Document Number: MQXRN
Rev 4.1.1, 08/2014

Freescale MQX™ RTOS Release Notes

Contents

1 Read Me

This document is the release notes for the Freescale MQX™ RTOS product released for Freescale Kinetis, Vybrid, and ColdFire processor families.

1.1 Development Tools Requirements

Freescale MQX RTOS was compiled and tested with these development tools:

- Kinetis Design Studio version IDE 1.1.0
  - Support available for Kinetis devices
  - See build projects in the Kinetis Design Studio subdirectory
  - Makefile build option: TOOL=kds
- CodeWarrior Development Studio for Microcontrollers version 10.6
  - Support available for Kinetis and ColdFire devices
  - See build projects in the cw10gcc subdirectory
  - Makefile build option (Kinetis GCC only): TOOL=cw10gcc
- IAR Embedded Workbench® for ARM® Version 7.10.3
  - Support available for Kinetis and Vybrid devices
  - See build projects in iar subdirectories
  - Makefile build option: TOOL=iar

- ARM Development Studio 5 (DS-5™) version 5.18.0
- DS-5 Starter Kit for Vybrid Controllers version 5.18.0
- DS-5 Vybrid Controller Edition 5.18.0
  - Support available for Vybrid devices
  - See build projects in the ds5 subdirectory
  - Makefile build option: TOOL=ds5
- ARM-MDK - Keil µVision version 5.10
  - Support available for Kinetis devices
  - See build projects in the uv4 subdirectories
  - Makefile build option: TOOL=uv4
- GNU Tools for ARM Embedded Processors version 4.8-2014-q1-update
  - Support available for Kinetis and Vybrid devices
  - Makefile build option: TOOL=gcc_arm
- Makefile support (mingw32-make version 3.8.2)
  - Library makefiles are located in `<MQX_root_dir>/build/make/<board>`
  - Application makefiles are located in `<example_dir>/make/<board>`

1.2 System Requirements

System requirements are based on the requirements for the development tools. There are no special host system requirements for hosting the Freescale MQX RTOS distribution itself.

Minimum PC configuration:

- As required by Development and Build Tools

Recommended PC configuration:

- 2 GHz processor – 2 GB RAM - 2 GB free disk space

Software requirements:

- OS: Windows® 7 or later
- OS: Ubuntu 13.10 or later

1.3 Target Requirements

Freescale MQX RTOS supports the evaluation boards mentioned below. There are no special requirements for the target hardware other than what each board requires for its operation (power supply, cabling, jumper settings, etc.). For more details about the board-specific setup for MQX applications, see *Getting Started with Freescale MQX™ RTOS*.

Evaluation boards supported:

- Vybrid
  - TWR-VF65GS10 Development Kit
  - AutoEVB Vybrid Evaluation Board
- Kinetis
  - FRDM-K64F Freescale Freedom platform
  - TWR-K20D50M Development Kit
  - TWR-K20D72M Development Kit
  - TWR-K21D50M Development Kit
  - TWR-K21F120M Development Kit
  - TWR-K40X256 Development Kit
  - TWR-K40D100M Development Kit

    - TWR-K53N512 Development Kit
    - TWR-K60N512 Development Kit
    - TWR-K60D100M Development Kit
    - TWR-K60F120M Development Kit
    - TWR-K64F120M Development Kit
    - TWR-K70F120M Development Kit
    - KwikStik - K40X256 based Evaluation Board
- ColdFire
    - TWR-MCF51JF Development Kit
    - TWR-MCF52259 Development Kit
    - TWR-MCF54418 Development Kit

1.4 Set up installation instructions and technical support

Run the self-extracting installer and proceed according to the instructions on the screen. If not specified otherwise, the MQX RTOS package will be installed to the `C:\Freescale\Freescale_MQX_4_1` directory. It is recommended to install MQX RTOS to a path without spaces to avoid build problems with certain tools.

NOTE

Since version 4.0, the pre-built libraries are not distributed in the MQX RTOS release package, which makes it necessary to compile all MQX RTOS libraries for a particular board before the first use. For detailed build instructions, see the Building the MQX RTOS Libraries section in *Getting Started with Freescale MQX™ RTOS*.

For a description of available support including commercial support options, please visit: Freescale MQX Support.

2 What is New

This section describes the major changes and new features implemented in this release.

- These Board Support Packages are added:
    - FRDM-K64F
    - TWR-K64F120M
- MQX RTOS 4.1.1 enables development on Linux machines (tested on Ubuntu 13.10). MQX RTOS is distributed in two variants, classic Windows® installer and Linux distribution in form of zip package. The Linux package supports these features:
    - GCC make
    - KDS
    - DS5
    - GDB debugging
- MQX RTOS 4.1.1 supports the new Freescale IDE, Kinetis Design Studio. The Kinetis Design Studio is a free development tool based on Eclipse and GCC.
- Example ISR and related documentation is updated. The example demonstrates various interrupt handling approaches for MQX RTOS.
- DHCPv6 client is modified to pass all TAHI tests.
- Power PC (PX) family support has been dropped. MQX RTOS version 4.1.0 is the last version with PX support.
- Readme files are added to examples without a description.
- Register definition header files for Kinetis MCUs (from CodeWarrior10.6/Processor Expert) are added into the MQX RTOS 4.1.1, for convenience when porting to other MCU family members.
- Both interrupt driven I2C master and slave mode drivers are changed to support synchronous blocking mode and share the same API with the polling driver variant. This way users can interchange the polling and interrupt I2C driver easily in their final application.

Resolved issues:

