Freescale MQX™ RTOS 3.4.0
Release Notes

<table>
<thead>
<tr>
<th>PRODUCT:</th>
<th>Freescale MQX™ RTOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT VERSION:</td>
<td>3.4.0</td>
</tr>
<tr>
<td>DESCRIPTION:</td>
<td>Freescale MQX™ RTOS Operating System, version 3.4.0</td>
</tr>
<tr>
<td>RELEASE DATE:</td>
<td>Sep 25th, 2009</td>
</tr>
</tbody>
</table>
How to Reach Us:

Home Page: www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

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

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor 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 Semiconductor 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 Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. ARC, the ARC logo, ARCanal, ARChitect, ARCompact, ARClangent, BlueForm, CASSEIA, High C/C++, High C++, ICON186, MetaDeveloper, MXD, Precise Solution, PreciseBlazeNet, Precise/EDS, Precise/MFS, Precise/MQX, Precise/MQX Test Suites, Precise/RTCS, RTCS, SeeCode, TotalCore, Turbo186, Turbo86, V8 microRISC, V8 microRISC, and VAutomation are trademarks of ARC International. High C and MetaWare are registered under ARC International. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.
Rev. 9
8/2009
Table of Contents

Freescale MQX™ RTOS 3.4.0 Release Notes ........................................................................................................ i

1 Read Me First .................................................................................................................................................. 3
   1.1 Requirements ........................................................................................................................................... 3
   1.2 Special instructions .............................................................................................................................. 4

2 What is New? .................................................................................................................................................. 5

3 Release Content .......................................................................................................................................... 6

4 MQX Release Overview ................................................................................................................................. 9
   4.1 MQX RTOS PSP ....................................................................................................................................... 9
   4.2 MQX RTOS BSPs .................................................................................................................................. 10
   4.3 MQX MFS .............................................................................................................................................. 14
   4.4 MQX RTCS ............................................................................................................................................ 14
   4.5 MQX USB Host .................................................................................................................................... 14
   4.6 MQX USB Device ................................................................................................................................ 15
   4.7 MQX Shell ........................................................................................................................................... 15
   4.8 Compile-time Configuration .................................................................................................................. 15
   4.9 Changing the MQX Source Files ......................................................................................................... 16
   4.10 Building the MQX Libraries .............................................................................................................. 16
   4.11 Example Applications ......................................................................................................................... 20
   4.12 CodeWarrior Task Aware Debugging Plug-in ................................................................................... 24

5 What is Missing? .......................................................................................................................................... 28
   5.1 In MQX PSP ......................................................................................................................................... 28
   5.2 In USB Support ................................................................................................................................... 28
   5.3 Peripheral Drivers ................................................................................................................................. 28
   5.4 PC Host Tools ..................................................................................................................................... 29

6 Known Issues ............................................................................................................................................. 30
   6.1 Performance of Code Running in MRAM ............................................................................................. 30
   6.2 Network Communication Performance and Delayed ACKs ............................................................. 30
   6.3 Default Kernel Configuration of Small-RAM Devices ....................................................................... 30
   6.4 USB Host HUB Examples .................................................................................................................... 31
   6.5 USB Host HUB Functionality with MCF5222x .................................................................................. 31
   6.6 BDM Connection Failures with MCF5223x ....................................................................................... 31
   6.7 ColdFire V1 Code Footprint in Debug Build Targets ........................................................................ 31
   6.8 OSBDM Firmware Compatibility with CodeWarrior 7.1.2 ................................................................ 32
   6.9 Supporting “Hot Device Uninstall” in MQX I/O Subsystem ............................................................... 33

7 Board-specific Information Related to MQX ........................................................................................... 35
   7.1 M52223EVB ....................................................................................................................................... 35
   7.2 M52233DEMO ................................................................................................................................... 35
   7.3 M52235EVB ....................................................................................................................................... 36
   7.4 M52259EVB ....................................................................................................................................... 36
   7.5 M52259DEMO ................................................................................................................................... 37
   7.6 M52277EVB ....................................................................................................................................... 38
   7.7 M54455EVB ....................................................................................................................................... 38
   7.8 TWR-MCF51CN-KIT ........................................................................................................................... 39
   7.9 TWR-MCF52259-KIT .......................................................................................................................... 40

8 Change Log .................................................................................................................................................. 42
1 Read Me First

This release note documents the Freescale MQX™ RTOS version 3.4 released for Freescale ColdFire® processor family.

1.1 Requirements

1.1.1 System Requirements

This Freescale MQX™ RTOS Release was compiled and tested with the following development environments:

- ColdFire V1 devices (MCF51xx): CodeWarrior Development Studio for Microcontrollers Version 6.2 (patch version 6.2.2, Build 9104)
- ColdFire V2-V4 devices (MCF52xx, MCF54xx): CodeWarrior Development Studio for ColdFire Architectures Version 7.1.2 (Build 11)

The system requirements are defined by the development tools requirements. There are no special host system requirements for hosting the Freescale MQX™ RTOS distribution itself.

Minimum PC configuration:
   As required by CodeWarrior Development Studio

Recommended PC configuration:
   2 GHz processor – 1 GB RAM - 1 GB free disk space

Software requirements:
   OS: As required by CodeWarrior Development Studio (Windows XP SP2 or later recommended)

1.1.2 Target Requirements

The Freescale MQX™ RTOS in this release supports the evaluation boards mentioned below. There are no special requirements for the target hardware which would be out of scope of what each board requires for its operation (power supply, cabling, jumper settings etc). Please refer to Section 7 which considers Board-specific Information Related to MQX.

Evaluation boards supported:

ColdFire V1

- TWR-MCF51CN-KIT (Rev.A) which consists of
  - MCF51CN128 “Kitchen” processor board
  - Serial Storey board
  - Two 4-storey elevator boards

ColdFire V2:

- M52223EVB Evaluation Board
- M52233DEMO Evaluation Board
- M52235EVB Evaluation Board
- M52259EVB Evaluation Board
- M52259DEMOKIT Evaluation Board
- TWR-MCF52259-KIT (Rev.A) which consists of
  - MCF52259 “Kitchen” processor board
  - Serial Storey board

Freescale MQX Release Notes

Freescale Semiconductor
1.2 Special instructions

1.2.1 Setup Installation instructions

Run the self-extracting MQX installer application and proceed according to instructions.

In case you change the default installation location (which is C:\Program Files\Freescale\Freescale MQX 3.4) it is recommended to re-compile all core libraries. Otherwise, any time the application is started under debugger, the debugger may ask for a path to MQX source code files.

To re-build the libraries, open and build the following CodeWarrior project:

<install_dir>/config/<board>/build_libs.mcp

See more details about re-building the MQX libraries in the following sections.

Pay attention to use correct CodeWarrior Development Studio when opening any MQX projects. There are two different CodeWarrior Studios for V1 family (MCF51xx) and V2-V4 families (MCF52xx-MCF54xx) of ColdFire microcontrollers.
2 What is New?

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

Version 3.4.0 (Sep 25th 2009)
- Support for Register ABI (register parameter passing) was implemented
  - RegABI build targets (for both Release and Debug configurations) added in all library build projects. The binary libraries compiled with RegABI configuration get the “_regabi” postfix.
  - Former build targets which use the Standard ABI are still maintained in the library build projects for backward compatibility. The targets were renamed to “StdABI”.
  - All example and stationery applications were reconfigured to use Register ABI and RegABI MQX libraries.
- M52277EVB BSP added
- M52233DEMO BSP added
- USB EHCI Host Support implemented
  - USB Host functionality enabled on MCF52277 and MCF54455.
- USB Device Stack reworked and enhanced
  - The USB Device Stack code has been partially rewritten to be consistent with the similar bare-metal stack available for Freescale HCS08 platform (released separately).
  - CDC class implementation examples added (virtual serial line and virtual network interface card).
  - PHDC medical class implementation examples added.
- SPI driver reworked to support all kinds of ColdFire SPI modules (SPI, QSPI and DSPI)
  - Former QSPI driver was removed from the release.
- IPC inter-processor communication files made available in the release.
  - IPC Example applications provided.
  - SCI-based IPC tested only.
- SPI-based SD Card Driver added
  - The driver was tested with Memory storey for Tower Kit only (MCF51CN and MCF52259 devices). Not tested with M52277EVB board.
- Bug fixed: Wrong SCI baud rate divisors calculation fixed.
- Bug fixed: Shell “dir” command file attribute filter is now applied correctly.
- Bug fixed: MFS read and write calls correctly return negative value when physical device returns access error. Please see also another known issue described in section 6.9 “Supporting “Hot Device Uninstall” in MQX I/O Subsystem”.

Freescale MQX Release Notes
## 3 Release Content

This section gives an overview about the release content.

