Document Number: MQXRTOS
Rev. 04, 04/2013

Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	Freescale MQX 4.0.1
DESCRIPTION:	Using IAR Embedded Workbench Tools with Freescale MQX™ RTOS
RELEASE DATE:	April 19th, 2013

© 2013 Freescale Semiconductor, Inc. All rights reserved.

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off.
All other product or service names are the property of their respective owners.
© 1994-2008 ARC™ International. All rights reserved.
© 2008-2013 Freescale Semiconductor, Inc. All rights reserved.

Document Number: MQXRTOS
Rev. 04, 04/2013

Table of Contents

Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench i

1 Read Me First 2

2 Building the MQX Libraries 3

2.1 Batch Build in IAR Embedded Workbench IDE 3

3 Running and Debugging the MQX application 4

3.1 Run MQX Hello World program 4
3.2 Multi-core debugging 5
3.3 Debugging (attach) the Application loaded by MQX Boot Loader 9
3.4 MQX Task Aware Debugging 10

4 Using the MQX DebugIO Driver with EWARM IDE 15

1 Read Me First

This document describes steps required to configure the IAR Embedded Workbench development tools and use it to build, run, and debug applications of the Freescale MQX™ RTOS operating system. Refer to “Getting Started” and other user documentation included within the latest Freescale MQX™ RTOS installation for more details not specifically related to IAR Embedded Workbench tools.

Get the latest Freescale MQX™ RTOS at http://www.freescale.com/mqx.

2 Building the MQX Libraries

See Chapter 2 of the MQX Getting started document for details on generic build process and compile time configuration. This document concentrates on steps specific to IAR tool chain only.

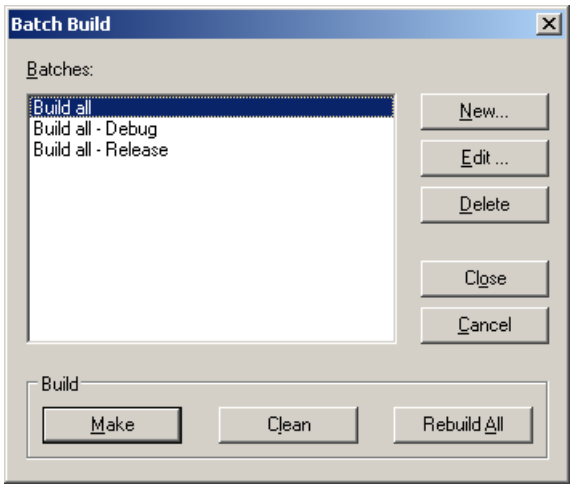
2.1 Batch Build in IAR Embedded Workbench IDE

With IAR, the MQX build process can be simplified by using Batch Build feature. For each supported board, an IAR Workspace file with all MQX libraries is provided as follows:

`<install_dir>/config/<board>/iar/build_libs.eww`

The Workspace file contains Batch Build configurations which make it possible to build all MQX libraries at once.

- Go to menu “Project / Batch Build” or press the F8 key in the IAR IDE.
- Select Batch configuration to build (refer to next section for more details about build targets)
- Press the “Make” button to start the batch build process.



3 Running and Debugging the MQX application

The description bellow is provided for twrk60n512 BSP and Hello World example application. The same procedure applies for all other BSPs and examples.

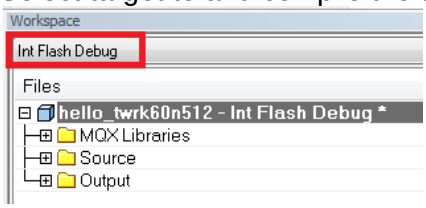
3.1 Run MQX Hello World program

Loading and debugging MQX applications is an easy task with IAR Embedded Workbench tool and it is not different from debugging classic non-OS applications. Make sure that you select the correct debugger interface in the project options and that you use the correct processor initialization Macro file.