- MQX-1486 Fixed hard fault during allocation/de-allocation. Problem was caused by un-handled concurrent access to shared memory pool variables.
- MQX-1488 CodeWarrior for ColdFire compilation issue in the release target was resolved with a workaround in the io_copr.c file.
- MQX-1497 TWR-K60F120M BSP library build fails when the Processor Expert is enabled. The issue was solved by including the mk60f12.h header file in the project.
- MQX-1398 The CDC USB host class bug introduced by NIO porting was resolved. Note that CDC has still a known issue related to the buffer size length.
- MQX-1346 Data written by the FlashX driver could not be read back immediately in high optimization setting. The instruction and data barriers are added to avoid this problem.
- RTCS bugs, HTTPSRV session timeout and UDP send bug, are fixed.
- MQX-1031 Vybrid-TWR DMA channels were in conflict with U-Boot. The issue prevented MQX RTOS 4.0.2 from running on Vybrid Cortex-M4 core and Linux running on the Cortex-A5 core at the same time. This conflict is resolved.
- MQX-1445 Vybrid Cortex-A5 cache invalidate function issue is resolved.
- MQX-1465 LED2 and LED3 definition is added into the FRDM-K64F BSP which means that all three LEDs have their own definitions in the BSP header. This allows lowpower and freq_change examples to address clock frequency changes by flashing the LED2.
- MQX- 3669 Lowpower example issue on K70F120M is fixed. MCG_C6 configuration is updated.
- MQX-4108 FFS NAND driver halves physical memory issue is fixed. The actual algorithm prepares enough reserved blocks for a backup algorithm. The mfs_nandflash example is updated.
- MQX- 4175 To avoid build issue when the MQX_USE_IPC is set to zero and the MQX_DISABLE_CONFIG_CHECK is not defined, the header includes for IPC are wrapped by the MQX_IS_MULTI_PROCESSOR macro.
- MQX-4216 The FFS NAND Flash FFlush malfunction issue is fixed. The IO_IOCTL_FLUSH_OUTPUT ioctl is introduced to flush media on FFS NAND memory and make sure that the flush process is complete. The low-level flush on MFS is implemented when closing the file handler.
- MQX-1488 PE_demo example issue in the TWR_MCF51JF release target is fixed. The CodeWarrior compiler could not handle code optimization properly. The optimization level for the io_dopr.c file is limited to level 0.

3 Release Content

This release contains these directories:

Table 1. Release Contents

Deliverable	Location
Configuration Files and Mass-Build Projects	`<install_dir>/config/...`
Configuration and mass-build project for all supported boards	`.../config/<board>`
MQX PSP, BSP Source Code, and Examples	`<install_dir>/mqx/...`
MQX PSP source code for Kinetis/Vybrid ARM Cortex-M core	`.../mqx/source/psp/cortex_m`
MQX PSP source code for Kinetis/Vybrid ARM Cortex-A core	`.../mqx/source/psp/cortex_a`
MQX PSP source code for ColdFire	`.../mqx/source/psp/coldfire`
MQX PSP build projects	`.../mqx/build/<compiler>/psp_<board>`
MQX BSP source code	`.../mqx/source/bsp/<board>`
MQX BSP build projects	`.../mqx/build/<compiler>/bsp_<board>`
RTCS source code and examples	`<install_dir>/rtcs/...`

Table continues on the next page...

Table 1. Release Contents (continued)

RTCS source code	.../rtcs/source
RTCS build projects	.../rtcs/build/<compiler>/rtcs_<board>
RTCS example applications	.../rtcs/examples
MFS source code and examples	<install_dir>/mfs/...
MFS source code	.../mfs/source
MFS build projects	.../mfs/build/<compiler>/mfs_<board>
MFS example applications	.../mfs/examples
USB Host driver source code and examples	<install_dir>/usb/host/...
USB Host source code and class drivers	.../usb/host/source
HUB Class Driver	.../usb/host/source/classes/hub
Audio Class Driver	.../usb/host/source/classes/audio
Personal Healthcare Device Class (PHDC) Driver	.../usb/host/source/classes/phdc
Human Interface Device (HID) Class Driver	.../usb/host/source/classes/hid
Mass Storage (MSD) Class Driver	.../usb/host/source/classes/msd
Printer Class Driver	.../usb/host/source/classes/printer
CDC Class Driver	.../usb/host/source/classes/cdc
USB Host build projects	.../usb/host/build/<compiler>/ usbh_<board>
USB Host example applications (HID, MSD, HUB)	.../usb/host/examples
USB Device drivers source code and examples	<install_dir>/usb/device/...
USB Device source code	.../usb/device/source
USB Device build projects	.../usb/device/build/<compiler>/ usbh_<board>
USB Device example applications (HID, MSD, CDC, PHDC)	.../usb/device/examples
Shell Library Source Code	<install_dir>/shell/...
Shell source code	.../shell/source
Shell build projects	.../shell/build/<compiler>/ shell_<board>
Build tools plug-ins	<CodeWarrior_dir>/...
FFS source code and examples	<install_dir>/ffs/…
FFS project	ffs/build/<compiler>/<board>
FFS source code	ffs/source
MFS on FFS example	ffs/examples/mfs_nandflash
Keil Task Aware Debugging plugin (TAD)	.../tools/keil_extensions/
IAR Task Aware Debugging plugin (TAD)	.../tools/iar_extensions/
PC Host tools	<install_dir>/tools
BSP cloning wizard	…/tools/BSPCloningWizard/ BSPCloningWizard.exe
TFS Make Utility	…/tools/mktfs.exe
Check for Latest Version tool	…/tools/webchk.exe
AWK interpreter (GNU General Public License)	…/tools/gawk.exe
SNMP code generation scripts	…/tools/snmp/*.awk

Table continues on the next page...

Table 1. Release Contents (continued)

Timing HTML report tool (for mqx/examples/benchmrk/timing)	…/tools/timing.exe
Code size HTML report tool (for mqx/examples/benchmrk/codesize)	…/tools/codesize.exe
TAD string and configuration files	…/tools/tad
Demo Applications	<install_dir>/demo
Various demo applications demonstrating complex MQX functionalities.	…/demo/...
Documentation	<install_dir>/doc
User Guides and Reference Manuals for MQX RTOS, RTCS, MFS, I/O Drivers, USB etc.	…/doc

This figure shows the Freescale MQX RTOS directories installed to the user's host computer (subdirectories reduced for clarity):