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Location</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compiled MQX Libraries</td>
<td>&lt;install_dir&gt;/lib/...</td>
<td></td>
</tr>
<tr>
<td>MQX PSP and BSP</td>
<td>.../lib/&lt;board&gt;/mqx</td>
<td>updated</td>
</tr>
<tr>
<td>MQX RTCS (TCP/IP network stack)</td>
<td>.../lib/&lt;board&gt;/rtcs</td>
<td>updated</td>
</tr>
<tr>
<td>MQX MFS (File System)</td>
<td>.../lib/&lt;board&gt;/mfs</td>
<td>updated</td>
</tr>
<tr>
<td>MQX USB Host Drivers</td>
<td>.../lib/&lt;board&gt;/usb/host</td>
<td>updated</td>
</tr>
<tr>
<td>MQX USB Device Drivers</td>
<td>.../lib/&lt;board&gt;/usb/device</td>
<td>updated</td>
</tr>
<tr>
<td>MQX Shell Library</td>
<td>.../lib/&lt;board&gt;/shell</td>
<td>updated</td>
</tr>
</tbody>
</table>

| Configuration Files and Mass-Build Projects | <install_dir>/config/... |          |
| Configuration and mass-build project for M52259EVB board | .../config/m52259evb | updated  |
| Configuration and mass-build project for M52259DEMO board | .../config/m52259demo | updated  |
| Configuration and mass-build project for MCF52259 Tower Kit | .../config/twrmcf52259 | updated  |
| Configuration and mass-build project for M52235EVB board | .../config/m52235evb | updated  |
| Configuration and mass-build project for M52223EVB board | .../config/m52223evb | updated  |
| Configuration and mass-build project for MCF521CN128 Tower Kit | .../config/twrmcf51cn | updated  |
| Configuration and mass-build project for M52227EVB board | .../config/m52227evb | new      |
| Configuration and mass-build project for M54455EVB board | .../config/m54455evb | new      |

| MQX PSP Source Code and Examples | <install_dir>/mqx/... |
| MQX PSP source code for ColdFire | .../mqx/source/psp/coldfire | updated  |
| MQX PSP build projects | .../mqx/build/codewarrior/psp_* | updated  |
| MQX example applications | .../mqx/examples/... | updated  |

| MQX BSP Source Code | <install_dir>/mqx/... |
| BSP source code for M52259EVB board | .../mqx/source/bsp/m52259evb | updated  |
| BSP source code for M52259DEMO board | .../mqx/source/bsp/m52259demo | updated  |
| BSP source code for MCF52259 Tower Kit | .../mqx/source/bsp/twrmcf52259 | updated  |
| BSP source code for M52235EVB board | .../mqx/source/bsp/m52235evb | updated  |
| BSP source code for M52223EVB board | .../mqx/source/bsp/m52223evb | updated  |
| BSP source code for MCF51CN128 Tower Kit | .../mqx/source/bsp/twrmcf51cn | updated  |
| BSP source code for M52277EVB board | .../mqx/source/bsp/m52277evb | new      |
| BSP source code for M54455EVB board | .../mqx/source/bsp/m54455evb | new      |
| BSP build projects | .../mqx/build/codewarrior/bsp_* | updated  |

| RTCS Source Code and Examples | <install_dir>/rtcs/... |
| RTCS source code | .../rtcs/source | updated  |
| RTCS build projects | .../rtcs/build/codewarrior | updated  |
| RTCS example applications | .../rtcs/examples | updated  |

| MFS Source Code and Examples | <install_dir>/mfs/... |
| MFS source code | .../mfs/source | updated  |
| MFS build projects | .../mfs/build/codewarrior | updated  |
| MFS example applications | .../mfs/examples | updated  |

<p>| USB Host Drivers Source Code and Examples | &lt;install_dir&gt;/usb/host/... |
| USB Host source code and class drivers | .../usb/host/source | updated  |
| HUB Class Driver | .../usb/host/source/classes/hub | updated  |
| Human Interface Device (HID) Class Driver | .../usb/host/source/classes/hid | updated  |
| Mass Storage (MASS) Class Driver | .../usb/host/source/classes/mass | updated  |
| Printer Class Driver | .../usb/host/source/classes/printer | updated  |</p>
<table>
<thead>
<tr>
<th>Component</th>
<th>Directory Path</th>
<th>Version Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDC Class Driver</td>
<td>.../usb/host/source/cdc</td>
<td>updated</td>
</tr>
<tr>
<td>USB Host build projects</td>
<td>.../usb/host/build/codewarrior</td>
<td>updated</td>
</tr>
<tr>
<td>USB Host example applications (HID, MASS)</td>
<td>.../usb/host/examples</td>
<td>updated</td>
</tr>
<tr>
<td>USB Device Drivers Source Code and Examples</td>
<td><code>&lt;install_dir&gt;/usb/device/...</code></td>
<td></td>
</tr>
<tr>
<td>USB Device source code</td>
<td>.../usb/device/source</td>
<td>reworked</td>
</tr>
<tr>
<td>USB Device build projects</td>
<td>.../usb/device/build/codewarrior</td>
<td>updated</td>
</tr>
<tr>
<td>USB Device example applications (HID, MASS)</td>
<td>.../usb/device/examples</td>
<td>reworked</td>
</tr>
<tr>
<td>Shell Library Source Code</td>
<td><code>&lt;install_dir&gt;/shell/...</code></td>
<td></td>
</tr>
<tr>
<td>Shell source code</td>
<td>.../shell/source</td>
<td>updated</td>
</tr>
<tr>
<td>Shell build projects</td>
<td>.../shell/build/codewarrior</td>
<td>updated</td>
</tr>
<tr>
<td>CodeWarrior Support</td>
<td><code>&lt;CodeWarrior_dir&gt;/...</code></td>
<td></td>
</tr>
<tr>
<td>MQX New project stationery</td>
<td><code>&lt;cw_dir&gt;/stationery/Freescale MQX 3.4</code></td>
<td>new</td>
</tr>
<tr>
<td>MQX Task-aware Debugger plug-in (TAD)</td>
<td><code>&lt;cw_dir&gt;/bin/plugins/debugger/rtos/</code></td>
<td>from 3.3</td>
</tr>
<tr>
<td>TAD and Stationery files for manual copy-installation into another CodeWarrior</td>
<td>.../tools/codewarrior_extensions/...</td>
<td>updated</td>
</tr>
<tr>
<td>PC Host Tools</td>
<td><code>&lt;install_dir&gt;/tools</code></td>
<td></td>
</tr>
<tr>
<td>TFS Make Utility</td>
<td>.../tools/mkfs.exe</td>
<td>from 3.0</td>
</tr>
<tr>
<td>Check for Latest Version tool</td>
<td>.../tools/webchk.exe</td>
<td>from 3.0</td>
</tr>
<tr>
<td>AWK interpreter (GNU General Public License)</td>
<td>.../tools/gawk.exe</td>
<td>from 3.1</td>
</tr>
<tr>
<td>SNMP code generation scripts</td>
<td>.../tools/snmp/*.awk</td>
<td>from 3.1</td>
</tr>
<tr>
<td>TAD string and configuration files</td>
<td>.../tools/tad</td>
<td>updated</td>
</tr>
<tr>
<td>Demo Applications</td>
<td><code>&lt;install_dir&gt;/demo</code></td>
<td></td>
</tr>
<tr>
<td>Various demo applications described in detail by step-by-step 'Lab' documents.</td>
<td>.../demo/...</td>
<td>from 3.3</td>
</tr>
<tr>
<td>Documentation</td>
<td><code>&lt;install_dir&gt;/doc</code></td>
<td></td>
</tr>
<tr>
<td>User Guides and Reference Manuals for MQX RTOS, RTCS, MFS, IO Drivers, USB etc.</td>
<td>.../doc</td>
<td>updated</td>
</tr>
</tbody>
</table>
The following picture shows the Freescale MQX™ RTOS directories installed to the user host computer:

- Freescale MQX[x.y]
  - config
    - common
    - twmd51.cn
    - n52223evb
    - n52235evb
    - n52259demo
    - n52259evb
  - doc
    - Documentation
  - lib
    - twmd51.cn.o
    - n52223evb.o
    - n52235evb.o
    - n52259demo.o
    - n52259evb.o
    - MQX libraries compiled in CodeWarrior for particular board (this is where MQX applications get the MQX from)
  - mfs
    - build
    - example
    - source
  - max
    - build
    - examples
    - source
  - rtems
    - build
    - examples
    - source
  - shell
    - build
    - source
  - tools
    - PC Host Tools
  - usb
    - device
      - build
      - examples
      - source
    - host
      - build
      - examples
      - source

Freescale MQX Release Notes
4 MQX Release Overview

This is MQX RTOS release done by Freescale Semiconductor. It is targeting Freescale ColdFire microcontrollers of V1 to V4 families.

It is based on the version 2.50 as it was released by the company ARC International. Freescale MQX™ RTOS contains also other core system components like RTCS network stack, MFS file system and USB drivers which used to be released and sold separately by ARC.

In addition to changed directory structure, the Freescale release also brings a new way of how user configures the kernel components before compiling. See more details in sections below.

The following table summarizes availability of various MQX components in the latest Freescale MQX release:

<table>
<thead>
<tr>
<th>System Component</th>
<th>TWRMCF51CN</th>
<th>M52223EVB</th>
<th>M52233DEMO</th>
<th>M52235EVB</th>
<th>M52259EVB</th>
<th>M52259DEMO</th>
<th>TWRMCF52259</th>
<th>M52277EVB</th>
<th>M54455EVB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQX PSP+BSP</td>
<td>●●●●●●●●● ●●●●●●●●●●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFS FileSystem</td>
<td>●●●●●●●● ●●●●●●●●●●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTCS TCP/IP Stack</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>USB Host</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB Device</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Shell Library</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

1) No Ethernet module available, PPP development in progress
● New in this release

4.1 MQX RTOS PSP

This version of Freescale MQX™ RTOS contains ColdFire Platform Support Package only. Support for PowerPC platform and other Freescale processor platforms is planned for future releases. Contact Embedded Access Inc. (www.embedded-access.com) for commercial-grade support and support with other Freescale platforms.

