Document Number: MQXBSPPG
Rev. 0, 02/2014

Freescale MQX™ RTOS BSP Porting Guide

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	4.1.0
DESCRIPTION:	Freescale MQX™ BSP Porting Guide
RELEASE DATE:	February, 2014

How to Reach Us:

Home Page:
www.freescale.com

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

Document Number: MQXBSPPG
Rev. 0, 02/2014

Table of Contents

1 Introduction

Freescale MQX™ RTOS is provided with source code and Board Support Packages (BSPs) for different Freescale development boards. The BSPs include the low-level code specific to the hardware both internal to the microcontroller (MCU) and external hardware on the circuit board. The low-level code, which is provided in the BSPs, includes the peripheral drivers, clock configuration, memory map, memory management, and more. The BSPs are an excellent starting point for customization.

This document provides is a guide to porting the MQX BSPs to the custom target hardware with a similar MCU derivative. An example of the porting process is provided in the *Freescale MQX™ RTOS BSP Porting Example User Guide* (document MQXBSPEXUG).

The included BSPs are written specifically for the Freescale development boards. Any of the following hardware differences between the custom hardware and a development board require some changes to the BSP:

- Different peripherals and peripheral drivers
- Different pins for the peripherals and GPIO
- Different clock source, clock configuration, and clock frequencies
- Custom drivers
- Different MCU derivatives

Although this document focuses on the Freescale Kinetis family of MCUs, it also applies to other Freescale architectures supported by the Freescale MQX.

1.1 BSP Porting Process Overview

These are the steps to port an MQX BSP. Some of these steps are optional, including the use of the Processor Expert (PEx).

1. Clone BSP – makes a copy of a provided BSP with a new name which is later modified.
2. Add PEx to the BSP project – only needed if using PEx with IAR Embedded Workbench® or MDK-ARM Keil™ tool chain.
3. Change the PEx CPU Component/Package – optional step if using PEx in the BSP.
4. Change MCU derivative.
5. Modify clock configuration.
6. Modify BSP source/include files and user_config.h.
7. Modify derivative-specific driver files.
8. Add/remove driver source files in BSP project.
9. Modify BSP Memory map and linker file.
10. Modify post-link batch files.
11. Change CodeWarrior debugger memory file – optional if using CodeWarrior and dependant on the derivative.
12. Port example applications to a new BSP – optional for testing the new BSP.

2 Cloning a Freescale BSP

These are the recommended steps when creating a customized MQX BSP:

1. Find the provided BSP in the MQX release that is closest to the device used in the customized BSP.
2. Clone that BSP and create a copy of the original with a new name, directory, and paths.
3. Modify the clone as needed.

2.1 Selecting closest Freescale BSP to clone

The BSP porting effort is easiest when selecting the proper BSP to clone. When the two derivatives are very similar, there are minimal changes required. Freescale does not provide an MQX BSP for every Kinetis derivative. Instead, the BSPs are provided for the different Kinetis development boards available. The MCU derivatives on the development boards are chosen because they are the superset devices for that MCU family. The other derivatives are similar devices, usually with reduced features or memory. Therefore, any Kinetis derivative BSP can be ported from one of the included MQX BSPs. It is important to understand which board and which MQX BSP is closest to the desired MCU derivative on the custom hardware. Appendix A in this document can be used to find the assist in making this determination.

2.2 BSP Cloning Wizard

After selecting the best BSP, the next step is to clone the BSP. Cloning the BSP copies all the files, directories, and projects, and renames all the references of the original BSP to the new BSP. The result is a BSP with a new name that is a clone of the original.

Freescale MQX installation includes a tool to automate the cloning process called the BSP Cloning Wizard. Using this tool reduces the cloning time and the potential for a manual error. The BSP Cloning Wizard is a Windows application installed with MQX at **<MQX Installation Directory>\tools\BSPCloningWizardBSPCloningWizard.exe**. For more information about the tool, see the *Getting Started with Freescale MQX™ BSP Cloning Wizard* (document MQXGSCLW).

This tool also provides the ability to export a newly cloned BSP and creates a new MQX directory structure for the selected BSP. Because the exported directory contains only the files for a particular BSP, it is very useful when sharing, archiving, or using version control for MQX source code.