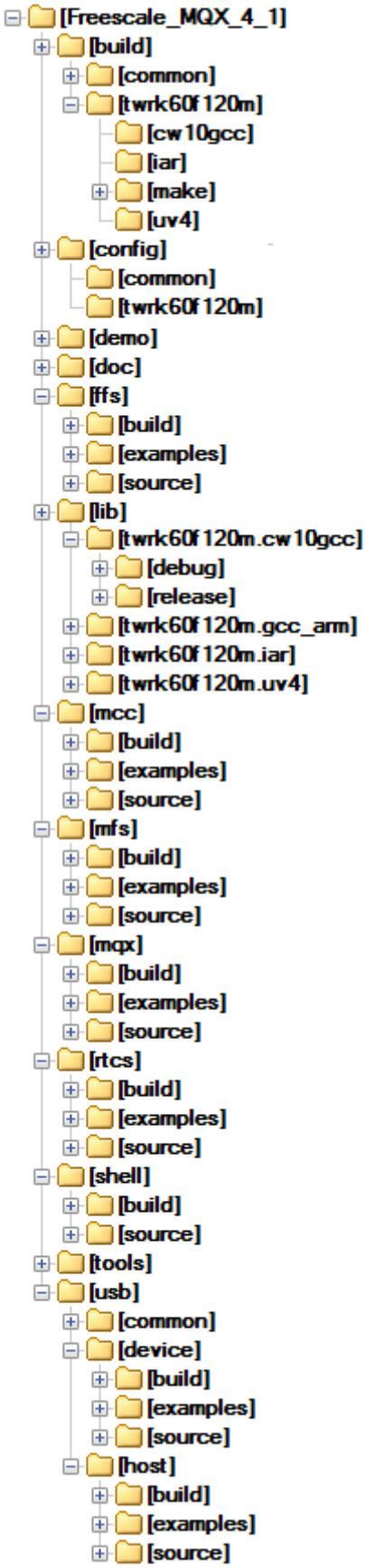

Figure 1. Freescale MQX RTOS Directories

4 MQX RTOS Release Overview

The Freescale MQX RTOS is intended for Kinetis, Vybrid, and ColdFire Microcontrollers. The release consists of:

- MQX real time kernel and system components
- TCP/IP networking stack (RTCS)
- FAT file system (MFS)
- NAND flash file system (FFS)
- USB Host and Device stacks
- Platform and Board support packages
- I/O drivers

This table shows the availability of various components and I/O drivers on supported boards.

	MQX PSP+BSP Libraries	MFS Library (FAT File System)	RTCS Library (TCP/IP Stack)	Shell Library	USB Host Library	USB Device Library	Flash File System (NAND)	UART (polled and interrupt driven	I2C (polled and interrupt driven)	SPI	LWGPIO	HW Timer (PIT, Systick, GPT)	LWADC	Audio driver I2S or SAI	QuadSPI	FLASHX	NAND flash driver	ESDHC	Compact Flash Card driver	SD Card driver (SPI or SDHC based)	RTC, IRTC (Real Time Clock)	TSS - Touch Sensing	DCU	FlexCAN / msCAN	Ethernet driver	USB Host/Device driver
Vybrid																										
TWRVF65GS10_A5	●	●	●	●	●	●	●	●	●	●	●	●	●		●		●	●		●	●		●		●	●
TWRVF65GS10_M4	●	●	●	●	●	●	●	●	●	●	●	●	●		●		●	●		●	●				●	●
AUTOEVB_VYBRID_A5	●	●	●	●	●	●		●	●	●	●	●	●				●	●		●	●		●		●	●
AUTOEVB_VYBRID_M4	●	●	●	●	●	●		●	●	●	●	●	●				●	●		●	●				●	●
Kinetis																										
FRDM-K64F	●	●	●	●	●	●		●	2)	●	●	●	●			●	●	●	●	●	●			●	●	●
TWRK20D50M	●	●	1)	●	●	●		●	2)	●	●	●	●			●		●		●	●			●		●
TWRK20D72M	●	●	1)	●	●	●		●	2)	●	●	●	●			●		●		●	●			●		●
TWRK21D50M	●	●	1)	●	●	●		●	2)	●	●		●	●				●		●	●			●		●
TWRK21F120M	●	●	1)	●	●	●		●	2)	●	●	●	●			●		●		●	●			●		●
TWRK40X256	●	●	1)	●	●	●		●	2)	●	●	●	●	●		●		●		●	●	●		●		●
TWRK40D100M	●	●	1)	●	●	●		●	2)	●	●	●	●			●		●		●	●	●		●		●
TWRK53N512	●	●	●	●	●	●		●	2)	●	●	●	●	●		●		●		●	●	●		●	●	●
TWRK60D100M	●	●	●	●	●	●		●	2)	●	●	●	●			●		●		●	●			●	●	●
TWRK60F120M	●	●	●	●	●	●	●	●	2)	●	●	●	●			●		●		●	●			●	●	●
TWRK60N512	●	●	●	●	●	●		●	2)	●	●	●	●	●		●		●		●	●	●		●	●	●
KwikStik (K40)	●	●	1)	●	●	●		●	2)	●	●	●	●			●		●		●	●	●		●		●
TWRK64F120M	●	●	●	●	●	●		●	2)	●	●	●	●			●	●	●	●	●	●			●	●	●
TWRK70F120M	●	●	●	●	●	●	●	●	2)	●	●	●	●			●		●		●	●	●		●	●	●
ColdFire																										
TWRMCF51JF	●	●	1)	●	●	●		●	●	●	●		●			●			●	●	●	●				●
TWRMCF52259	●	●	●	●	●	●		●	●	●	●		●			●			●	●	●			●	●	●
TWRMCF54418	●	●	●	●	●	●	●	●	●	●	●			●			●	●	●	●	●				●	●

Figure 2. MQX Release Overview

NOTE

- Items in red represent new features in this release

1. Onchip Ethernet not available, RTCS can be used with PPP or custom ENET driver (e.g Wi-Fi over SPI)
2. The I2C driver on Kinetis platforms supports the synchronous blocking mode in both interrupt and polling mode driver variants.

4.1 MQX RTOS PSP

Freescale MQX RTOS release contains ARM Cortex-M, ARM Cortex-A, and ColdFire Platform Support Packages. Contact Freescale Support for other Freescale platforms.

The platform-specific code from `/mqx/source/psp/<platform>` is built together with the generic MQX core files. These two parts form a static library generally referred to as "PSP" which enables the target application to access RTOS features.

4.2 MQX RTOS BSPs

Freescale MQX RTOS release includes Board Support Packages for the boards mentioned in Target Requirements.

The board-specific code from `/mqx/source/bsp/<board>` is built together with I/O driver files from `/mqx/source/io`. These two parts form a static library generally referred as "BSP". The functions included in this library enable the board and operating system to boot up and use the I/O driver functions.

The following section describes drivers supported by the MQX BSPs.

4.3 I/O drivers supported

The following list describes I/O drivers available in the latest MQX RTOS release. The drivers are an optional part of the MQX RTOS and their installation can be enabled or disabled in the BSP startup code. To provide the optimal code and RAM application size, most of the drivers are disabled in the `/config/<board>/user_config.h` file by default. The drivers required by demonstration applications (in the /demo folder) are enabled by default.

See *MQX™ RTOS I/O Drivers User's Guide* (document MQXIOUG) for details.

