Halo Release for NXP MQX™ RTOS 4.2

Release Notes

PRODUCT:	Halo Release for NXP MQX™ RTOS 4.2.0
PRODUCT VERSION:	1.6
DESCRIPTION:	Adding support for Halo MAC57Dxx to NXP MQX™ RTOS 4.2.0
RELEASE DATE:	Jan 9, 2017

How to Reach Us:

Home Page:
www.nxp.com

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

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

Europe, Middle East, and Africa:
NXP 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.nxp.com/support

Japan:
NXP 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@nxp.com

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

For Literature Requests Only:
NXP 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
LDCForNXPSemiconductor@hibbertgroup.com

Rev. 1
11/2010

Embedded Access Inc

Table of Contents

1. Read Me First

This release note documents the release Halo Release for NXP MQX™ RTOS version 4.2, released for the NXP MAC57D5xx processor.

For more detailed information about MQX, see NXP MQX™ 4.2 Release notes and Getting Started documents.

2. Requirements

2.1. Development Tools

This NXP MQX™ RTOS Release was compiled and tested with the following development tools:

- IAR Embedded Workbench version 7.50.1
- See MQX build projects suitable for IAR environment in `iar` subdirectories
- NXP S32 Development System version 1.3
- See MQX build projects suitable for IAR environment in `s32ds` subdirectories

2.2. System Requirements

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

2.3. Target Requirements

The Halo Release was tested with the following hardware configuration:

- MAC57D5-516DC daughter board and interface board
- MAC57D5MB mother board
- IAR I-jet with Seger J-Link adapter or P&E Multilink (for S32 tools)
- Tower Serial Card with 3 jumper cables

What's New

Version 4.2

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

- Initial Pre-Release
- Support for all standard M4 and A5 drivers
- Multicore support
- Open Vector Graphics support for both HDMI and TFT displays ports
- TinyUI support

3. Release Overview

This MQX Release brings updated support for the Halo platform. Support for standard M4 and A5 drivers are supplied. Support for driver selection and control through the PSP core configuration. Support for S32 development is added in this release.

4. Supported Features

4.1. M4 Core Support

I/O drivers including ePIT, GPIO, LINFlex (UART), FlexCAN, FTM.
Interrupt and Polled I2C and Interrupt and DMA SPI drivers are supported.
DMA support, flashx, Ethernet, Real Time Clock (RTC), SDCARD, ADC

4.2. A5 Core Support

GIC, LINFlex (UART), PIT, FlexCAN, FTM.
Interrupt and Polled I2C and Interrupt and DMA SPI drivers are supported.
DMA support, Ethernet, Real Time Clock (RTC), SDCARD, ADC, OVG, TinyUI

4.3. Example Applications

MQX 4.2 Halo Release contains applications that demonstrate kernel and peripheral functionality on the Halo board. The applications can be found on following location:

`<install_dir>/mqx/examples` - a limited set of examples for kernel features and basic peripheral drivers

- hello (M4 and A5 multicore debug)
- gpio
- can
- rtc
- ddr_stress_test
- spi
- demo
- event
- i2c
- lwadc

- flashx (M4 only)
- lwdemo
- multicore

- `<install_dir>/mfs/examples` - a limited set of examples for mfs
  - mfs_sdcard

- `<install_dir>/rtcs/examples` - a limited set of examples for rtcs
  - rtcs_shell

`<install_dir>/mcc/examples`
- pingpong

`<install_dir>/ovg/examples`
- dual_screen
- logo
- pbuffer_1
- pbuffer_2
- tiger

`<install_dir>/tui/tests`
- tui_performance

Release Content

This section gives an overview of the release content. Where <tool> is either iar or s32ds