The major PSP modifications as compared to the ARC release:

- The compile-time configuration of the MQX PSP (as well as all the other MQX components) is now accomplished by editing the ../config/<board>/user_config.h file. In the previous versions, the user configuration macros were passed to the build process on a command line by the makefile or CodeWarrior prefix file.
- The PSP is configured and built for specific board. Originally, the PSP component was built once for the particular device. While the PSP code still remains board-independent, this feature enables finer kernel tuning for a specific board.
- MQX now supports typed memory, which allows additional information to be displayed in the Task Aware Debugger Plugin.
4.2 MQX RTOS BSPs

This release includes Board Support Package for boards mentioned above in Section 1. In addition to the board initialization code, BSP projects contain I/O drivers suitable for each board. The following section gives an overview about drivers supported in the MQX BSPs.

4.2.1 I/O Drivers Supported

The following table gives an overview about I/O drivers available in the last MQX release. Like the whole I/O subsystem, the drivers are optional part of the MQX RTOS and their installation can be enabled or disabled in the BSP startup code. To save Code memory and RAM, most of the drivers are disabled in the /config/<board>/user_config.h file by default. Only the drivers required by demonstration applications (in the /demo folder) are enabled by default.

Note: When BSPCFG_driver-enabling macros are not set in the /config/<board>/user_config.h file, the default setting is taken from BSP-specific header file located in /mqx/source/bsp/<board>/<board>.h.

It is an application programmer decision to enable automatic installation of the driver in the BSP startup code (by enabling appropriate BSPCFG_ENABLE_x macro in user_config.h) or manually in the application code.

<table>
<thead>
<tr>
<th>I/O Device</th>
<th>Peripheral Module</th>
<th>Configuration Macro (user_config.h)</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttya:</td>
<td>UART0 (polled)</td>
<td>BSPCFG_ENABLE_TTYA</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ttyb:</td>
<td>UART1 (polled)</td>
<td>BSPCFG_ENABLE_TTYB</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ttyc:</td>
<td>UART2 (polled)</td>
<td>BSPCFG_ENABLE_TTYC</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ittya:</td>
<td>UART0 (irq.driven)</td>
<td>BSPCFG_ENABLE_ITTYA</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ittyb:</td>
<td>UART1 (irq.driven)</td>
<td>BSPCFG_ENABLE_ITTYB</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ittyc:</td>
<td>UART2 (irq.driven)</td>
<td>BSPCFG_ENABLE_ITTYC</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>i2c0:</td>
<td>I2C0 (polled)</td>
<td>BSPCFG_ENABLE_I2C0</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>i2c1:</td>
<td>I2C1 (polled)</td>
<td>BSPCFG_ENABLE_I2C1</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ii2c0:</td>
<td>I2C0 (irq.driven)</td>
<td>BSPCFG_ENABLE_I2C0</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ii2c1:</td>
<td>I2C1 (irq.driven)</td>
<td>BSPCFG_ENABLE_I2C1</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>spi0:</td>
<td>SPI0 (polled)</td>
<td>BSPCFG_ENABLE_SPI0</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>spi1:</td>
<td>SPI1 (polled)</td>
<td>BSPCFG_ENABLE_SPI1</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ispi0:</td>
<td>SPI0 (irq.driven)</td>
<td>BSPCFG_ENABLE_ISPI0</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>ispi1:</td>
<td>SPI1 (irq.driven)</td>
<td>BSPCFG_ENABLE_ISPI1</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>gpio:</td>
<td>GPIO</td>
<td>BSPCFG_ENABLE_GPIODEV</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>adc:</td>
<td>ADC</td>
<td>BSPCFG_ENABLE_ADC</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>flashx:</td>
<td>CFM</td>
<td>BSPCFG_ENABLE_FLASHX</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>flash0:</td>
<td>External Flash</td>
<td>BSPCFG_ENABLE_EXT_FLASH0</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>flash1:</td>
<td>External Flash</td>
<td>BSPCFG_ENABLE_EXT_FLASH1</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>-</td>
<td>RTC</td>
<td>BSPCFG_ENABLE_RTCDEV</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>pccarda:</td>
<td>FlexBus + CPLD</td>
<td>BSPCFG_ENABLE_PCFLASH</td>
<td>● ● ● ● ● ● ● ●</td>
</tr>
</tbody>
</table>
TFS – Trivial Filesystem
Tiny filesystem which can be used as a simple read-only file repository instead of fully featured MFS. TFS is not installed in BSP startup code. It is a task for the application to initialize the TFS and pass a pointer to the filesystem image data. 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 use of TFS.

I2C I/O Driver
– This driver supports a polled I2C interface in master mode. If enabled in user configuration the I2C device drivers are installed during the BSP startup code as “i2c0:” and “i2c1:”. Example application is provided in the MQX source tree.

SPI I/O Driver
This driver supports various SPI interfaces available in ColdFire V1 to V4 (SPI, QSPI, DSPI). The driver supports in master mode of operation only. If enabled in user configuration the SPI device drivers are installed during the BSP startup code as “spi0:” and “spi1:” (polled-mode) and “ispi0:” and “ispi1:” (interrupt-mode). Example application is provided in the MQX source tree.

FlexCAN Driver
This driver provides a C language API to the FlexCAN peripheral module. It is not a true I/O driver as defined by MQX terminology. Example application is provided in the MQX source tree.

RTC Driver
This driver provides a C language API to the Real Time Clock peripheral module and functions helping to synchronize clock time between RTC and MQX system. If enabled in user configuration, the RTC module is initialized and MQX time is renewed automatically during BSP startup. The Real Time Counter module of MCF51xxx family is supported by the RTC driver as well. Some missing features of this module (comparing it to Real Time Clock module of MCF52xx) are emulated in the software driver.

Serial I/O Driver
The standard 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 “ittyta:”, “ittytb:” and “ittytc:” (interrupt mode) automatically during BSP startup.

GPIO I/O Driver
The I/O driver providing a uniform interface to all GPIO pins available. If enabled in user configuration, the GPIO driver is installed as “gpio:” automatically by the BSP startup.

ADC Driver
The I/O driver providing a uniform interface to ADC channels. ADC uses internal PIT timer for periodic sampling. If enabled in user configuration, the ADC driver is installed as “adc:” device automatically by the BSP startup.

This release contains an initial version of the driver ported to two different ADC modules available in MCF51xx and MCF52xx families of microcontrollers. The code (including the API) is subject to improvement in future releases. Current version limitations:

- Channel resolution setting & scaling is not yet implemented.
- Generic features implemented only, not able to use HW specific features of different ADC modules. HW-specific IOCTL commands to be implemented.
Flash I/O Driver
The I/O driver providing a standard interface to 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 (“flash0”, “flash1” etc. device names are used for FlashX device installed for external Flash memory).

For devices with internal Flash memory, the FlashX driver depends on several parameters passed in form of global symbols from application or from Linker Command File. See the example application provided in the MQX source tree.

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

PCCard I/O Driver
The I/O driver providing a low-level access to the CPLD-driven PCCard functionality. If enabled in the user configuration, the PCCard device driver is installed as “pccarda:” automatically during the BSP startup.

This driver is only used in M52259EVB BSP only.

PCFlash I/O Driver
The Compact Flash Card I/O driver which uses 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
The I/O driver implementing a subset of SD protocol v2.0 (SDHC). The current version of the driver uses MQX SPI I/O driver to communicate with SD Card device. The driver should be installed at the application level, passing it a SPI driver handle. The MFS file system can be installed on top of this device.

4.2.2 Serial Console
One of the serial communication devices installed by MQX BSP may be used as the default serial console. By default, the console is used with the serial device for which standard RS232 9-pin connector is available on board. The default selection is made in BSP-specific header file (see how BSP_DEFAULT_IO_CHANNEL macro is defined in for example in the mqx\source\bsp\m52259evb\m52259evb.h file).
4.2.3 MQX PSP and BSP Directory Structure

The RTOS files are located in the `mqx` subdirectory of the Freescale MQX™ RTOS installation. The directory structure is briefly described in the picture below.

- `mqx` directory
  - `build` directory
    - `codewarrior` directory (CodeWarrior project files to build PSP and BSP libraries)
  - `examples` directory
  - `source` directory
    - `bsp` directory
      - `bwrncf51.cn` directory
      - `n52299demo` directory (BSP source code specific to each board (build to BSP library))
      - `n52299evb` directory
    - `ods` directory
    - `edserial` directory
    - `event` directory
    - `fio` directory
    - `include` directory
  - `io` directory (I/O Drivers source code (build to BSP library))
    - `kernel` directory
    - `log` directory
      - `log` directory
      - `lwevent` directory
      - `lilog` directory
      - `lwmom` directory
      - `lwmsg` directory
      - `lwtimer` directory
      - `message` directory
      - `mutex` directory
      - `name` directory
      - `part` directory
      - `profile` directory
    - `psp` directory (PSP sources - the platform-specific part of PSP library)
      - `coldfire` directory
      - `queue` directory
      - `sem` directory
      - `string` directory
      - `timer` directory
      - `watchdog` directory
  - `{ MQX generic sources (build to PSP library) }
  - `{ Rest of the MQX generic sources (build to PSP library) }

Freescale MQX Release Notes

Freescale Semiconductor
4.3 MQX MFS

This release of Freescale MQX™ RTOS includes the MFS File System libraries. The MFS is based on the version 2.20 released by ARC.