NOTE

When BSPCFG_ driver-enabling macros are missing in the `/config/<board>/user_config.h` file, the default setting is taken from the BSP-specific header file located in the `/mqx/source/bsp/<board>/<board>.h`. The user decides whether to enable the automatic installation of the driver in the BSP startup code (by enabling the appropriate BSPCFG_ENABLE_XXX macro in the `user_config.h`), or manually in the application code.

TFS – Trivial Filesystem

Trivial Filesystem is used as a simple read-only file repository instead of the fully featured MFS. TFS is not installed in the BSP startup code. Applications must initialize the TFS and pass a pointer to the filesystem data image. The **mktfs** tool is available (both as executable and Perl script) to generate the image from the existing directory structure. The RTCS HTTP example demonstrates the use of TFS.

I2C I/O Driver

This driver supports the I2C interface in both master and slave mode. If enabled in user configuration, the I2C driver is installed during the BSP startup code as the "i2cx" in polled mode and as the "ii2cx" in interrupt mode where "x" stands for a specified I2C channel number). Example applications are provided in the MQX RTOS source tree for both master and slave mode.

I2S and SAI I/O Driver

This driver supports an I2S interface in a master mode. If enabled in user configuration, the I2S device driver is installed during the BSP startup code as “i2s0:”. An example application is provided in the MQX RTOS source tree.

SPI I/O Driver

This driver supports the operation master mode. If enabled in user configuration, the SPI device drivers are installed during the BSP startup code as “spi0:” (or “spiX:” where X is index of the SPI module used). The SPI driver was significantly rewritten in MQX RTOS 4.0, so that there is no distinct interrupt or polled driver type. See *MQX™ RTOS I/O Drivers User's Guide* (document MQXIOUG) for details. On Kinetis and Vybrid platforms, the driver uses DMA to function.

QuadSPI I/O Driver

This driver provides a C language API to the QuadSPI peripheral module. If enabled in user configuration, the QuadSPI device drivers are installed during BSP startup code as "qspi0:" (or "qspiX:" where X is index of QSPI module used). See *MQX™ RTOS I/O Drivers User's Guide* (document MQXIOUG) for details.

FlexCAN Driver

This driver provides a C language API to the FlexCAN peripheral module. An example application is provided in the MQX RTOS source tree.

msCAN Driver

This driver provides a C language API to the msCAN peripheral module. An example application is provided in the MQX RTOS source tree.

RTC Driver

This driver provides a C language API to the Real Time Clock peripheral module and functions, and synchronizes the clock time between RTC and MQX RTOS systems. If enabled in user configuration, the RTC module is initialized and MQX RTOS time is renewed automatically during BSP startup.

Serial I/O Driver

The standard SCI (UART) driver supports both polled and interrupt-driven modes. If enabled in user configuration, the serial devices are installed as “ttya:”, “ttyb:” and “ttyc:” (polled mode) and “ittya:”, “ittyb:” and “ittyc:” (interrupt mode) automatically during BSP startup.

GPIO I/O Driver (obsolete)

Support of this driver has been discontinued in Freescale MQX RTOS. This driver is replaced by the LWGPIO driver for all supported platforms.

LWGPIO I/O Driver

This driver provides a C language API to all GPIO ports available on a particular device. It is significantly faster and has a smaller footprint than the GPIO driver.

ADC Driver (obsolete)

This I/O driver provides a uniform interface to ADC channels. This driver has been replaced by LWADC I/O driver.

LWADC I/O Driver

This driver provides a C language API to ensure a uniform access to ADC peripheral basic features.

DAC Driver (obsolete, driver provided by Processor Expert driver suite now)

The full version of this driver is provided as part of the Processor Expert driver suite. This driver provides C language API to DAC peripheral module. It is adopted from the Freescale Processor Expert toolbox. The DAC driver is installed and used directly from the application code.

Flash I/O Driver

This I/O driver provides a standard interface to either internal or external Flash memory. If enabled in user configuration, the Flash driver (called FlashX) is installed as “flashx:” device automatically by the BSP startup code. Note that “flash0”, “flash1” etc. device names are used for FlashX devices installed for external Flash memory. For devices with internal Flash memory, the FlashX driver depends on several parameters passed in a form of global symbols from an application or from a Linker Command File. For more information, see driver installation code in the BSP and an example application provided in the MQX RTOS source tree.

ENET Driver

The low-level Ethernet driver is used by the RTCS TCP/IP software stack. The driver is initialized directly by the application before RTCS is used for the first time. The RTCS Shell and HTTP examples demonstrate the use of this driver.

PCCard I/O Driver

This I/O driver provides a low-level access to the PCCard functionality by using Flexbus and CPLD circuit. The CPLD code can be found in the `<install_dir>/mqx/source/io/pccard/<card_name>`. If enabled in the user configuration, the PCCard device driver is installed as “pccarda:” automatically during the BSP startup.

PCFlash I/O Driver

The Compact Flash Card I/O driver is installed on top of the PCCard low-level driver and enables standard disk drive operations. The MFS file system can be installed on top of this device. If enabled in user configuration, the PCFlash device driver is installed as “pcflasha:” automatically during the BSP startup.

SD Card I/O Driver

This I/O driver implements a subset of the SD protocol v2.0 (SDHC). The driver can use either the MQX RTOS SPI driver or the MQX RTOS (e)SDHC driver to communicate with the SD Card device. Install the driver at the application level, and pass a lower-layer driver handle to it. The MFS file system can be installed on top of this device.

(E)SDHC I/O Driver

This I/O driver covers the (e)SDHC peripheral module and provides low-level communication interface for various types of cards including SD, SDHC, SDIO, SDCOMBO, SDHCCOMBO, MMC, and CE-ATA.

Resistive Touch-Screen Driver

This I/O driver accesses the ADC and GPIO modules to detect touch events and acquire touch coordinates on a resistive touch-screen unit.

DIU Display Driver

This driver provides a generic C language API to frame buffer-based display units. It is initialized and used from a user-application. Since the support for MPC5125 BSP was removed in the MQX RTOS 4.0, this driver is not currently used by any BSP.

I/ODebug Driver

This driver redirects I/O functions, such as printf, to a debug probe-based communication channel. The CodeWarrior 10, IAR EWARM, or Keil µVision debugger consoles are supported. See *Getting Started with Freescale MQX™ RTOS* for details about the setup and use of this feature.

HWTimer Driver

This driver provides a C language API for uniform access to the features of various HW timer modules such as PIT and SysTick.