Deliverable	Location
Pre-compiled MQX Libraries	<install_dir>/lib/...
MQX BSP M4	.../lib/mac57d5mb_m4.<tool>/debug/bsp
MQX BSP A5	.../lib/mac57d5mb_a5.<tool>/debug/bsp
MQX PSP M4	.../lib/mac57d5mb_m4.<tool>/debug/psp
MQX PSP A5	.../lib/mac57d5mb_a5.<tool> /debug/psp
MQX MFS (File System) M4	.../lib/mac57d5mb_m4.<tool>/mfs
MQX MFS (File System) A5	.../lib/mac57d5mb_a5.<tool>/mfs
MQX Shell Library M4	.../lib/mac57d5mb_m4.<tool>/shell
MQX Shell Library A5	.../lib/mac57d5mb_a5.<tool>/shell
MQX Multicore M4	.../lib/mac57d5mb_m4.<tool>/mcc
MQX Multicore A5	.../lib/mac57d5mb_a5.<tool>/mcc
MQX PSP Source Code and Examples	<install_dir>/mqx/...
MQX PSP source code for Cortex A5	.../mqx/source/psp/cortex_a
MQX PSP source code for Cortex M4	.../mqx/source/psp/cortex_m
MQX PSP common source code	.../mqx/source/psp/cpu
MQX example applications	.../mqx/examples/...
MQX BSP Source Code	<install_dir>/mqx/...
MQX BSP source code for Halo MAC57D EVB board M4 core	.../mqx/source/bsp/mac57d5mb_m4
MQX BSP source code for Halo MAC57D EVB board M4 core	.../mqx/source/bsp/mac57d5mb_a5
MQX BSP source code for Halo MAC57D EVB board M0+ core	.../mqx/source/bsp/mac57d5mb_m0p
MFS Source Code and Examples	<install_dir>/mfs/...
MFS source code	.../mfs/source
MFS build projects	.../mfs/build/<tool>
MFS example applications	.../mfs/examples
MCC Source Code and Examples	<install_dir>/mfs/...
MCC source code	.../mcc/source
MCC build projects	.../mcc/build/<tool>
MCC example applications	.../mcc/examples
RTCS Source Code and Examples	<install_dir>/rtcs/...
RTCS source code	.../rtcs/source
RTCS build projects	.../rtcs/build/<tool>
RTCS example applications	.../rtcs/examples
Shell Library Source Code	<install_dir>/shell/...
Shell source code	.../shell/source
Shell build projects	.../shell/build/<tool>
OVG Source Code and Examples	
OVG library code	.../ovg/lib
OVG source code	.../ovg/source
OVG build projects	.../ovg/build/<tool>
TinyUI Source Code and Examples	
TUI example code	.../tui/tests
TUI source code	.../tui/source
TUI build projects	.../tui/build/<tool>
IAR Support	<IAR_dir>/...

Deliverable	Location
MQX Task-aware Debugger plug-in for IAR	…/tools/iar_extensions
PC Host Tools	<install_dir>/tools
TFS Make Utility	…/tools/mktfs.exe
Check for Latest Version tool	…/tools/webchk.exe
Documentation	<install_dir>/doc
User Guides and Reference Manuals for MQX RTOS, RTCS, MFS, IO Drivers, etc.	…/doc

5. Known Issues

SDRAM DDR memory should be disabled in user_config.h if board does not have mod and power jumpers installed. M4 and A5 cores will compete for shared LINFlex UART it is up to application to implement semaphores.
With S32DS workspaces first time builds may require a second invocation to create non-existent directories.

5.1. Tool Chain Versions

Using older versions than Workbench 7.5 (compiler 7.3) of the IAR tool chains will cause problems:

Code generation issues were reported in earlier versions. Support for MAC57D5xx is not available.

For S32 releases before release (ARM v1.3) have debug issues.

6. Board-Specific Information Related to Halo Release

All jumper and other hardware switches not specifically described below are expected in factory-default positions. See the Halo board user's guide for the default settings.

To test DDR memory a mod is required and 2 extra jumpers. See DDR Modifications notes.

DDR software configuration must be disabled if mods and jumpers are not installed.

6.1. Halo MAC57DMB development board

The Halo Release supports the following hardware configuration:

- Halo MAC57D54H

The default serial port is set to use the serial port on the MAC57DMB J20. The signals are at board levels so a RS232 driver is required to connect to a standard serial port. Note: A Tower Serial board can be use to convert the signal levels.

6.1.1. Board-specific build targets:

The Halo libraries support 1 build targets, Only the Debug build, the Release build is not currently supported target for ARM mode.

The Halo applications support Flash Int Debug- these targets build applications suitable for downloading directly to the Internal Flash.

Additional targets for will be available in the next release.

See the MQX Getting Started document (chapter 2.2) for more details about standard build targets.

6.1.2. Board-specific workspaces

The Halo release contains the following workspaces:

```
build\mac57d5mb_m4\iar\build_libs.eww
build\mac57d5mb_a5\iar\build_libs.eww
```

This workspace contains the MQX library projects, and is used to build the MQX libraries. The following build order is recommended: PSP, BSP, MCC, MFS, OVG, TUI, RTCS, SHELL.
Projects must be rebuilt after installation.