The major MFS modifications as compared to the ARC release:

- The compile-time configuration of the MFS is now accomplished by editing the ...
  /config/<board>/user_config.h file.
- MFS was significantly optimized for devices with limited RAM resources. In particular, MFS now performs sector caching, not cluster caching. This generally results in better performance with a lower RAM footprint.
- MFS supports allocation of resources from a specific memory pool.
- Compile time and run-time configurable options were added in order to selectively enable and disable individual MFS features.
- Similarly as the PSP, the MFS is configured and built for specific board. Originally, the MFS component was built once for the particular device or even the whole ColdFire family. While the MFS code still remains board-independent, this feature enables finer feature tuning for a specific board.

4.4 MQX RTCS

This release of Freescale MQX includes the RTCS TCP/IP stack libraries. The RTCS is based and is API-level compatible with version 2.97 released by ARC.

The major RTCS modifications as compared to the ARC release:

- The compile-time configuration of the RTCS is now accomplished by editing the ...
  /config/<board>/user_config.h file.
- Similarly as the PSP, the RTCS is configured and built for specific board. Originally, the RTCS component was built once for the particular device or even the whole ColdFire family. While the RTCS code still remains board-independent, this feature enables finer feature tuning for a specific board.
- The RTCS was significantly optimized for devices with limited RAM resources. Several configurable options were added in order to selectively enable and disable individual TCP/IP features and protocols.
- The new FTP Client and Server have been moved from the example directory to the apps directory.
- IPCFG feature in RTCS allows to monitor Ethernet link status and performs automated IP address binding, either static IP, DHCP or DHCP with Auto IP assignment. Status may be monitored in dedicated background task or by polling calls. New “ipconfig” shell command is available as a user interface to the IPCFG functionality.
- RTCS supports allocation of memory resources from a specific memory pool.
- The HTTP server code was added. This server enables standard to implement web server with a support for CGI-like callbacks or ASP-like page placeholder callbacks.

4.5 MQX USB Host

This release of Freescale MQX includes the USB Host drivers and USB class drivers. The USB code is based on version 1.2.0 released by ARC.
The major USB modifications as compared to the ARC release:

- The compile-time configuration of the USB Host library is now accomplished by editing the
  `.../config/<board>/user_config.h` file.
- The separate Device-specific Driver, Host Development Kit library and USB Class library
  projects were merged into a single library called HDK. The board-specific code was moved
to BSP library.
- Similarly as the PSP, the HDK library is now configured and built for specific board.

4.5.1 USB Class Drivers supported by Host Development Kit

- Human-interface Class (HID) used to access mouse, keyboard and similar devices.
- Mass storage Class used to access USB drives.
- 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. User application only needs to be
  modified to be ready to handle multiple USB devices simultaneously. Keyboard/Mouse
  example application is provided.
- Basic Printer class.

4.6 MQX USB Device

This release of Freescale MQX includes the USB Device drivers and example applications.

Features:

- The compile-time configuration of the USB Device Development Kit (DDK) library is now
  accomplished by editing the `.../config/<board>/user_config.h` file.
- Similarly as the PSP, the DDK library is now configured and built for specific board.
- The USB Device communication stack implements a generic API and does not contain any
  specific class drivers. The class support is implemented at the application level. This concept
  is subject to change in future versions.

4.6.1 Device Class Example Applications

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

4.7 MQX Shell

The shell and command-line handling code is implemented as a separate library called Shell.
Similarly as other components, the shell library is configured and built for each specific board
separately.

4.8 Compile-time Configuration

One of the major changes when comparing the Freescale release with the original ARC release is
the approach to compile-time configuration of kernel and other core components. All compile-time
configuration options were centralized into a single user configuration file located in
This user configuration file is included internally by private configuration files in MQX PSP and BSP.
To share configuration settings between different boards, the user_config.h may include other header files with common settings. The header files may only be located in the same <board> directory or in the common directory:

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

All MQX configuration files are also indirectly used by other core components like RTCS, MFS, etc. “Indirectly” means that the MQX PSP and BSP must be build first, which causes the configuration file being copied into the output (lib) directory. The other components then include the configuration file from the /lib output directory.

**Caution:** Until the PSP or BSP libraries are rebuilt, the configuration change made in the user_config.h file is not used by any other MQX component. On the other hand, after the PSP and BSP libraries are re-compiled with a new configuration, it is important to recompile the other libraries so the compiled code is consistent with the configuration file. See the next section for more details.

### 4.9 Changing the MQX Source Files

The Freescale MQX RTOS is distributed in the source code form. However, unless you are advanced in C programming and MQX kernel knowledge, it is recommended NOT to modify any of the source files other than compile-time configuration files. This recommendation applies to all files under “source” and “build” sub-directories in all MQX, RTCS, MFS, USB and other core components folders.

Only in case of creating custom board support packages or adding additional I/O drivers, there are two directories where the new files and subdirectories may need to be added:

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

### 4.10 Building the MQX Libraries

Pay attention to use correct CodeWarrior Development Studio when opening any MQX projects. There are two different CodeWarrior Studios for MCF51xx and MCF52xx ColdFire families of microcontrollers. See also section 1.1.1 above for more details.

#### 4.10.1 Build Process

After any change to the compile-time user configuration file or MQX kernel source files, the MQX libraries need to be re-built. The build process is similar with all core components:

- The output directory is `<install_dir>/lib/<board>.cw/<component>`
- For example the MQX PSP and BSP libraries for M52259EVB board are built in the `.../lib/m52259evb.cw/mqx` directory.
- All public header files needed by application to make use of the library are copied from internal include folders to the same output directory as the library itself.
- During PSP or BSP build process, also the `user_config.h` file and other header files from the `config/<board>` and `config/common` directories are copied into the `.../lib/<board>.cw` output directory.
- Other components like RTCS, MFS, etc. use the copied configuration files only.
- Similarly as the PSP and BSP, the other libraries build into the output directories inside 
  .../lib/<board>.cw

To summarize the points above, there are simple rules to obey when re-building the MQX libraries.

- After any change to the .../config/common/user_config.h file, all MQX libraries should 
  be re-built.
- The PSP and BSP libraries must be build first, before the RTCS, MFS and other libraries.
- No changes should be made to header files in the output build directory (/lib). The files get 
  overwritten any time the libraries are built.

To build all MQX libraries at once, the special mass-build project may be used. The project is 
located together with the user_config.h file in directory:

<install_dir>/config/<board>/build_libs.mcp

4.10.2 Debug and Release Build Targets

Each CodeWarrior 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. All output 
  libraries (or executables) have _d postfix in the file name (e.g. rtcs_d.lib).
- **Release** – the compiler optimizations are set to maximum. The compiled code is very hard 
to debug and should be used for final applications only. There is no postfix in the output file 
  name (e.g. rtcs.lib).

Similarly, two different types of build targets exist for different application binary interface (ABI) 
settings:

- **StdABI** – the compiler is set to generate functions calls with standard parameter (on-stack) 
  passing interface. This was the only available option in the MQX versions prior 3.4.
- **RegABI** – starting in MQX release 3.4, the register parameter passing is enabled and is set 
as default in all example and stationery projects. All output libraries have the “_regabi” 
  postfix in the file name (e.g. rtcs_regabi.a). Register parameter passing option gives 
generally smaller and faster code.

Combining the complier settings described above, the following build targets exist in MQX library 
build projects:

- **Debug RegABI** – register parameter passing ABI, easy to debug code
- **Release RegABI** – register parameter passing ABI, optimization set to maximum
- **Debug StdABI** – on-stack parameter passing ABI, easy to debug code
- **Release StdABI** – on-stack parameter passing ABI, optimizations set to maximum
- **Build all** – this target contains all other targets for convenient mass-build.
Unlike the library build projects, the application and example projects are set to use register parameter passing ABI. Each project contains **Debug** and **Release** build targets and other targets specific for board memory configuration.

### Devices with internal Flash memory (e.g. MCF522xx):

- **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 internal SRAM memory.

- **Int. Flash Debug** – same as above, only the Debug-compiled libraries are used. This target is suitable for debugging before deployment. On boards without external memory, this is the only target suitable for debugging larger applications.

**Note for CodeWarrior Development Studio for ColdFire:** Application executables built with Flash build targets need to be written to the Flash memory using the CodeWarrior FlashProgrammer tool. Don’t forget to load proper FlashProgrammer XML Settings file before using the FlashProgrammer tool. The Settings files for all supported processors are available in the CodeWarrior installation directory or in the MQX installation folder in `tools/flash_programmer/config`.

### Devices with internal SRAM memory (MCF5225x etc.):

- **Int. SRAM Debug** – solely for debugging purposes with both code and variables allocated in the internal SRAM memory. This target is unusable with larger applications. Application executable is loaded to RAM automatically by the debugger.

  **Note:** in this target, the RAM area of is split to Code and Data parts in approximate ratio of 8:1. It may happen that this split does not suit well all applications and may need to be changed in linker command file (`intram.lcf`). Remember that this target is only suitable for tiny MQX-only applications. This target is not available in applications where the application can not fit into RAM memory (RTCS, MFS or USB applications).