DMA Driver

This driver provides the C language API and essential functionality to control the DMA peripheral module.

FTM Quadrature Decoder Driver

This driver provides API related to the FTM quadrature decoder functionality which is implemented on Vybrid only.

I/O Expander Driver

This driver controls an off-chip I/O expander device and provides a convenient interface for individual pin handling. Currently, it only supports the MAX7310 device.

4.4 Default I/O Channel

An I/O communication device installed by MQX BSP can be used as the standard I/O channel. See *Getting Started with Freescale MQX™ RTOS* for the default console setting for each supported development board.

4.5 MQX RTOS PSP and BSP Directory Structure

RTOS files are located in the mqx subdirectory of the Freescale MQX RTOS installation. The directory structure is shown in this image.

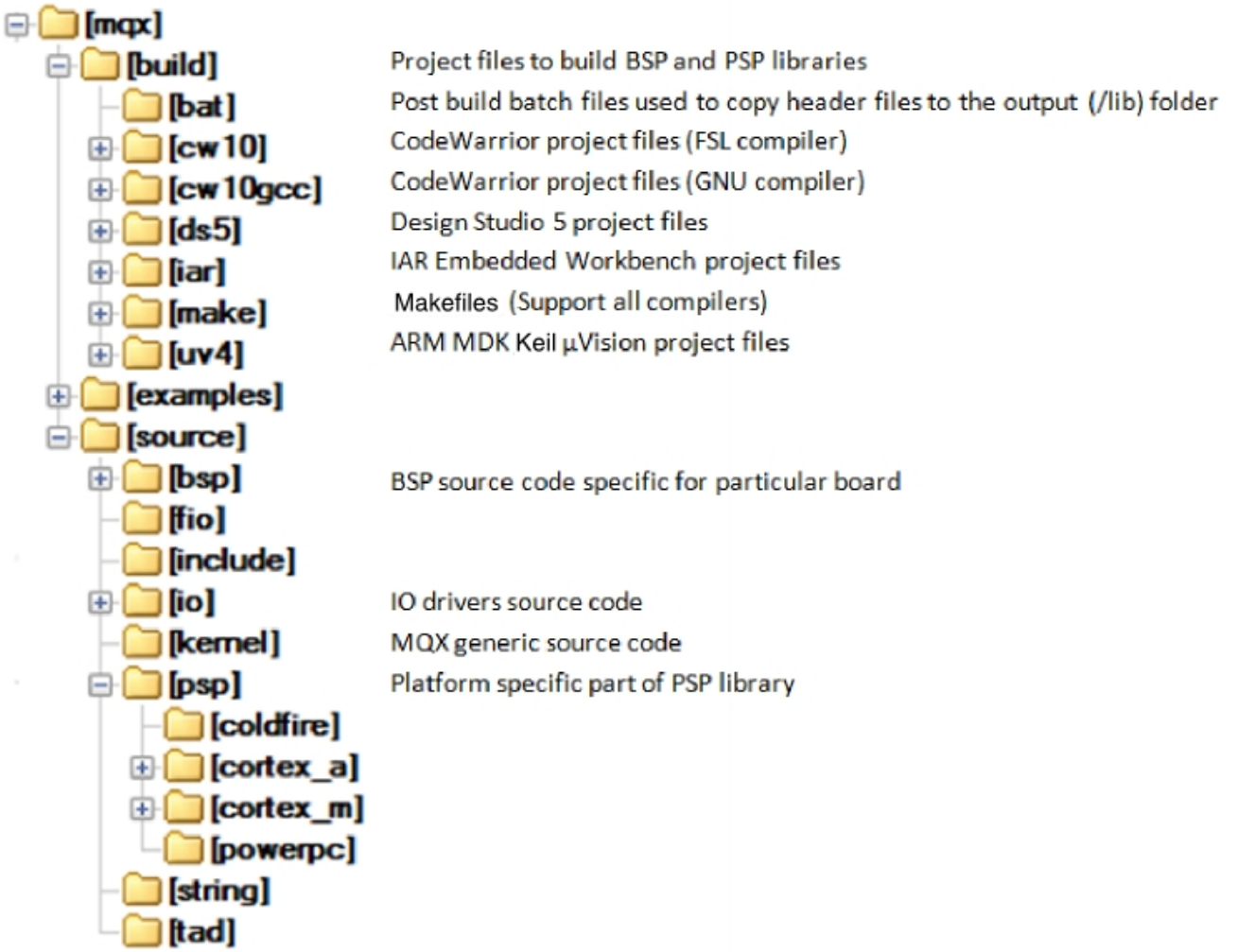

Figure 3. MQX PSP and BSP Directory Structure

4.6 MFS for MQX RTOS

MFS files from the `/mfs/source` directory are built into a static library. When linked to the user application, the MFS library enables the application to access FAT12, FAT16, or FAT32-formatted drives.

4.7 RTCS for MQX RTOS

RTCS files from the `/rtcs/source` directory are built into a static library. When linked to the user application, the RTCS library enables the application to provide and consume network services of the TCP/IP protocol family.

The MQX RTOS 4.1.1 RTCS stack is IPv6 ready with respect to IPv6 Ready Logo certification and has passed all required tests. IPv6 support is available as a separate update package available from Freescale.

4.8 USB Host for MQX RTOS

Freescale MQX RTOS release includes the USB Host drivers and USB class drivers. The USB HDK (Host Development Kit) files from the `/usb/host/source` directory are built into a static library. When linked to the user application, the USB HDK library enables the application to communicate with various USB devices attached on the USB bus.

The HDK contains the following USB class drivers:

- USB Hub class used to attach multiple devices to a single host port. If enabled at the application level, the HUB support is fully transparent. Only the user application needs to be modified to handle multiple USB devices simultaneously. A Keyboard/Mouse example application is provided.
- Human-interface Class (HID) used to access mouse, keyboard, and similar devices.
- Mass storage device (MSD) Class used to access USB drives.
- Communication Device Class (CDC) used as a serial communication device implementing virtual “tty” ports.
- Audio Class.
- Basic Printer Class.

4.9 USB Device for MQX RTOS

Freescale MQX RTOS release includes the USB Device drivers and example applications implementing various USB devices. The USB DDK (Device Development Kit) files from the `/usb/device/source` directory are built into a static library. When linked to the user application, the USB DDK library enables the application to act as a USB device supporting one or more of the following classes:

- HID (mouse functionality demonstrated)
- MSD (internal RAM area accessed as mass storage device)
- CDC COM (virtual serial line implementation)
- CDC NIC (virtual network interface card implementation)
- PHDC (medical applications)
- Audio

4.10 MQX RTOS Shell

The shell and command-line handling code is implemented as a separate library called Shell.

