firmware before using this target. When using the OSBDM interface, power the TWR-MCF54418-KIT by using the USB attached to the primary elevator.
- Recompile all MQX libraries after changing the /config/common/user_config.h file.
- Important: Do not change header files in the output build directory (/lib). The files are overwritten each time the libraries are built. After changing the /config/common/user_config.h file, recompile all MQX libraries.
- Important: No changes should be made to header files in the output build directory (/lib). The files are overwritten each time the libraries are built.
- See Chapter 2.3 “Library Build Targets” for more details.

NAND Flashing Procedure:

The CodeWarrior Development Studios for ColdFire V7.2.2 and for MCU v10 do not provide NAND Flashing functionality. This functionality is planned for future releases. The MQX release contains a standalone CFFlashprog utility which enables NAND Flashing by using P&E Microcomputer Systems BDM interface (OSBDM is not supported). The NAND Flashing Procedure is as follows:

- Connect the P&E Microcomputer Systems BDM cable to the board and switch on the power.
- Compile the PEBDM Ext Flash target in the selected CodeWarrior project.
- Ensure you have SW1 DIP switch set correctly for the NAND booting (see jumper setting above).
- Locate <output_name>.rbin file which should be created in the application output directory.
- Open Windows Command Line Prompt (run cmd.exe) and change directory to “ <install_dir>\tools\flash_programmer \CFFlashprog”
- Use cf.exe NAND erase M54418TWR_nand 0 200000 command to erase the first 2 MB of NAND Flash memory.
- Use cf.exe NAND write M54418TWR_nand 0 200000 1 <path_to_rbin_file>” command to write the application code to the NAND Flash memory.
- For a more detailed description of the NAND Flash tool, see: <install_dir>/tools/flash_programmer/CFFlashprog/ ReadMe.txt

Known Issues:

- Some Compact Flash cards do not work correctly with the TWR-MEM and MQX CF Card driver. There may be several reasons:
  - An issue in the TWR-MEM CPLD code, Rev. A, causes incorrect communication with certain types of cards (e.g. Kingston). A fixed CPLD firmware is available in <install_dir>/mqx/source/io/

pccardtwr_mem_pccard_cpld/ folder. The firmware can be loaded to the TWR-MEM CPLD by using the Altera Quartus II design tool and a BLASTER connection cable.

- M54418 MQX CF card driver incorrectly detects the card in the slot. If you experience this behavior, connect a two pull-up resistors between the card detect pins (CF_CD1, CF_CD2) and 3.3V VCC.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Document Number: MQXGSRTOS
Rev. 4.1.1, 08/2014