### Boards with external MRAM memory (M52259EVB etc.):

- **Ext. MRAM Debug** – solely for debugging purposes with code located in external MRAM memory (available e.g. on the M52259EVB). Variables are located in internal SRAM. Part of the external MRAM memory may also be used for additional RAM memory pools. Application executable is loaded to MRAM automatically by the debugger.
Boards with external RAM memory (M54455EVB etc.):

- **Ext. Ram Debug** – solely for debugging purposes with code located in external RAM memory (available as SDRAM e.g. on the M54455EVB). Both code and variables are located in this external memory. Application executable is loaded to RAM automatically by the debugger.
The following table gives an overview about what build targets are available for which BSP/board.

<table>
<thead>
<tr>
<th>Target Name</th>
<th>Availability</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int. Flash Release</td>
<td>● ● ● ● ● ●</td>
<td>For M51CN128 Tower Kit, targets are available in two instances: one for P&amp;E BDM cable, one for on-board OSBDM interface</td>
</tr>
<tr>
<td>Int. Flash Debug</td>
<td>● ● ● ● ● ●</td>
<td>Only a few of M52259DEMO example applications use this target.</td>
</tr>
<tr>
<td>Int. SRAM Debug</td>
<td>● ● ● ● ● ●</td>
<td>External memory card with MRAM is needed for MCF52259 Tower Kit.</td>
</tr>
<tr>
<td>Ext. MRAM Debug</td>
<td>● ● ● ● ● ●</td>
<td>See board-specific section for more details.</td>
</tr>
</tbody>
</table>

**4.11 Example Applications**

There are example „Lab“ applications in the directory:

```
<install_dir>/demo
```

The examples are accompanied with Lab guide documents – describing step-by-step how to run them on the target board. The examples were 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 following tables summarize all demo and example applications provided in this release.