4.11 Changing the MQX RTOS source files

The Freescale MQX RTOS is distributed in source code form. Do not modify any of the source files other than the compile-time configuration files. This recommendation applies to all files under “source” and “build” sub-directories in all MQX RTOS, RTCS, MFS, USB, and other core components folders.

If you are creating custom board support packages or adding additional I/O drivers, add the new files and subdirectories to the following directories:

```
<install_dir>/mqx/source/bsp
<install_dir>/mqx/source/io
```

4.12 Building the MQX RTOS libraries

For more details about building MQX RTOS libraries and applications, see *Getting Started with Freescale MQX™ RTOS*.

When using MQX RTOS for the first time and making changes to the compile-time user configuration file or MQX kernel source files, rebuild MQX RTOS libraries to ensure that the changes are propagated to the user applications.

4.13 Example applications

Demo applications are in this directory:

```
<install_dir>/demo
```

The examples are written to demonstrate the most frequently used features of the Freescale MQX RTOS.

In addition to these demo applications, there are simpler example applications available in MQX, RTCS, MFS, and USB directories.

The tables summarize all demo and example applications provided in this release.

MQX Example Applications

Table 2. `mqx/examples`

Name	Description
access_usr	Demonstration of memory protection between the user and privilege tasks.
benchmrk	Contains benchmarks codes for timing and code size for different components.
bootloader	Contains basic functions for boot loader application.
bootloader_vybrid	Shows how to load application images from FAT system on SD card or flash QSPI boot loader and application images to QSPI Flash memory with SD boot loader.
bootloader_vybrid_qspixip	Shows how the QSPI boot loader boots from the QSPI flash memory and loads application images from FAT file system on SD card or the raw image data from the flash memory.
can/flexcan	Shows usage of FlexCAN API functions to transmit and receive CAN frames.
clkapi	Shows usage of the clktree management APIs.
cplus	Shows simple C++ application.

Table continues on the next page...

Table 2. `mqx/examples` (continued)

Name	Description
dcu4	Shows the use of the DCU driver functionality on multiple layers: display, event handling, timing update, and alpha blending.
demo	Shows MQX multitasking and inter-process communication using standard objects like semaphores, events, or messages. See lwdemo for the same example using the lightweight objects.
esai_asrc	Shows how to play back/record by using ESAI and how to enable ASRC for sample rate conversion while playing ESAI audio.
event	Simple demonstration of MQX events.
fbdev	Shows the use of the abstract FBdev driver functionality on the frame buffer configuration, multiple buffers flipping, and VSYNC waiting.
flashx	Demonstration of FlashX driver functionality.
flashx_swap	A demonstration of FlashX driver's swap and reset functionality.
flexnvm	Demonstrates use of the Flex NVM functionality on Kinetis platforms where this feature is supported.
freq_change	Show dynamic frequency change for Kinetis and ColdFire+ platforms only.
ftm	Demonstrates usage of the FTM Quadrature Decoder Driver.
gpio	Shows usage of LWGPIO driver to control on-board LEDs and switches.
hello	A trivial Hello World application using a single task.
hello2	A trivial Hello World application spread across two tasks.
hmi	Demonstrates integration of the MQX application and the TouchSensing library.
hwtimer	Shows usage of HW timer driver abstraction. Demonstrates how to initialize HW timer for various modules, set frequency, callback, start, and stop the timer.
i2c	Shows how to read/write data from/to external EEPROM. Additional HW setup is needed.
i2c_slave	Shows usage of the I2C driver in slave mode – emulates the external EEPROM behavior. The current version of the MQX I/O driver supports two I2C IP modules. The legacy module requires an additional configuration. To confirm which IP you are using, see the figure in the MQX Release Overview section.
i2s_demo	Demonstrates use of audio I2S driver. TWR-AUDIO card is needed to run this example.
io	Demonstrates use of an alternate UART port as a console output.
io_expander	Shows how to operate a pin on the I/O Expander chip.
ipc	UART-based inter-processor communication demonstration.
irda	Demonstrates use of IrDA driver for transmit and receive.
isr	Shows how to install an interrupt service routine and how to chain it with the previous handler.
klog	Shows kernel events being logged and later the log entries dumped on the console.
log	Shows the application-specific logging feature.
lowpower	Shows how to switch between several predefined low-power operation-modes.
lowpower_vybrid	Shows Vybrid circling through different power modes (RUN / SLEEP / WAIT / STOP).
lwadc	Shows usage of the ADC driver, sampling analog values from the two ADC channels.
lwdemo	Same as the "demo" application, but implemented using lightweight components only.
lwdemo_usr	Shows MQX multitasking and inter-process communication using user mode tasks and lightweight objects such as semaphores, events, or messages.
lwevent	Simple demonstration of MQX lightweight events.
lwlog	Simple demonstration of MQX lightweight log feature.

Table continues on the next page...

Table 2. `mqx/examples` (continued)

Name	Description
lwmsgq	Simple demonstration of MQX lightweight inter-process messaging.
lwsem	Simple demonstration of MQX task synchronization using the lightweight semaphore object.
lwsem_usr	Simple demonstration of MQX user/privilege task synchronization using the lightweight semaphore object.
msg	Simple demonstration of MQX inter-process message passing.
multicore	Shows usage of the multicore communication components.
multidrop	Demonstration of the UART-based multidrop.
mutex	Simple demonstration of MQX task synchronization using the mutex object.
nandflash	A demonstration of the NAND Flash driver functionality.
nill	Even simpler than Hello World. A void application which may be used for copy/paste to start custom application.
qspi	Demonstrates basic operation of QuadSPI driver, interfacing to QSPI flash.
rs485	Shows how to use the RS485 over a serial driver.
rtc	Shows the Real Time Clock module API. Demonstrates how to synchronize RTC and MQX time and how to use RTC alarm interrupts.
sai_dma_demo	Shows the use of the DMA driven SAI driver. TWR-AUDIO-SGTL board is needed for this example.
sem	Simple demonstration of MQX task synchronization using the semaphore object.
spi, spi_legacy	Shows how to read/write data from/to external SPI EEPROM. Additional HW setup is needed.
taskat	Shows how task can be created within statically allocated memory buffer (avoid heap allocation for task stack and context).
taskq	Shows custom task queue and how the queue can be suspended and resumed.
tchres	Example applications demonstrating resistive touchscreen functionality with the TWR-LCD board.
test	Shows the self-testing feature of each MQX component.
tfs	Shows the usage of ROM-based Trivial File System in an MQX application.
timer	Simple demonstration of MQX timer component.
usermode	Memory management and dynamic task creation from user-mode tasks.
watchdog	Simple demonstration of the MQX task timeout detection using the kernel (not to be confused with watchdog) component.