The hello workspace contains a link to a project to build a sample M0+ load which may be required for other projects so it is recommend to build this project before other examples. Currently this code is not supported and the M0+ core must be disabled in user_config.h

```
..mqx\examples\hello\build\iar\hello_mac57d5mb_m4.eww
```

NOTE: Not all examples are designed for the Halo platform.

For S32 workspaces and the meta file are in the <example>/build/s32ds folder. BSP, PSP and libraries must be built in correct order A5 first including example app then M4. To reduce the release size not all workspace meta files are included but can be recreated by importing required projects into a workspace. It is recommend to use the example/build/s32ds directory as a location for the workspace.

7. Quick Start

7.1. Running Hello Example

Required Software:

- IAR Embedded Workbench Version 7.5 or later
- NXP MQX 4.2 (standard distribution)
- NXP MQX 4.2 Halo release for MAC57D

To use the Halo Release, begin by installing the software as follows:

1. Install IAR tools to the default directory.
2. Install standard NXP MQX 4.2 release to its default directory. Doing so will add the MQX Task Aware Debug feature to the IAR installation, and set the appropriate register entries.
3. Unzip the Halo Preliminary Phase 1 Release to `c:\NXP\NXP MQX 4.2 Halo`.

To rebuild the MQX Libraries:

1. Launch IAR Embedded Workbench.
2. Open the `...\mqx\examples\hello\build\iar\hello_mac57d5mb_a5\hello_mac57d5mb_a5.eww` workspace. (Note: build A5 workspace before building M4 workspace)
3. Build all three projects (BSP, PSP, hello_mac57d5mb_a5 –**Flash Int Debug**)
    a. Select the project by clicking on the project tab.
    b. Make the project by typing F7, or by selecting Project, Make on the menu.
    c. Choose and make the hello_mac57d5mb_a5 –**Flash Int Debug** by selecting the workspace drop down menu and pressing F7 or right click and select Make.
4. Close the A5 workspace
5. Open the `mqx\examples\hello\build\iar\hello_mac57d5mb_m4\hello_mac57d5mb_m4.eww` workspace.
6. Build each of the projects (BSP, PSP, coreexample_cm0p –Debug, hello_mac57d5mb_m4 – **Flash Int Debug**):

Hardware set-up:

1. Attach the IAR I-jet to the JTAG connector of the MAC57d5MB board with Seger J-Link adapter.
2. Attach jumper wires from J20 on MaC57d5MB to tower serial J17, J19, Ground. (see schematics). Be sure to supply 5V power to Tower Serial with Mini USB connector.
3. Attach a DB-9 cable connector between Tower Serial DB9 and your PC or USB to serial adapter.
4. Connect the power supply, and power up the board and Tower Serial Board.
5. Insert the J-Link USB cable into your PC.
6. Allow Windows to install the J-Link driver.

To run the MQX examples:

1. Launch IAR Embedded Workbench.
2. Open the `mqx\examples\hello\build\iar\hello_mac57d5mb_m4\hello_mac57d5mb_m4.eww` workspace.
3. For other examples open the workspace in the example folder:

Select the project (by clicking on the project tab) and make the example hello the active project.

Make the project by typing F7, or by selecting Project, Make on the menu.

Download the project to the target by typing ctrl-D, or by selecting Project, Download and Debug on the menu. Or by pressing the green debug arrow.

Note that the first time you use the J-Link, it may ask for permission to update the J-Link software. The J-Link software should be updated.

When the hello project runs it will open a slave window for the A5 project using the A5 workspace. When both workspaces have opened run the M4 project this will set the A5 core to the start location. Then start the A5 core. To debug, reset the A5 core or M4 core. The A5 core will return to the reset vector you can then set break points and step through the A5 core.

Note: The A5 core must be running with valid code before it can be attached to, and debugged. There must be some valid code to run in the A5 so IAR can connect the debugger and halt the processor. The M4 downloads and programs all binaries and starts all cores simultaneously via a WFI reset. Currently there is no valid code for the M0+ core so it must be disabled. However the hello example demonstrates how you could build and load the M0+ with sample code.

7.2. Running A Driver Example

Under the example build directories, there will be 2 directories. One will have an '_m4' suffix and '_a5' suffix. For example, for RTCS_Shell:

```
...\rtcs\examples\shell\build\iar\rtcs_shell_mac57d5mb_m4
...\rtcs\examples\shell\build\iar\rtcs_shell_mac57d5mb_a5
```

Each directory contains the project files for running the test on the core indicated by the suffix.

7.2.1. Examples on M4 core

Tools and software requirements are the same as for the Hello example.