**MQX Example Applications**

```
>mxx/examples/...
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adc</td>
<td>● ● ● ● ● ●</td>
<td>Shows usage of the ADC driver, sampling analog values from two ADC channels.</td>
</tr>
<tr>
<td>can/flexcan</td>
<td>● ● ● ● ● ●</td>
<td>Shows usage of FlexCAN API functions to transmit and receive standard or remote frames.</td>
</tr>
<tr>
<td>demo</td>
<td>● ● ● ● ● ●</td>
<td>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.</td>
</tr>
<tr>
<td>event</td>
<td>● ● ● ● ● ●</td>
<td>Simple demonstration of MQX events.</td>
</tr>
<tr>
<td>gpio</td>
<td>● ● ● ● ● ●</td>
<td>Shows usage of GPIO driver to control on-board LEDs and switches.</td>
</tr>
<tr>
<td>hello</td>
<td>● ● ● ● ● ●</td>
<td>A trivial Hello World application using a single task.</td>
</tr>
<tr>
<td>hello2</td>
<td>● ● ● ● ● ●</td>
<td>A trivial Hello World application spread across two tasks.</td>
</tr>
<tr>
<td>i2c</td>
<td>Shows how to read/write data from/to external SPI EEPROM. Additional HW setup is needed.</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>io</td>
<td>Shows using an alternate UART port as a console output.</td>
<td></td>
</tr>
<tr>
<td>isr</td>
<td>Shows how to install interrupt service routine and how to chain it with the previous handler.</td>
<td></td>
</tr>
<tr>
<td>klog</td>
<td>Shows kernel events being logged and later the log entries dumped on console.</td>
<td></td>
</tr>
<tr>
<td>log</td>
<td>Shows application-specific logging feature.</td>
<td></td>
</tr>
<tr>
<td>lwdemo</td>
<td>Same as the &quot;demo&quot; application, but implemented using lightweight components only.</td>
<td></td>
</tr>
<tr>
<td>lwevent</td>
<td>Simple demonstration of MQX lightweight events.</td>
<td></td>
</tr>
<tr>
<td>lwlog</td>
<td>Simple demonstration of MQX lightweight log feature.</td>
<td></td>
</tr>
<tr>
<td>lwmsgq</td>
<td>Simple demonstration of MQX lightweight inter-process messaging.</td>
<td></td>
</tr>
<tr>
<td>lwsem</td>
<td>Simple demonstration of MQX task synchronization using the lightweight semaphore object.</td>
<td></td>
</tr>
<tr>
<td>msg</td>
<td>Simple demonstration of MQX inter-process message passing.</td>
<td></td>
</tr>
<tr>
<td>mutex</td>
<td>Simple demonstration of MQX task synchronization using the mutex object.</td>
<td></td>
</tr>
<tr>
<td>nill</td>
<td>Yet simpler than Hello World. A void application which may be used for copy/paste to start custom application.</td>
<td></td>
</tr>
<tr>
<td>rtc</td>
<td>Shows the Real Time Clock module API. Demonstrates how to synchronize RTC and MQX time and how to use RTC alarm interrupts.</td>
<td></td>
</tr>
<tr>
<td>sem</td>
<td>Simple demonstration of MQX task synchronization using the semaphore object.</td>
<td></td>
</tr>
<tr>
<td>spi</td>
<td>Shows how to read/write data from/to external SPI EEPROM. Additional HW setup is needed.</td>
<td></td>
</tr>
<tr>
<td>taskat</td>
<td>Shows how task can be created within statically allocated memory buffer (avoid heap allocation for task stack and context).</td>
<td></td>
</tr>
<tr>
<td>taskq</td>
<td>Shows custom task queue and how the queue can be suspended and resumed.</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>Shows the self-testing feature of each MQX component.</td>
<td></td>
</tr>
<tr>
<td>tfs</td>
<td>Shows the usage of ROM-based Trivial File System in an MQX application.</td>
<td></td>
</tr>
<tr>
<td>timer</td>
<td>Simple demonstration of MQX timer component.</td>
<td></td>
</tr>
<tr>
<td>watchdog</td>
<td>Simple demonstration of MQX task timeout detection using the watchdog component.</td>
<td></td>
</tr>
</tbody>
</table>
## RTCS Example Applications

### Availability

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>httpsrv</td>
<td></td>
<td>Simple web server with CGI scripts and few web pages stored in internal flash.</td>
</tr>
<tr>
<td>shell</td>
<td></td>
<td>Shell command line providing commands for network management.</td>
</tr>
<tr>
<td>snmp</td>
<td></td>
<td>SNMP protocol example providing microprocessor state information.</td>
</tr>
</tbody>
</table>

## MFS Example Applications

### Availability

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mfs_ftp</td>
<td></td>
<td>RTCS FTP demo accessing MFS filesystem mounted on the USB mass storage. For FTP example without USB functionality, refer to RTCS Shell demo.</td>
</tr>
<tr>
<td>mfs_usb</td>
<td></td>
<td>Console shell-based example showing how to access MFS filesystem mounted on the USB mass storage.</td>
</tr>
<tr>
<td>mfs_cfcard_usb</td>
<td></td>
<td>Console shell-based example showing the MFS filesystem used with both USB and CFCard storage. Files can be copied between the two devices.</td>
</tr>
<tr>
<td>ramdisk</td>
<td></td>
<td>Shows use of MFS accessing the external RAM (or MRAM).</td>
</tr>
<tr>
<td>sdcard</td>
<td></td>
<td>Shows use of MFS accessing the SPI-connect SD Card.</td>
</tr>
</tbody>
</table>

## USB Host Example Applications

### Availability

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hid/keyboard</td>
<td></td>
<td>This application echoes keys pressed on the USB keyboard onto serial console.</td>
</tr>
<tr>
<td>hid/mouse</td>
<td></td>
<td>Displays USB mouse events on serial console.</td>
</tr>
<tr>
<td>hid/keyboard+mouse</td>
<td></td>
<td>Keyboard and mouse demos combined in a single application.</td>
</tr>
<tr>
<td>msd/mad_commands</td>
<td></td>
<td>Executes the standard &quot;mass storage device&quot; commands to the USB disk and shows the response on the serial console (see mfs demos for USB filesystem access).</td>
</tr>
</tbody>
</table>

## USB Device Example Applications

### Availability

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
</table>
### Lab Tutorial Demos

demo/...

<table>
<thead>
<tr>
<th>Name</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hvac</td>
<td>● ● ●</td>
<td>Simple implementation of console-based HVAC, with optional USB logging and FTP access.</td>
</tr>
<tr>
<td>hvac_error</td>
<td>● ● ●</td>
<td>Intentional error injected to the HVAC demo code to demonstrate power of TAD plug-in.</td>
</tr>
<tr>
<td>web_hvac</td>
<td>● ● ●</td>
<td>The HVAC demo with HTTP server implementing the GUI. Ajax-based pages demonstrate an advanced use of HTTP server.</td>
</tr>
<tr>
<td>telnet_to_serial</td>
<td>● ● ● ● ● ●</td>
<td>Simple character passing between UART console and telnet session. Shows custom &quot;lightweight&quot; telnet server.</td>
</tr>
<tr>
<td>security_email</td>
<td>● ● ● ● ● ●</td>
<td>Small-footprint application originally developed for MCF51CN. Example of simple SMTP client, able to send email message upon activity detected on GPIO input pins.</td>
</tr>
<tr>
<td>security_telnet</td>
<td>● ⋄</td>
<td>Telnet-based security monitor, displaying status of GPIO and ADC values.</td>
</tr>
<tr>
<td>security_webserver</td>
<td>● ⋄</td>
<td>WEB server-based security monitor, displaying status of GPIO and ADC values.</td>
</tr>
</tbody>
</table>

- New in this release
4.12 CodeWarrior Task Aware Debugging Plug-in

Freescale MQX™ RTOS brings a new version of Task Aware Debugging Plug-in (so called TAD) for CodeWarrior Development Studio. Comparing it to the previous ARC release, the TAD was significantly re-written in order to achieve better stability and portability across different CodeWarrior versions.

Note: The TAD plug-in is not available for CodeWarrior Development Studio for Microcontrollers of ColdFire V1 family.

4.12.1 Installing CodeWarrior TAD

TAD plug-in DLL is installed into the selected CodeWarrior tool automatically during Freescale MQX™ RTOS setup process. Perform the following steps to install TAD manually (for example to a newly installed CodeWarrior studio):

1. Locate the tools\tad directory in Freescale MQX™ RTOS installation folder (by default C:\Program Files\Freescale\Freescale MQX x.y\tools\tad)
2. Locate the plugins directory in the CodeWarrior installation folder (by default C:\Program Files\Freescale\CodeWarrior for ColdFire V7.1)
3. Copy the CFrtos_MQX.dll file from the MQX tools\tad folder to <CodeWarrior>\bin\Plugins\Debugger\rtos folder.
4. Re-start the CodeWarrior Studio.
5. In the CodeWarrior environment, you should be now able to enable MQX TAD by selecting “MQX” as “Target OS” in the “CF Debugger Settings” panel of project settings. All example applications coming with Freescale MQX™ RTOS are already configured so.
4.12.2 CodeWarrior TAD Features

The TAD plug-in enables more advanced and user-friendly debugging of MQX-based applications.

Using the MQX or RTCS menu in CodeWarrior IDE, several TAD “screens” may be opened during the debugging session. The most helpful and frequently used screens are shown in the picture below:

- **Task Summary** – overview about all tasks created in the MQX application.
- **Stack Usage Summary** – displays information about interrupt and task stacks. Typically, stack overflow is a root cause of vast majority of problems in MQX user applications.
- **Memory Block Summary** – displays address, size and type information about each memory block allocated in the default memory pool by the MQX system or applications. Additional memory pools (if used) may be displayed using “Memory Pools” screen.
TAD also provides native debugger support for multi-tasking MQX environment. When an application is stopped at breakpoint or by pressing the (red-square) “Break” button, the name of active task is displayed in the drop-down list in the debugger window. You also have a chance to use the drop-down list to see current execution point of any other task.

Note: This CodeWarrior feature needs to be enabled in general Debugger settings first. Go to the Edit / Preferences menu in the CodeWarrior main window and select “Debugger / Window Settings” panel. In the “Debugging Windows” section, enable the “Show Processes in Separate Windows” option and disable the “Show Threads in Separate Windows” option.
4.12.3 Backward Compatibility in TAD

Each release of Freescale MQX™ RTOS brings an updated version of the TAD plugin and installs it into the CodeWarrior Development Studio selected by user. In the case that older version of MQX RTOS is installed on the same computer the older TAD plugin gets overwritten by the latest version.

Debugging different MQX versions on the same host

In many cases you may want to develop and debug different applications in different versions of the MQX. For example when a development starts and continues with MQX 3.1 while other applications are developed with MQX 3.3. This should never cause a problem because different MQX versions are completely independent one on each other. The only common resource between different versions is the latest TAD plugin installed in the CodeWarrior Studio.

TAD plugins are designed to be backward compatible, so updating to the latest MQX version should not cause any issue. In case of a problem with compatibility, old TAD plugin versions are available and you can switch to them manually. In project settings (for the selected build target), go to the CF Debugger Settings panel and select the TAD version from the Target OS drop down list.
5 What is Missing?

The MQX development teams in Freescale are working on continuous improvement of the MQX code, bringing new features as well as enabling and validating legacy MQX features with the latest supported devices.

This section provides a list of legacy MQX features which were removed and are not available in the current release of Freescale MQX™ RTOS. Features may either be removed temporarily, until the code is re-tested and validated with the latest supported devices; or permanently when the feature is considered obsolete.

5.1 In MQX PSP

5.1.1 MMU and Virtual Memory Handling
Memory Management Unit and Virtual Memory is not available on devices currently supported by Freescale MQX™ RTOS.

5.2 In USB Support

5.2.1 OTG Support
The USB OTG Drivers are not included in the current release of Freescale MQX™ RTOS.

5.3 Peripheral Drivers

As described in the following sections, not all peripheral modules available on-chip are supported by an MQX I/O driver.

5.3.1 Cryptographic Acceleration Unit
No legacy driver is available in MQX. This module is not suitable for general purpose device driver. Module features will be used internally by future kernel versions.

5.3.2 Random Number Generator
No legacy driver is available in MQX.

A simple generic I/O driver for this module will be used as an example in Application Note document, demonstrating how to create device driver step-by-step. This driver will be part of future MQX releases.

5.3.3 Various Timer Modules
No legacy driver is available in MQX. This driver is a subject of future development.

5.3.4 I2C and SPI Slave Mode Communication
The I2C and SPI drivers now support Master Mode only.

5.3.5 PWM Module