Note that the *MQX Board Support Package Porting Guide* (document AN4287) described the cloning process before the BSP Cloning Wizard was created. Even though the document can still be used as a guide, the Cloning Wizard automates the process and saves time and effort.

3 Using Processor Expert (PEx) in the BSP

Processor Expert (PEx) is a code-generation tool from Freescale. It can generate the driver and the initialization code for the MCU peripherals and also make the MCU clock setup easy. It enables a quick hardware setup and can also be used as a tool to learn about the peripherals and the usage of registers. PEx can be obtained either as a part of the Freescale's CodeWarrior tool chain or it can be downloaded as a stand-alone application Processor Expert Driver Suite and used with other tool chains. To learn more about PEx, find documentation and training, or to download, visit freescale.com/processorexpert.

PEx is used to create the MQX Kinetis BSPs. In particular, the clock configurations used in the BSPs are generated by PEx. For more information about how PEx can be used for the clock configuration of the BSP, see *How-to Change Default Clock Settings in Kinetis BSPs* (document MQXGSCLKBSP).

Using PEx is optional for MQX and a matter of preference. Some of the benefits to using PEx in the MQX BSP include:

- Simplifies clock configuration. PEx quickly and easily changes the clock sources and frequencies of the MCU. Even if PEx is not used in the BSP, it can still be used to quickly generate the clock configuration code. See Clock Configuration.
- Having multiple clock configurations. Some applications need to dynamically change clock frequencies and clock sources, especially in low-power applications. PEx enables both the multiple clock configurations and APIs to change them at runtime. The MQX Low Power Manager (LPM) driver utilizes the different clock configurations for different low-power states. PEx makes it simple to add and manage the clock configurations.
- Adds or replaces the peripheral drivers to the MQX BSP. PEx adds drivers for the peripherals that are available in the MCU, but not MQX. See the CW *for Microcontrollers V10 and MQX*™ (document MQXCWPP) for more information.

The MQX source code can work with or without PEx. When PEx is used, the hardware initialization and clock configuration generated from PEx are used. Otherwise, the project defaults to the initialization and clock configuration code already present in the MQX BSP source code. The macro PE_LDD_VERSION is used in the build to determine if the PEx code should be used or not. When PEx is used, the macro is defined and the PEX generated code is loaded over the MQX source code.

4 Changing the MCU derivative in MQX projects

If the device used in the customized BSP is different than the original BSP, the MCU derivative needs to be changed. Here is a summary of the process for changing the MCU derivative:

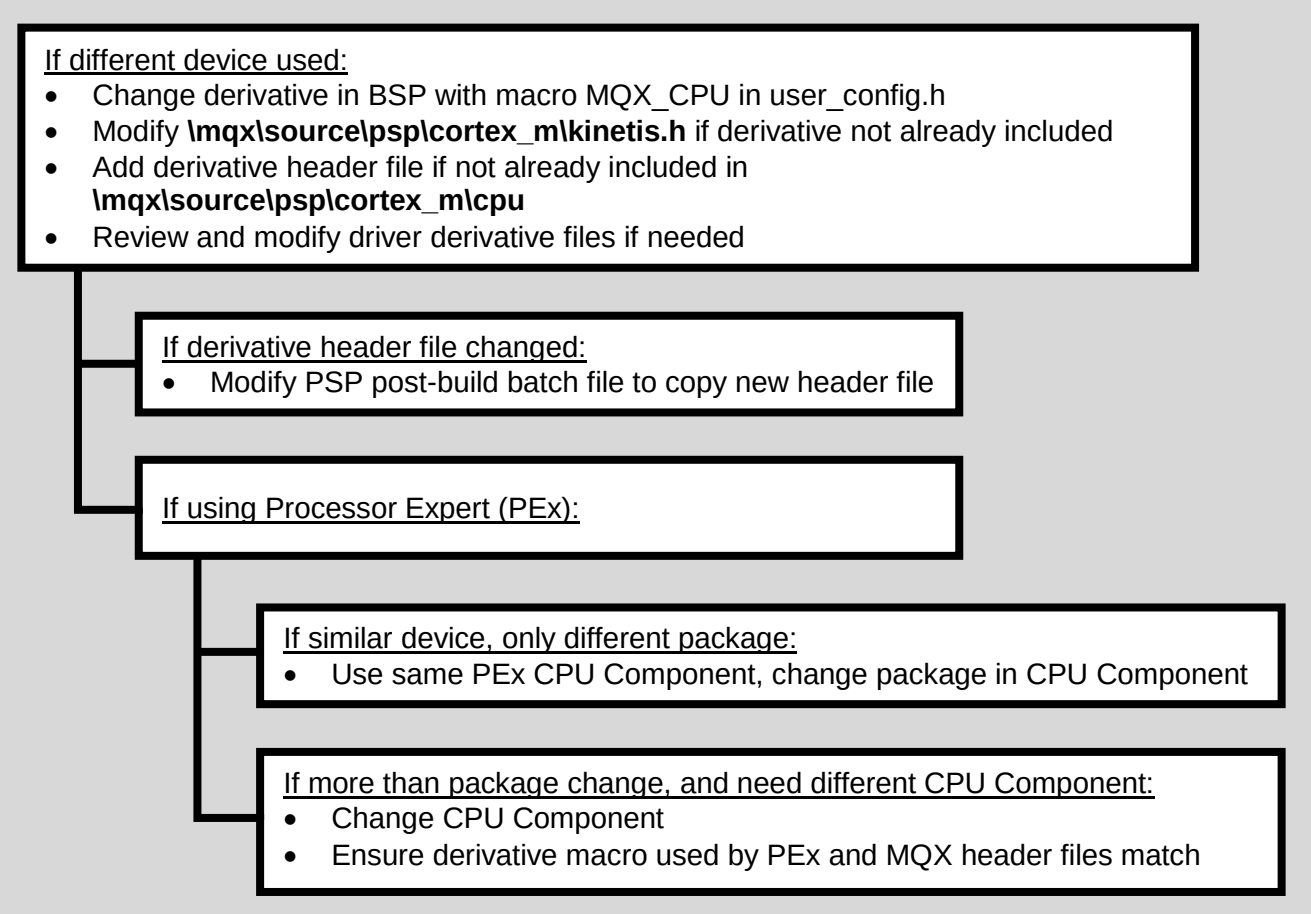

Figure 1: Changing MCU Derivative Process Summary

MQX uses macros for the MCU derivative, board specifics, and derivative header files. This derivative is defined with the macro MQX_CPU in the user_config.h. The file **\mqx\source\psp\cortex_m\kinetis.h** uses this derivative to include the derivative header file into the PSP. Changing the MCU derivative macro can also cause changes in the driver derivative files, (see Modifying Driver Derivative Files).

If the derivative header file is changed, the PSP post-build batch file needs to be modified. The batch file runs after the PSP project is built and copies header files to the **\lib** folder to be used when compiling the application. The batch file must be updated to copy the different header file.

If using PEx and the MCU derivative changes, PEx should also be updated with the new derivative. If the only difference in the derivatives is the package, the same CPU component can be used and the package can be changed in the CPU Component. The MCU derivative does not need to change in that case. If the derivative difference is more than the package, then the PEx CPU component needs to change. The derivative macro used in the PEx CPU component must match the derivative macro used in the MQX header files. Otherwise, two register header files will be included, and cause build errors. Any time the CPU component is changed, the derivative macro in the MQX headers must be changed to match it.

4.1 Modifying Driver Derivative Files

Many of the MQX drivers include source files with specifics about the MCU derivative, such as the number of peripherals on that derivative, the base addresses of the peripherals, and their interrupt vectors. The driver files included with the MQX release do not cover all MCU derivatives; they have support for the derivatives used on the supported boards. Therefore, changing the MCU derivative used in the MQX BSP can require changes to the corresponding driver source files.