RTCS Example Applications

Table 3. `rtcs/examples/...`

Name	Description
eth_to_serial	Simple character passing between the UART console and the telnet session. Shows custom "lightweight" telnet.
httpsrv	Simple web server with CGI-like scripts and web pages stored in internal flash.
shell	Shell command line providing commands for network management.
snmp	SNMP protocol example providing microprocessor state information.

FFS Example Applications

Table 4. `ffs/examples/mfs_nandflash/...`

Name	Description
mfs_nandflash	Console shell-based example showing how to access an MFS filesystem mounted on the NAND flash memory.

MFS Example Applications

Table 5. `mfs/examples/...`

Name	Description
cfcard	Console shell-based example showing the MFS filesystem used with and CFCard storage.
mfs_ftp	RTCS FTP demo accessing the MFS filesystem mounted on the USB mass storage. For an FTP example without the USB functionality, see the RTCS Shell demo.
mfs_usb	Console shell-based example showing how to access MFS filesystem mounted on the USB mass storage.
ramdisk	Shows use of MFS accessing the external RAM (or MRAM).
sdcard	Shows use of MFS accessing the SDHC or SPI-connect SD Card.

USB Host Example Applications

Table 6. `usb/host/examples/...`

Name	Description
audio/microphone	Enables connecting a USB microphone and record the sound to SD Card (wav formatl).
audio/speaker	Enables connecting a USB speaker and play the sound from SD Card (wav format).
cdc/cdc_serial	This example demonstrates the virtual serial port capability with abstract control model. Redirects the communication from CDC device, which is connected to the board, to the serial port.
hid/keyboard	This application echoes keys pressed on the USB keyboard onto the serial console.
hid/mouse	Displays USB mouse events on the serial console.
hid/keyboard+mouse	Keyboard and mouse demos combined in a single application.
msd/msd_commands	Executes the standard "mass storage device" commands to the USB disk and shows the response on the serial console (see MFS examples for USB filesystem access).

USB Device Example Applications

Table 7. `usb/device/examples/...`

Name	Description
audio/generator	Acts as a USB microphone, playing out a short audio loop.
audio/speaker	Receives audio stream data from the host (PC) and plays it out through the I2S driver.
cdc/virtual_com	Implements a virtual serial line loopback.cdc/virtual_nicImplements a virtual network interface cards.
cdc/virtual_nic	Implements a virtual network interface cards.
hid/mouse	Creates a virtual mouse which keeps moving in a square loop, 100 pixels in size.

Table continues on the next page...

Table 7. `usb/device/examples/...` (continued)

Name	Description
msd/disk	Implements small storage device in internal RAM memory.

Demo Applications

Table 8. `demo/...`

Name	Description
hvac	Simple implementation of console-based HVAC with optional USB logging and FTP access.
hvac_error	Intentional error inserted to the HVAC demo code to demonstrate the efficacy of the TAD plug-in.
pe_demo	This example demonstrates how to configure MQX BSP using Processor Expert.
web_hvac	HVAC demo with the HTTP server implementing the GUI. Ajax-based pages demonstrating the advanced use of the HTTP server.

5 Known Issues and Limitations

Performance of Code Running in MRAM

Runtime performance of the MRAM-based targets is approximately 8x degraded when compared to the Flash-based execution. MRAM is an external memory device connected to the ColdFire core by 8-bit data bus with one wait-state generated for each access. In order to fetch one 32-bit value (instruction) from the MRAM memory, four accesses need to occur. Each access inserts one wait-state clock. This behavior is normal by design and applies to other processors using the external memory to store executable code.

Default Kernel Configuration of Small-RAM Devices

The default kernel configuration of small-RAM devices is optimized to run demonstration applications located in the /demo folder. To meet tight RAM constraints, some MQX RTOS or RTCS features are disabled by default. Additionally, some I/O drivers, which are not used by the main demo applications, are disabled. Typically, a compile-time error message occurs if trying to run an example application while the required kernel feature or I/O driver is missing in the library code. To execute some example applications, enable the required features in the `/config/<board>/user_config.h` file and recompile all MQX libraries.

USB Host HUB Examples

HUB class support is enabled in HID example applications. The applications run correctly with the USB device attached either directly or through the hub. However, the example code only handles a single device. A combined Mouse+Keyboard demo handles one mouse and one keyboard simultaneously. The same kind of multiple devices, which are attached through the hub, cannot be used in the example applications.

OSBDM / OSJTAG Firmware Compatibility

The latest versions of the CodeWarrior, IAR, and Keil tools add a new version of the OSBDM / OSJTAG Debugger interface with improved performance and stability. The new interface requires a firmware update. See the instructions provided in the Release Notes which apply to the development tools you are using. The latest OSJTAG firmware, drivers and tools are available on this web page: http://www.pemicro.com/osbdm/index.cfm

Supporting “Hot Device Uninstall” in MQX I/O Subsystem

In the current implementation of the MQX I/O subsystem, the application is responsible for dealing with application tasks which have opened file handles while uninstalling a device driver. A typical demonstration of the problem is USB mass storage handling: When a USB attach event is detected, an application installs the MFS partition manager and MFS file system "device" on top of the USB driver. The application runs tasks, such as shell, which open and access files provided by the MFS filesystem device. When the user unplugs the USB mass storage device, the application has a limited way to detect an opened file before uninstalling the MFS filesystem device. The file I/O functions start reporting errors when accessing the device after it is physically detached. Design the application code so that the tasks close all files affected by the detach event before the MFS filesystem driver can be uninstalled. If there is an attempt to close the MFS handle prior to closing all related files, a sharing violation error is returned. An example application "mfs_usb" demonstrates how to close files by retrying the closing operation of the MFS handle. If a task keeps one or more files open for an extensive time period, use a suitable method to notify it about the ongoing filesystem un-installation. This implementation may add additional application overhead. Work is ongoing on the MQX I/O subsystem to ensure that file operations safely return error states even after the underlying device driver is uninstalled. This enhancement will simplify the application code error recovery.