No legacy driver is available in MQX. This driver is a subject of future development.
5.4 PC Host Tools

5.4.1 Legacy Host Tools
PC Host tools known from previous MQX releases (Design tool, EDS client tool, Performance tool) are not included in the current release of Freescale MQX™ RTOS.

5.4.2 TAD Plug-in for MCF51xx
The TAD debugger plug-in has not been ported to external debugger used in CodeWarrior Development Studio for Microcontrollers (supporting the ColdFire V1 family).
6 Known Issues

6.1 Performance of Code Running in MRAM

As described above, the “Ext. MRAM Debug” build-target provides the most convenient way how to download and run code on the M52259EVB board. One limitation of this solution is that the code runtime performance is approximately 8x degraded comparing to Flash-based execution. MRAM is an external memory device connected to the ColdFire core by 8-bit data bus and having 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 inserting one wait-state clock.

This behavior is normal by design.

6.2 Network Communication Performance and Delayed ACKs

Significant differences in performance of host to target TCP communication can be observed, depending on which client operating system is used. These differences are a result of the difference in implementation of the TCP delayed ACK algorithm.

Linux is more aggressive in acknowledging packets, and temporarily disables the delayed ACK algorithm on connection startup. As a result, there is no performance issue observed with Linux-based browsers (Mozilla Firefox tested).

Microsoft Windows based operating systems take a common approach to implementing the delayed ACK algorithm. As a result, web server performance can be significantly worse when it is accessed from Microsoft Windows-based browsers (Internet Explorer and Mozilla Firefox tested). This issue and a workaround are described at http://support.microsoft.com/kb/328890.

This issue is not directly related to the RTCS stack, however limited RAM resources on MCF52259 imply lower buffer and data window sizes, making it more difficult to work around the issue in RTCS software.

6.3 Default Kernel Configuration of Small-RAM Devices

The default kernel configuration for is optimized in order to run the demonstration applications located in the /demo folder. To meet tight RAM constraints, some other MQX or RTCS features are disabled by default. Also some of the I/O drivers not used by demo applications are disabled.

If using an I/O driver is required in your application, you need to enable the it manually in the config/<board>/user_config.h file.

- Initialization of TTYB and ITTYB serial device drivers is disabled in user_config.h (used in mqx/examples/io).
- Initialization of I2C and QSPI device drivers (SPI in case of MCF51xx) is disabled in user_config.h (used in mqx/examples/i2c and mqx/examples/qspi)
- Initialization of real-time clock (RTC) device driver is disabled in user_config.h (used in mqx/examples/rtc).
- Initialization of PCFLASH device driver is disabled in user_config.h (used in mfs/examples/mfs_cfcard_usb)
- The Ethernet device driver is configured to have only 4 receive and 2 transmit buffers (3 receive buffers on MCF5223x and MCF51CNxxx). Size and number of the Ethernet buffers is often further configured from the application.
6.4 USB Host HUB Examples

HUB class support is enabled in HID example applications. The applications run correctly with the USB device attached both directly, or through the hub. However, the example code is still ready to handle a single device only. Combined Mouse+Keyboard demo is able to handle one mouse and one keyboard simultaneously. Using multiple devices of the same kind attached through the hub is not supported by the example applications.

6.5 USB Host HUB Functionality with MCF5222x

The HUB interfacing is problematic with the older MCF52223 device. The issue is addressed by chip errata document.

6.6 BDM Connection Failures with MCF5223x

A problem of BDM communication loss was occasionally observed when debugging M52235EVB applications. This issue is not related to MQX RTOS.

Workaround: This issue may be solved by decreasing the BDM communication speed by factor 2 or higher in the “Remote Debugging” settings panel for PEMICRO_USB connection. Press the “Edit Connection” button and specify the Speed factor value “2”.

6.7 ColdFire V1 Code Footprint in Debug Build Targets

This release of Freescale MQX™ RTOS brings the first version of the RTCS TCP/IP stack for ColdFire V1 devices. Even with the smallest device supported by this MQX release (MCF51CN128), the applications are provided to demonstrate HTTP, DHCP, ICMP and SNTP protocols over the
TCP and UDP. With some of these applications, the size of executable compiled without size optimizations exceeds the Flash memory available on MCF51CN128.

As a workaround, size optimizations were enabled in the “Debug” build targets of the MQX, RTCS and other libraries for the ColdFire V1 processors. All optimizations remain disabled in “Debug” build target of the application projects itself so the application code debugging is not affected.

6.8 OSBDM Firmware Compatibility with CodeWarrior 7.1.2

CodeWarrior patch 7.1.2 brings a new version of the OSBDM Debugger interface with improved performance and stability. This new interface requires the OSBDM firmware (the HCS08JM60 code) to be updated to the latest version (see CodeWarrior 7.1.2 Release Notes for instructions).

This MQX release was tested in the following OSBDM configurations on M52259DEMO and TWR-MCF52259-KIT boards:

- An older OSBDM firmware (factory default for M52259DEMO) with CodeWarrior 7.1.1 using osbdm_cf2_gdi.dll in OSBDM Connection Setting. No issues.

- The latest OSBDM firmware (factory default for TWR-MCF52259-KIT) with CodeWarrior 7.1.2 using osbdm-jm60_gdi_cfv234.dll in OSBDM Connection Setting. No issues.

- Debugging issue was observed when using the CodeWarrior 7.1.2 with an older OSBDM firmware of M52259DEMO board. The workaround is either to upgrade the OSBDM firmware or to switch the Debugger Connection Setting to use osbdm_cf2_gdi.dll.

In project settings, go to “Remote Debugging” panel and press the “Edit Connection” button for “ColdFire v2-v4 JM60 OSBDM” option.
Debugger issue was observed when using the CodeWarrior 7.1.2 with the latest OSBDM firmware when debugging MQX with MQX_ROM_VECTORS option enabled. The workaround is to disable CF Exception catching by the Debugger.

In project settings, go to “CF Exceptions” panel and make sure no exception handling is enabled (uncheck all checkboxes on that panel).

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

With today’s implementation of MQX I/O subsystem, special attention is needed to uninstall a device driver while there is one or more device files open at the application layer. In other words, the
application is responsible for dealing with application tasks that have opened file handles when a device driver is being uninstalled.

A typical demonstration of the problem is USB mass storage handling:
- When USB attach event is detected, an application installs the MFS partition manager and MFS filesystem ‘device’ on top of the USB driver.
- The application runs tasks (e.g. shell) which open and access files provided by the MFS filesystem device.
- When user unplugs the USB mass storage device, the application has only a limited way to detect an opened file exists before uninstalling the MFS filesystem device.
- The file I/O functions begin to report errors when accessing the device after it is physically detached. The application code should be designed in a way that the tasks close all files affected by the detach event before the MFS filesystem driver can be uninstalled.
- When MFS filesystem device is uninstalled while open files are still accessed from other tasks, an unhandled exception may occur.

We recognize that this implementation may add additional application overhead. We are enhancing MQX I/O subsystem so the file operations will safely return error states even after the underlying device driver is uninstalled. This enhancement will simplify the application code error recovery.
7 Board-specific Information Related to MQX

This section contains board-specific information which should be considered before running the MQX RTOS on any of the supported evaluation boards.

All jumper and other hardware switches not specifically described below are expected in factory-default positions. Please refer to the board User’s Guide for the default settings.

7.1 M52223EVB

Important jumper settings:
None.

Board-specific build targets:
None. See section 4.10.2 for more details about standard build targets.

7.2 M52233DEMO

Important jumper settings:
None

Board-specific build targets:
None. See section 4.10.2 for more details about standard build targets.
7.3 M52235EVB

Important jumper settings:
- To enable CAN demo operation (disables UART2)
  - COM_SELA at position 2-3
  - COM_SELB at position 2-3
  - COM_SELC at position 2-3

Board-specific build targets:
None. See section 4.10.2 for more details about standard build targets.

7.4 M52259EVB

Important jumper settings:
- To enable MDIO/MDC communication between processor and Ethernet PHY device (needed in RTCS applications to detect Ethernet link status)
  - J9 at position 2-3 (FEC_MDC)
  - J10 at position 2-3 (FEC_MDIO)
- To enable RTC operation from external crystal
o **H2** at position 1-2
- To enable RTC sourced from battery
  o **J17** at position 1-2

**Board-specific build targets:**
None. See section 4.10.2 for more details about standard build targets.

### 7.5 M52259DEMO

![M52259DEMO board](image)

**Important jumper settings:**
None.

**Other notes:**
- The FEC_MDC pin is shared with GPIO signal used to sense the SW1 button state. The Ethernet link status monitoring is not functional in demos which use SW1 button (all HVAC demos).
- The OSBDM Firmware compatibility issue may affect application debugging. Please see Section 6.8 for more details.

**Board-specific build targets:**
None. See section 4.10.2 for more details about standard build targets.
### 7.6 M52277EVB

**Important jumper settings** (board rev B, schematic D1):

- For USB operation
  - J7 shunt on position 3-4 (VBUSON)
  - J9 shunt on position 1-2 (USB_VBUC_OC)

**Board-specific build targets:**

- **Ext Flash (Debug and Release)** – This target enables a standalone operation from on-board flash memory. External SDRAM memory is used for variables by default. The linker command file can be changed easily to allocate variables in the internal SRAM memory.
  - See also section 4.10.2 for more details about standard build targets.

### 7.7 M54455EVB

**Important jumper settings:**

- To assure correct external Flash mapping as it is assumed by MQX setup
  - SW1[3] must be ON (default setting) so the FLASH0_CS is mapped to FB_CS0 and FLASH1_CS to FB_CS1
- To enable both Ethernet interfaces
  - **SW1[5]** set ON to enable PHY1 (disables ATA interface)

- To enable ATA operation
  - **SW1[5]** set OFF (disables second PHY device)

- For USB Host operation
  - The on-chip transceiver is supported only (external ULPI is not supported).
    Use USB Host connector to connect devices.

- For USB Device operation
  - Use USB Dual connector only to connect to host device.

**Board-specific build targets:**

- **Flash0 Boot (Debug and Release)** – On M54455EVB, this target enables to build applications suitable for booting the system up from the Flash0 memory. The initialization code of the application is located in Flash0. After reset, it copies the rest of the application to SDRAM memory and continues execution there. Configuration file to be used in CodeWarrior FlashProgrammer tool with this target is available as `tools/flash_programmer/config/M54455EVB_EXTFLASH0_512KB.xml`

- **Flash1 Image (Debug and Release)** – This target builds potentially large applications in a form of run-able image. This image can be flashed into Flash1 on M54455EVB and may be started from uBoot or other kind of bootloader. After start, this image copies itself to SDRAM and continues execution there. Configuration file to be used in CodeWarrior FlashProgrammer tool with this target is available as `tools/flash_programmer/config/M54455EVB_EXTFLASH1_16MB.xml`

- See also section 4.10.2 for more details about standard build targets.

### 7.8 TWR-MCF51CN-KIT

**Important jumper settings:**

- To use 25 MHz clock source (BSPCFG_USE_25MHZ_XTAL must be set 1)
- To use 32.768 kHz clock source (BSPCFG_USE_32KHZ_XTAL must be set 1)
  - KTN board, J11 on position 1-2
  - KTN board, J12 on position 1-2

- To enable ADC sensing of the potentiometer in various MQX examples (e.g. security demos)
  - KTN board, J2 on position 2-3 routes ADP3 to MCU_ADP3

- To enable external MRAM memory (available on Memory Storey board)