This is a list of driver source files specific to the MCU derivative for the Kinetis BSPs and valid for MQX version 4.0.2 and later:

- ADC driver: **\mqx\source\io\adc\kadc\ADC_mkxx.c/.h**
- CAN driver: **\mqx\source\io\can\flexcan\FlexCAN_mkxx.c**
- Ethernet driver: **\mqx\source\io\enet\macnet\MACNET_mkxx.c**
- SDHC driver: **\mqx\source\io\esdhc\eSDHC_mkxx.c**
- I2C driver: **\mqx\source\io\i2c\I2C_mkxx.c**
- I2S driver: **\mqx\source\io\i2s\I2S_mkxx.c**
- LWADC driver: **\mqx\source\io\lwadc\lwadc_kxx.c**
- NAND flash driver: **\mqx\source\io\nandflash\nfc\nfc_kxx.c**
- SAI driver: **\mqx\source\io\sai\sai_mkxx.c**
- UART driver: **\mqx\source\io\serial\serl_mkxx.c**
- SPI driver: **\mqx\source\io\spi\spi_mkxx.c**
- USB DCD driver: **\mqx\source\io\usb_dcd\usb_dcd_mkxx.c**

MQX release includes several driver source files for the different Kinetis sub-families (K20, K40, K60, etc.) But not all sub-families are included because MQX only supports the derivatives used on the supported boards. Note that a driver source file for a specific sub-family may not include support for the desired derivative. For example, when porting the TWRK60D100M to a K40, both of these derivatives have one SAI peripheral in the silicon. The TWRK60D100M BSP enables one SAI in the SAI driver with the file **\mqx\source\io\sai\sai_mk60.c**. However, because the other K60 BSPs have two SAIs, the driver file **sai_mk60.c** conditionally sets the number of supported SAIs with macros defined in the BSP, with code like this:

```
switch (dev_num)
    {
        case 0:
            addr = (pointer)I2S0_BASE_PTR;
            break;
#if !defined(MCU_MK60D10)
         case 1:
            addr = (pointer)I2S1_BASE_PTR;
            break;
#endif
```

When the MCU derivative is changed and is no longer MCU_MK60D10, the file **sai_mk60.c** tries to include two SAIs when the silicon only supports one. This causes build errors because the K40 header file doesn't include the symbols for the second SAI peripheral. The build error can be fixed by modifying **sai_mk60.c** to exclude the second SAI reference for this derivative, or change the BSP project to use the **sai_mk40.c** file instead.

Derivative specific driver source files should be reviewed for the desired derivative and modified as needed. See the device documentation for what is supported on the derivative. In particular, the chip configuration chapter in the device reference manual and the family product brief are useful for details about the peripherals available on a specific derivative.

5 Clock Configuration

Frequently, when a BSP is customized, the clock configuration must be changed. This is done to use a different clock source and frequency and to change the frequencies and dividers of the internal clocks.

Using Processor Expert simplifies these changes because the clock configurations can be easily changed with the PEx GUI. Even if PEx is not included in the BSP project, it can still be used to simplify the process by creating a new PEx project and configuring the CPU component for the desired clock settings. Finally, the generated code is copied into the BSP source files.

For more details about using PEx to change the clock configuration in the BSP, see the *How-to Change Default Clock Settings in Kinetis BSPs* (document MQXGSCLKBSP).

If PEx is not used in the BSP project, then the BSP uses the file **\mqx\source\bsp\<Board_Name>\bsp_cm.c** to configure the clocks. The functions used in this file to configure the clocks actually come from PEx. If PEx is not included in the BSP, PEx in a new project can configure the clocks, and then the functions can be copied from **\Generated_Code\Cpu.c to bsp_cm.c**. Alternatively, the **bsp_cm.c** can be modified manually without PEx.

6 BSP include files

The BSP is customized by several header files located in the **\mqx\source\bsp\<Board_Name>**. All files in this directory should be reviewed and modified as needed for the board and application. These are some of the key header files to consider.

6.1 <Board_Name>.h File

Each BSP has a header file named for the board, such as twrk60d100m.h. This file has most of the hardware customization for the BSP and should be reviewed when customizing the BSP. Some key settings include:

- MCU Memory Map - In addition to the linker command files, the BSP includes macros for the memory map of the specific MCU derivative. These include memory map symbols for the board, and can include internal and external memory, such as the base addresses and sizes of different memory areas (Flash, RAM, external peripherals, etc.). See a device reference manual for details about the memory map for a specific device.
- Clock Configuration Changes - In addition to the changes in Clock Configuration, this file can also be used to change these clock options:
    - RTOS Tick Period. This file includes a macro BSP_ALARM_FREQUENCY to define the period of the tick timer for the scheduler. The macro defines the number of ticks per second. MQX defaults to a 200 for a 5 ms tick.
    - RTOS Tick Timer. This file has some BSP_SYSTIMER macros to define which hardware timer is used for the RTOS Tick. The Kinetis BSPs default to using the SysTick Timer in the ARM® Cortex®-M4 core. This can be changed to other hardware timers.
- GPIO Board Specifications - This file contains macros for the board-specific GPIO used for several of the included examples. It can be useful to customize these for the board and use the examples to test the BSP.
- I/O Driver Default Options - Many of the drivers use macros defined in this file for the default options. These options should be reviewed when customizing the BSP.
- Default MQX Initialization - When MQX is initialized, it uses certain default settings that are included in this file.