The M4 example projects are run in the same way as any standard single core project:

1. Launch IAR Embedded Workbench.
2. Open the m4 project e.g. `...\rtcs\examples\shell\build\iar\rtcs_shell_mac57d5mb_a5\rtcs_shell_mac57d5mb_m4.eww` workspace.
3. Build each of the projects (BSP, PSP, rtcs_shell_mac57d5mb_m4 – **Flash Int Debug**):

Download the project to the target by typing ctrl-D, or by selecting Project, Download and Debug on the menu. Or by pressing the green debug arrow.

7.2.2. Multi-Core notes

After the initial system startup reboot, the system undergoes a soft reset with all enabled cores starting at the same time. There needs to be a valid executable binary in flash for all active cores, otherwise a runaway core can bring down the system.

i.e. When running an M4 test, if the A5 core is enabled it will simultaneously run both the M4 and whatever load had been previously flashed for the A5.

Because we have an initial M4 boot, then reset followed by all cores come up simultaneously, the A5 examples each have 2 projects:

- A master project that runs on the M4
- A slave project that runs on the A5.

Master Project

The master project runs on the M4 and its function is to write the A5 binary to flash before starting the slave project, which can now run this flashed binary.

When the master project is built, it will contain:

- An M4 load. This is MQX 4.2 with a stub application file that is identical to the Hello example, since we are more interested with the software on the A5.
- The master project links the .bin file generated by the slave build. (e.g. rtcs_shell_mac57d5mb_slave_a5.bin) so that this binary is also flashed when the master project is loaded. **Note:** The slave project must be built before the master.

Slave Project

The slave project has 2 functions:

- It builds the A5 binary, and this binary is linked with the master project to be written to flash.
- When IAR runs the master project, it will start up a second slave instance of IAR running the slave project. This slave instance connects to the A5 that is running the slave binary for debugging.

7.3. Examples on the A5 core

As an example, this section will cover the RTCS_Shell example. This project can be found under ...\rtcs\examples\shell\build\iar\rtcs_shell_mac57d5mb_a5.

First, build the slave project:

1. Launch IAR Embedded Workbench.
2. Open the ...\rtcs\examples\shell\build\iar\rtcs_shell_slave_a5 workspace.
3. Build the BSP and PSP projects.
4. Build the MFS, Shell and RTCS projects.
5. Choose and make the rtcs_shell_mac57d5mb_slave_a5 –**Flash Int Debug** by selecting the workspace drop down menu and pressing F7 or right click and select Make.

This should generate the rtcs_shell_mac57d5mb_slave_a5.bin file. You can close IAR at this time.

Now start the master project

1. Launch IAR Embedded Workbench.
2. Open the ...\rtcs\examples\shell\build\iar\rtcs_shell_master_m4 workspace.
3. Build the BSP and PSP projects.
4. Choose and make the rtcs_shell_mac57d5mb_master_a5 –**Flash local** by selecting the workspace drop down menu and pressing F7 or right click and select Make.

Download the project to the target by typing ctrl-D, or by selecting Project, Download and Debug on the menu. Or by pressing the green debug arrow.

When the RTCS_Shell project runs it will open a slave window for the A5 project using the A5 workspace. When both workspaces have opened run the M4 project this will set the A5 core to the start location. Then start the A5 core. To debug, reset the A5 core or M4 core. The A5 core will return to the reset vector you can then set break points and step through the A5 core.

Note: The A5 core must be running with valid code before it can be attached to and debugged. There must be some valid code to run in the A5 so IAR can connect the debugger and halt the processor. The M4 downloads

and programs all binaries and starts all cores simultaneously via a WFI reset. Currently there is no valid code for the M0+ core so it must be disabled. However the hello example demonstrates how you could build and load the M0+ with sample code.

Hardware set-up:

The setup is identical to the Hello example above, with the addition of connecting the ethernet port to a local network.

8. Core Configuration

The A5 and M0+ core can be enabled or disabled in the M4 project user_config.h file.

```
#define BSPCFG_ENABLE_A5_CORE          1  // A5 enabled
#define BSPCFG_ENABLE_M0P_CORE         0  // M0+ disabled
```

In examples where a core is not used it is important to disable it, otherwise the core will run any code that was flashed previously or invalid code which can take down the system.

Problems can arise if more than one core tries to control a single instance of a device. Default device to core assignments are specified in core_cnfg_MAC57D54.h, and you can over-ride these in user_config.h. This topic is covered in section 3.3 of the document 'MQX_Using_Multicore.pdf' under ...\doc\mqx.