TWR-MEM Compact Flash Interface Issues

Some Compact Flash cards do not work correctly with TWR-MEM and MQX CF card driver. These are the possible remedies: An issue in the TWR-MEM CPLD code, Rev. A, causes incorrect communication with some types of cards, such as Kingston. A fixed CPLD firmware is available in the `<install_dir>/mqx/source/io/pccard/twr_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. In some cases, the MQX driver incorrectly detects the card in the slot. First, check whether all CF related jumpers are set correctly as described in *Getting Started with Freescale MQX™ RTOS*. If the incorrect behavior persists, connect the two pull-up resistors between the card detect pins (CF_CD1, CF_CD2) and the 3.3V VCC.

Idle Task Required on Kinetis Platforms

The Kinetis kernel, by design, cannot operate without an idle task. The MQX_USE_IDLE_TASK configuration option must be set to 1.

USB EHCI and KHCI Stack Buffer Restrictions

Align the buffers used by KHCI at 4B. Align the buffers used by EHCI at cache line and the size to cache line boundary in the cached area. If the goal is to optimize performance, allocate the buffer used by EHCI in the un-cached memory space.

Flash Cache Disabled on TWR-K60N512 BSP – erratum e2647

A workaround is implemented for erratum e2647 on the MK60N512 based-BSP. The workaround disables the use of the Flash cache for processors revision 1.0 (mask 0M33Z). It fixes sudden application crashes, but results in lower CPU performance (~30% performance reduction). Newer processor mask sets with fixed e2647 erratum are not affected.

TWR-K60N512 BSP 256KB Flash Boundary Issue

During TWR-K60N512 BSP testing, an unexpected application crash occurs when the application code spans over the 256 KB boundary. The issue is related to the silicon revision 1.0 (mask 0M33Z) only. The TWR-K60N512 linker files are prepared so that the code is placed in the lower addresses and constant data is placed in higher addresses in the Flash memory.

ARP Entries Issue

When the board is put into a busy Ethernet environment with many ARP requests, the ARP entries cause memory fragmentation, which leads to RTCSERR_TCPIP_NO_BUFFS when connect() is called.

FlexCan Driver Issues

Several issues are identified during the development of the FlexCAN driver: On TWR-K70F120M board, the TX/RX signals are not routed to the elevator by default and the FlexCAN example does not work. To enable the FlexCAN operation, solder the zero-ohm resistors, R22 and R23, on TWR-K70F120M board. The 10-kbit baudrate doesn't generally work. FlexCAN detects bit0 errors in its own transmitted messages.

ARM MDK Keil µVision Support – Issue Linking the TWR-K40X256 RTCS Library

The Keil µVision linker fails to link the TWR-K40X256 RTCS-based application projects. The linker failure occurs because the Keil linker tries to place the functions which are not used in the final application. This issue was confirmed by ARM and can be solved by using __weak modifier before the definition of the failing function.

Android USB MSD Cannot Be Interfaced

If certain types of Android phones are connected to the system, the attach event is not generated. The issue is currently investigated and will be fixed in a future MQX RTOS version.

User Mode Functionality in CW10 GCC

The User Mode functionality is not supported in the GCC compiler.

MFS Does Not Check Validity of Directory Rename

MFS_Rename_file() function does not check the necessary precondition when renaming a directory. If the directory is renamed to its own subdirectory, the directory becomes inaccessible and lost cluster chains are created.

CodeWarrior 10.3 and 10.4 GCC Compiler Requires Latest Version of New Project Wizard

Linker problems may arise during the compilation of the projects created by the New Project Wizard in CodeWarrior 10.4 and 10.3 GCC. There are two kind of possible issues: explicit linker errors or an incorrectly linked binary, which starts but does not reach the main function.

To fix this issue, either update the New Project Wizard to the newest version (at least 1.1.1) by using the “Help/Install New Software…” menu, or change the command line pattern in the linker section of the project preferences to this:

```
"${ARMSourceryDir}/${COMMAND}" -Xlinker --start-group ${INPUTS} -Xlinker --end-group
${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT}
```

Bool Type Conflict in Processor Expert TWR-MCF51JF BSP and MQX Applications

Starting with MQX RTOS 4.1.0, the standard C99 types (from stdlib.h) have replaced the MQX RTOS types. When using Processor Expert PE_Types.h TWR-MCF51JF header, the bool type declaration from `stdlib.h` is redefined, causing a compiler error (declaration specifier conflict: illegal 'bool' sequence). This workaround can be used until this issue is resolved:

In PE_Types replace `typedef unsigned char bool;` with

```
#ifndef bool
typedef unsigned char bool;
#endif
```

EHCI HUB functionality limitation

The HUB functionality of the EHCI is not fully supported (dynamically attach/detach is not supported). This issue will be fixed in upcoming releases.

UTF8 support in MFS

The UTF8 support in MFS is limited to read-only access and for long file names. The UTF8 support for write access will be implemented in a future release.

DSPI issues related to the DMA usage

When the DSPI uses the eDMA, it may transfer data incorrectly or fail when eDMA is used for another purpose. If the DSPI driver is the only user of eDMA, it should operate correctly. This behavior is a result of the silicon design of the DSPI. DMA usage can be disabled in the DSPI driver by redefining the macro BSPCFG_DSPIx_USE_DMA to 0 in user_config.h.

The TWR-K53N512 BSP does not work for the TWR-K53N512, Rev.C schematics, Rev.B BOM

The TWR-K53N512 BSP code supports the TWR-K53N512, Rev.C schematics, Rev.X3 BOM (700-26694). This development board is populated by the PK53N512CMD100 processor. The BSP does not work for the newer boards populated by the MK53DN512CMD10 processors. A workaround involves using the TWR-K60D100M BSP code for the TWR-K53N512 board.

USB Host CDC function works with limited input.

In the USB Host CDC example, no more than 200 characters are allowed in one line when inputting to the terminal and no more than 160 characters in one line when transferring files. Otherwise, USB Host CDC example may malfunction.

DSPI FIFO length setting

The DSPI driver FIFO length is currently defined as a constant value (DSPI_FIFO_DEPTH = 4) and it is used for all SPI modules on chip. However there exist devices such as K64f and K21F where the FIFO length is different for some SPI modules. Driver currently does support various length per a module. As a workaround it is suggested to decrease FIFO depth to 1 in case channels with smaller FIFO length are used.

How to Reach Us:

Home Page:
www.freescale.com

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

Document Number: MQXRN
Rev. 4.1.1, 08/2014