6.2 BSP_prv.h

This file includes several declarations for private functions and structures in the BSP. It does not require major changes to build the customized BSP.

6.3 BSP.h

This file includes header files needed for the drivers and BSP functions, and public declarations used by the application. It should be modified for different driver support.

6.4 user_config.h

This is the master configuration file for the MQX libraries. This file should be reviewed and customized for board and application.

7 BSP Initialization files

The BSP is customized by several source files located in **\mqx\source\bsp\<Board_Name>**. All files in this directory should be reviewed and modified as needed for board and application. These are key source files to consider.

7.1 init_bsp.c

This file initializes the BSP and the drivers that are enabled by default in the BSP.

- **_bsp_pre_init()** and **bsp_init()** - These functions initialize the hardware enabled by default in the BSP. It is called after the kernel is initialized in **mqx()** in **mqx.c** in the PSP. Because many of the driver initialization functions are determined by the macros defined in user_config.h, not many changes are required in this file. However, it should be reviewed.
- Hardware timer for kernel tick - MQX now uses a hardware timer driver for the kernel tick. This driver is also initialized in **init_bsp.c**. If the custom BSP requires a change in the hardware timer peripheral used for this tick, see the HWTIMER driver chapter in the *Freescale MQX™ I/O Drivers User Guide* (document MQXIOUG) and update the source file as needed.

7.2 init_HW.c

This file needs hardware initialization before the kernel is initialized. **init_hardware()** in this file is called by the PSP before starting MQX. If the external memory is required for the kernel, it is usually initialized in this file.

7.3 init_GPIO.c

This is a main file in the BSP that needs to be customized. It controls the pin setup for the device including the pin muxing options for the driver signals.

7.4 MQX_init.c

This file includes the structure, **MQX_init_struct**, which includes the default settings when the kernel is initialized. The parameters used in this structure are macros defined in the header files already discussed. This file does not need to be changed for a customized BSP.

7.5 vectors.c

This file is used to program the Kinetis flash configuration field and the hardware interrupt vector table. Changes are not required when customizing a BSP, but they may be desired depending on the application. If the file needs to be changed, see the device reference manual for the specifics and modify as needed. See “Handling Interrupts and Exceptions” in the Freescale MQX™ RTOS User Guide for details about the way MQX manages interrupts.

7.6 bsp_cm.c

Changes to this file are covered in Section Clock Configuration.

8 BSP Driver Changes

Part of porting the BSP is customizing the drivers for the MCU derivative, board, and application. While the driver source files in **\mqx\source\io** are common across different BSPs, the BSP customizes the drivers using files in **\mqx\source\bsp\<Board_Name>**. All files in this directory should be reviewed and modified as needed for the board and application.

Depending on the driver, certain derivative-specific source files may need to be modified. See Modifying Driver Derivative Files.

If the application requires drivers that are not included in the MQX release, they can be created and added to the BSP. See the Application Note AN3902 How to develop I/O Drivers for MQX for details about creating the MQX driver compatible with the MQX I/O subsystem. PEx can be used to generate the peripheral driver code and include it in the BSP. See CW *for Microcontrollers V10 and MQX*™ (document MQXCWPP) for a tutorial using the PEx-generated drivers in a BSP. Using PEx like this requires PEx to be included in the BSP. See Using Processor Expert (PEx) in the BSP for more details.

8.1 init_<driver>.c Files

These are the files included in the BSP in **\mqx\source\bsp\<Board_Name>** that customize the driver. They include settings for the MCU derivative, board, and application.