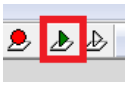
- Connect a serial cable to the TWR-SER DB9 connector. Set the communication speed to 115200.
- Select menu ***Project/Add Existing Project*** and select Hello World example application:

<mqx_install_dir>/mqx/*examples/hello/iar/hello_twrk60n512.ewp*

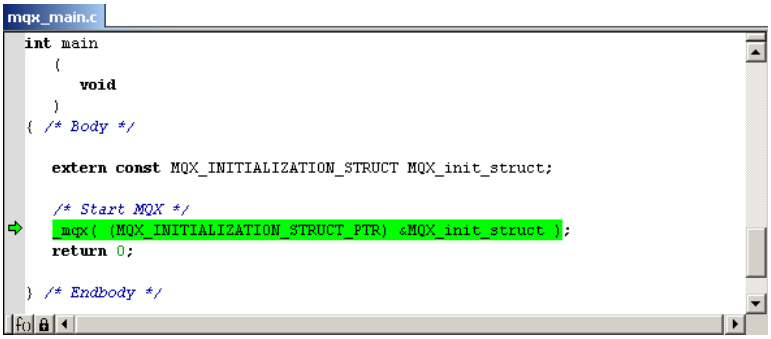
- Select target to and compile the application using ***Project/Make menu***.



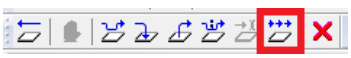
- When a MQX application is compiled and linked to all MQX libraries, press the debug button to download the application to target.



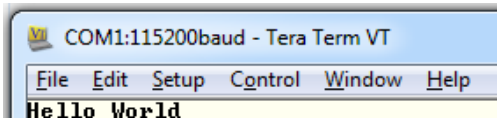
The application gets executed and stops at the default C language entry point in the *main()* function. Be aware that, at this breakpoint, the MQX Operating System is not yet running. Therefore, the use of TAD plugin features, as described in subsequent sections, is limited.



- Press the run button to start the application.



- Hello World is printed on serial console terminal.



3.2 Multi-core debugging

This chapter describes the basics of multi-core debugging with MQX RTOS. Description is provided for twrvf65gs10_a5 a twrvf65gs10_m4 (Vybrid) board support package and MCC (multi-core communication) example application. In this case, the CortexA5 is the primary core while the CortexM4 is set up as an auxiliary core.

3.2.1 Debugging with JLINK

Before you start, ensure that the IAR 6.50.2 or newer and the latest JLINK drivers from SEGGER is used.

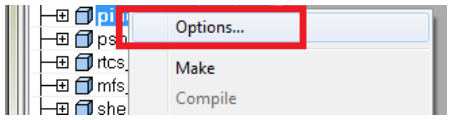
- Open two instances of IAR IDE. In the first IDE instance, open the CortexA5 example application. In the second IDE instance, open the application for CortexM4 core.