  - KTN board, J13 on position 2-3 to enable external MRAM latch
  - MEM board, J10 on position 1-2
  - MEM board, J11 shunt removed

- To enable correct Ethernet duplex operation
  - SERIAL board, J12 shunt on pins 15-16

- To enable SD Card operation
  - KTN board, J5 remove three accelerometer shunts ACCX, ACCY, ACCZ
  - KTN board, J3 install three SPI shunts to route SPI signals to memory board
  - KTN board, J2 on position 1-2 to route SPI clock signal to memory board

**Board-specific build targets:**

None. See section 4.10.2 for more details about standard build targets.

### 7.9 TWR-MCF52259-KIT

**Important jumper settings:**

- To enable external MRAM memory (available on Memory Storey board)
  - MEM board, J10 on position 1-2
  - MEM board, J11 shunt removed

- To enable correct Ethernet duplex operation
  - SERIAL board, J12 shunt on pins 15-16

- To enable USB operation in HOST mode
  - SERIAL board, J16 shunt on pins 1-2
  - SERIAL board, J10 shunt on pins 1-2

- To enable USB operation in DEVICE mode
  - SERIAL board, J16 shunt on pins 3-4
  - SERIAL board, J10 shunt on pins 2-3

- To enable SD Card operation
  - MEM Board, J4 on position 1-2 to route QSPI_PCS0 to SD Card Chip Select
MEM Board, J6 remove shunt on pins 1-2 to disable QSPI_PCS0 routing to serial Flash

Board-specific build targets:
- None. See section 4.10.2 for more details about standard build targets. The Ext. MRAM Debug target can be used only with Memory Storey Board.

Other notes:
- The OSBDM Firmware compatibility issue may affect application debugging. Please see Section 6.8 for more details.
8 Change Log

Version 3.0.0 (December 5th 2008)
- This is initial release supporting the MCF52259 processor, M52259EVB and M52259DEMO boards.

Version 3.0.1 (January 22nd 2009)
- Small enhancements throughout the whole code base of MQX kernel and other components
  o Memory block “type” information added to all system memory allocations. TAD is now able to give detailed information about each memory block allocated by kernel or system component.
  o Dedicated memory allocation routines in RTCS, MFS and USB simplify the memory pool usage.
- IPCFG ethernet link monitoring features and automatic IP address binding functionality was added to RTCS. The “ipconfig” shell command replaces the old “bind”, “ifbind” and “dhcp” shell commands.
- The HTTP server in RTCS was re-written to enable multiple sessions to be served by a single task. This feature brings more reliable HTTP server while maintaining low memory requirements.
- USB Host HUB class was added.
- MQX User Guide and several other documents are included in the setup package.

Version 3.1.0 (April 3rd 2009)
- USB Device low-level driver has been implemented for MCF522xx family and example applications were created for M52259EVB and M52223EVB.
- The “usb” subdirectory of the /lib output folder was split to “host” and “device” parts.
- Added PSP, BSPs and other support files for MCF52223 and MCF52235 evaluation boards:
  o USB Host and Device libraries were ported to M52223EVB
  o RTCS library was ported to M52235EVB
  o MFS and Shell libraries ported to both new boards
- USB Host HUB issue resolved with MCF52259 (excessive number of errors observed when USB devices were accessed through USB HUB). The issue was solved by implementation of SOF frame scheduler in the USB Host low-level driver. An issue with a similar impact remains on MCF5222x implementation (silicon problem, no workaround known).
- Error codes naming and numbering convention has been made more consistent across the MQX, RTCS and other libraries. Old error code names remain implemented for backward compatibility (numeric values changed).
- Kernel Data and other internal structures were optimized for size. Parts of the structures were made conditionally compiled based on the user configuration.
- TAD (CodeWarrior debugger plugin) was updated to handle renaming in internal structures.
- Memory type information was added also to lightweight memory structures. TAD is now able to display the type information for memory blocks allocated using lightweight memory allocator.

- Lightweight memory allocator has been made the default option for all BSPs.

- The `/config/<board>/user_config.cw` files were eliminated. Such files have contained a subset of configuration options for assembler-coded files in the MQX kernel. All kernel assembler-coded files are now preprocessed using C preprocessor and make use of standard C header macros. The C-language-specific content of the header files is conditionally excluded during assembler compilation by using `__ASM__` macro.

- The assembler `vectors.cw` file in each BSP was re-coded to C syntax. A new user configuration macro (`MQX_ROM_VECTORS`) may be set non-zero to avoid vector table being copied to RAM.

- The BSP, PSP, RTCS and other library build projects were changed to be easier to understand (virtual folders inside each CodeWarrior project was updated).

- SNMPv2 code was changed to enable ROM-based MIB structures. Example application is provided demonstrating custom MIB nodes and user traps.

- Several source code files were renamed to better reflect the content, especially in the I/O driver directories.

- Added “root directory” concept to the FTP server, “rename” command added.

- SPI I/O driver added (Master mode only). Example application is available.

- Initial ADC I/O driver was added. This driver (including its API) is subject to improvement in future releases. ADC device driver usage is demonstrated in ADC example application.

- RTC API updated, example application added.

- The new IPCFG API updated and documented

- The new HTTP server API documented

- MQX I/O Driver User Guide added

**Version 3.2.0 (May 15th 2009)**

- Added PSP, BSPs and other support files for TWR-MCF51CN-KIT Tower Kit with ColdFire V1 processor MCF51CN128.
  - RTCS library was also ported to MCF51CN128 device.

- Ethernet driver significantly re-written and generalized to cover both ColdFire V1 and V2 processors. Application is now able to change some of the driver parameters dynamically, without need to recompile the driver code.

- Most of the features missing in the previous release of the ADC driver were implemented. The driver documentation is still on the to-do list.

- ADC driver partially re-written and low-level layer ported to ADC module of MCF51CN device. There are implementation differences between MCF51CNxx and MCF522xx drivers which are still to be addressed by future releases.
  - MCF51CN driver uses internal lightweight timer component
  - MCF522xx driver still uses PIT timer for internal timing. Implementation with lightweight timer (lower performance) is planned for future versions
- New GPIO callback-on-interrupt feature was added to GPIO driver of MCF51CN. This feature will be ported to other supported devices in future releases.

- Three new “security” demo applications were created for MCF51CN device demonstrating the ADC and GPIO drivers, network protocols and low-power mode of operation. The applications will be ported to other devices in future.

**Version 3.2.1 (May 28th 2009)**

- Sparse interrupt table implemented and made a default option for small-RAM devices. This feature saves up to 1k of RAM. The TAD CodeWarrior plug-in was updated to support this feature.

- The legacy MQX Flash driver (called FlashX) was ported to support internal Flash memory of all supported devices (MCF51CNxxx, MCF522xx). The Flash driver requires Flash parameters to be passed from the application or linker command file. The linker command files were updated in all BSP. Sample application is provided.

- The GPIO driver callback-on-interrupt feature was ported to MCF52xx devices. This feature is still a subject to change in future versions. The MCF51xx driver was optimized for code size.

- Minor bug fixes in Serial driver (fflush system call implemented).

- MCF51CNxx SPI example enhanced to support both SPI EEPROM and SPI Flash devices. The example applications may be used with Memory Storey board.

- Fixed known issue with MQX Stationery as it was described in the last release notes. The MQX_PATH named source tree is no longer used for referencing the debugger initialization file (this was causing Flash Programmer crash). Copies of debugger initialization and memory configuration files were made part of all stationery projects and are referenced using relative path only in the debugger settings.


**Version 3.3.0 (Aug 7th 2009)**

- Added BSP for MCF52259 Tower Kit.
  - CodeWarrior projects, demos and example applications were ported from M52259EVB to this BSP.
  - CodeWarrior projects and the BSP also support debugging targets and running code from external MRAM memory. This is prepared for future support of Memory Storey board for the Tower Kit.

- Added PSP, BSPs and other support files for M54455EVB ColdFire V4 system. This is the first V4 device supported by Freescale MQX operating system. Few new features were added to PSP:
  - Cache control support added. Non-cached memory pool allocation API added.
  - Flash-to-RAM code copying enabled in startup.
  - Startup code is now part of PSP. Two files (`CF_startup.c` and `ROMCopy.c`) were reused from CodeWarrior runtime library. These files were modified to support ROM-to-RAM code copying.
  - Flash Programmer config files available in `tools/flash_programmer_scripts/config` directory.
SPI driver for M54455EVB created (the DSPI module), example supports onboard SPI Flash

- Ethernet driver and RTCS were significantly re-written.
  o Device-independent part of the driver was re-written to support multiple Ethernet MAC devices of the same or even different kinds. A support for memory-optimized handling of small frames was also added.
  o Device-dependent part of the Ethernet driver was re-written to support ColdFire V1-V4 Fast Ethernet Controller module.
  o Several RTCS features were re-tested and fixed to support multiple physical controllers.

- RTCS updated
  o Shell example now uses the MFS lib (with RAM-disk) and supports all FTP commands (conditional compilation)
  o Minor changes in IPCFG API to support multiple devices, potentially of different kind than ENET.

- TAD updated (see section 4.12.3 for compatibility considerations)
  o Strings displayed in TAD moved to separate text files available inside MQX installation.
  o New Ethernet driver screen added.
  o Symbolic reader enhanced some objects (LWsem, LWevent, ...) are displayed not only with address information, but also with symbolic names.

- Several key TAD-like screen dumps implemented also on embedded application side in PSP. These functions may be used to print out the TAD-like debugging data by the application.

- Shell library updated
  o RTCS support commands changed (ipconfig, netstat).
  o MFS format command was made available (also see mfs/examples/ramdisk application)
  o Shell interface to TAD-like screens dumps added.

- Workaround for MCF5223x Ethernet PHY auto-negotiation issue implemented according to chip errata.

- FlashX driver updated:
  o Support of external Flash devices made available. Tested with external flash memory devices of M54455EVB.
  o FlashX example application modified to support external flash1 device on M54455EVB.
  o Write protect ioctl command implemented for external Flash devices.

- USB Host updated:
  o Code refactored and generally updated (MASS storage class renamed to MSD).
  o CDC class added. Example application shows how to forward characters between standard UART TTY and a virtual USB TTY device.

- I2C driver refactored and updated, eeprom example changed.

Freescale MQX Release Notes
- CodeWarrior “Stationery” templates for creating new MQX projects were changed:
  - All kinds of stationery projects now have full set of libraries added to project (BSP, PSP, MFS, RTCS, USB Host, USB Device, Shell). The standard CodeWarrior linker will optimize out the unreferenced code. Feel free to remove the unused libraries from the project.
  - MQX-Only stationery is a simple “hello-world” like example.
  - MQX+MFS stationery is simple shell example working with RAM disk.
  - MQX+RTCS stationery is simple shell example with basic networking commands and telnet server.
  - MQX+RTCS+MFS stationery is a union of the two examples above.
  - USB Host is a simple application able to detect devices attached to the host port or devices attached through USB hub. You still need to refer to the USB host example applications to see how to interface attached devices.
  - The USB Device stationery is not available in this release. Please refer to the USB Device example applications.

- Device-specific sections added to this release notes document. Please read the required jumper settings for each evaluation board supported.

Version 3.4.0 (Sep 25th 2009)
- See Section 2 “What is New” above.