8.2 Drivers for External Components

In addition to drivers for the internal peripherals, the BSP may include drivers for external components. Freescale MQX release includes several external drivers for components available on the Freescale Tower System or other development boards. These can be modified or used as a reference for drivers needed in the custom BSP. Some external drivers included in the Freescale MQX release are:

- Ethernet PHY examples located in **\mqx\source\io\enet\phy**
- External Flash examples located in **\mqx\source\io\flashx**
- Segment LCD examples located in **\mqx\source\io\lcd**
- NAND flash examples located in **\mqx\source\io\nandflash\nand_devices**

8.3 Add/Remove Driver Source files in BSP Project

When porting to a different MCU derivative, some peripherals may be added or removed. These driver source files can be added or removed from the BSP project as needed. For example, if the original K60 BSP is ported to a K40 derivative, the ENET Ethernet drivers can be removed and the Segment LCD drivers can be added.

9 BSP Memory Map and Linker files

When changing MCU derivatives, frequently the memory sizes and sometimes the memory addresses change. The memory map used by the BSP must be updated to reflect these changes. The BSP macros related to the memory map are in the file **<Board_Name>.h**, <Board_Name>.h File. In addition, the memory map in the linker files may need to be changed. The linker files are located in **\mqx\source\bsp\<Board_Name>\<toolchain>**. See the tool chain documentation for the details and syntax for the linker files.

10 Post-Link Batch Files

MQX uses batch files that are run after any library is built and copies the header files needed by the application to the **\lib** folder. When using the BSP Cloning Wizard, the custom BSP batch files are also cloned, and may not need to be modified. But if drivers are added or removed from the BSP, then the BSP batch file needs to be modified. The PSP batch file also needs to be modified if the MCU derivative header file is changed. The batch files are located here:

- BSP - **\mqx\build\bat\bsp_<Board_Name>.bat**
- PSP - **\mqx\build\bat\psp_<Board_Name>.bat**

11 CodeWarrior Debugger Memory File

If using the CodeWarrior debugger with the BSP, some configuration files are included in the BSP to configure the debugger. TCL files are used to initialize the Kinetis debugger connection, but these should not need to be modified. However, the .MEM file configures the memory map for the debugger and may need to be modified if the memory map of the MCU in the custom BSP is different than the original MCU. The file is **\mqx\source\bsp\<Board_Name>\<toolchain>\dbg\<Board_Name>.mem**. This is a text file, and is simple to modify manually. Another option is to use the .MEM file provided by CodeWarrior and use the files located in **<CodeWarrior_Installation_Path>\MCU\ARM_EABI_Support\Memory_Config_Files**.

Note that this step is not needed for IAR or Keil because they are configured by selecting the derivative in the project options.

12 Porting Example Applications

The example projects included in the MQX release offer a quick way to test the changes in the customized BSP. When using the BSP Cloning Wizard to clone the BSP, the tool offers the option to clone the example projects. This is a useful feature because the tool also ports all the example projects to the new BSP name. Those examples can be built to ensure there are no build errors in the libraries and application projects and the projects can run on the custom target hardware.

If the MCU derivative was changed in the BSP, change the tool chain setting for the MCU derivative in the application example project. Most tool chains require that the project specify the MCU derivative which is used for the debugger and the flash programmer. In CodeWarrior, the target type is changed in the connection for the debug configuration.

13 Conclusion

Freescale MQX is a scalable RTOS that complements Freescale's broad family of micro-controllers including Kinetis. The MQX BSPs which are provided can be ported across all supported Freescale devices using the steps outlined in this guide. In addition to the MQX documentation, Freescale also provides tools to assist with the porting process including the MQX BSP Cloning Wizard, Processor Expert, and the Solution Advisor tool. For the most up-to-date MQX information and documentation or to download MQX, visit the freescale.com/mqx.

14 Appendix A: Find Closest MQX BSP

Form last updated 11/5/2013

Use this form to find the closest MQX BSP to clone based on the Kinetis part number. See Selecting closest Freescale BSP to clone for more details. Enter the full Kinetis part number in the field below, and click the look-up button. Be sure to enter the exact part number with no additional characters or white spaces. This form uses JavaScript and works with the latest version of Adobe Reader. Ensure that the JavaScript is enabled in Adobe Reader.

Enter Full Kinetis Part Number:

Closest MQX BSP Board Name