`<mqx_install_dir>/mcc/examples/pingpong/iar/pingpong_example_twrvf65gs10_a5`

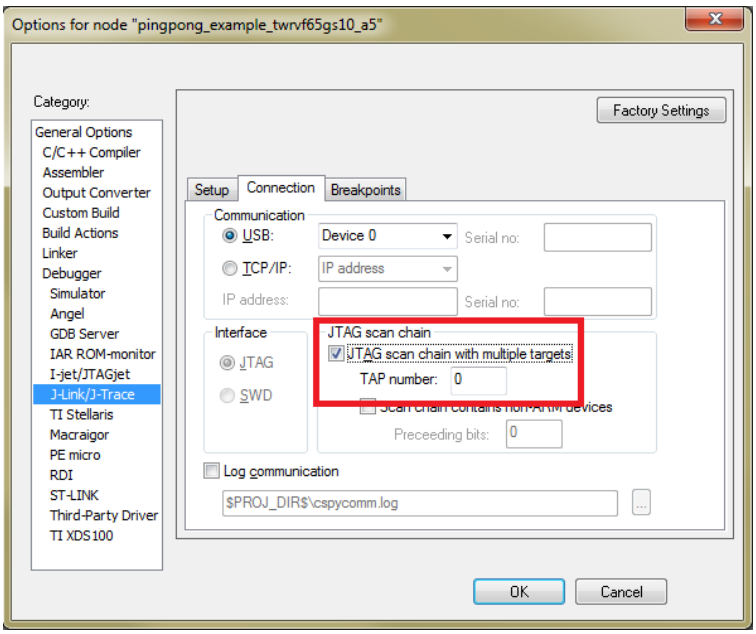
And

`<mqx_install_dir>/mcc/examples/pingpong/iar/pingpong_example_twrvf65gs10_m4`

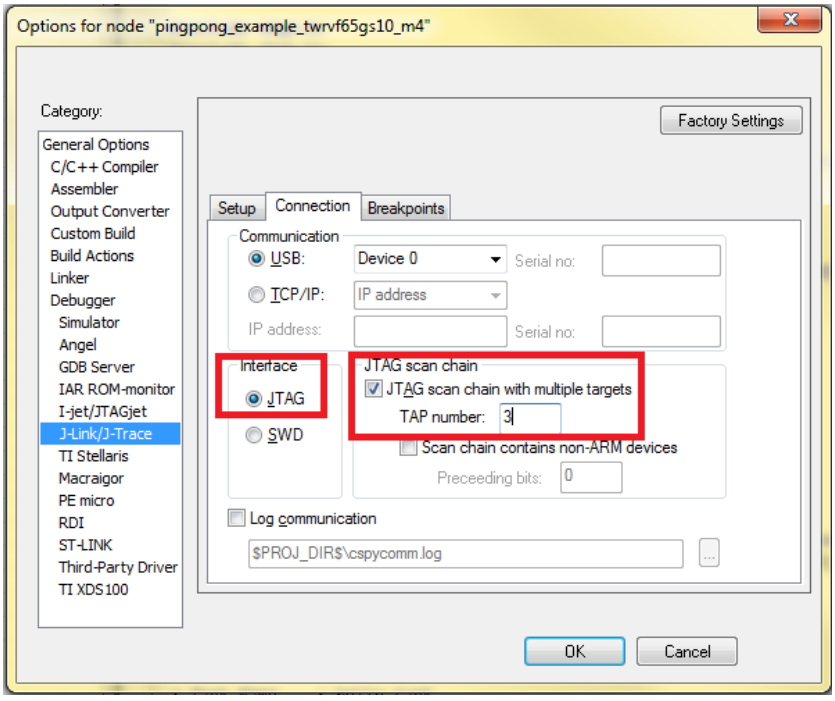
- Ensure that both MQX and MCC (`<mqx_install_dir>/mcc/build`) libraries are compiled for each core prior to the compilation of the example application. Set the following parameters before debugging:
- For primary core (CortexA5), set in project *Options*:



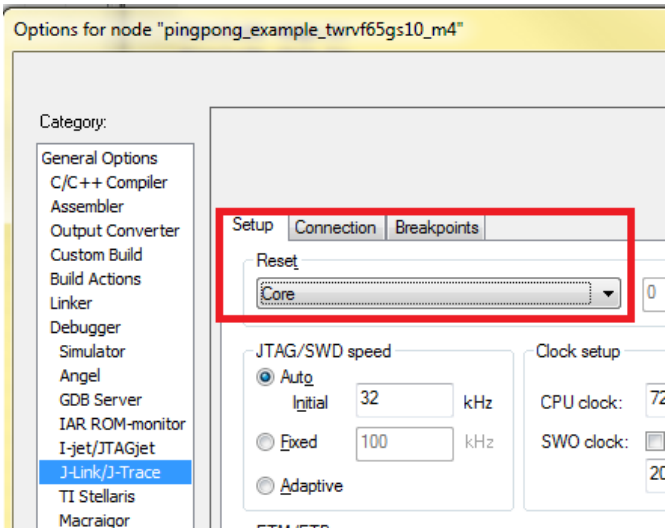
Set “JTAG with multiple targets” and TAP number “0” in the JLINK/J-Trace setting.



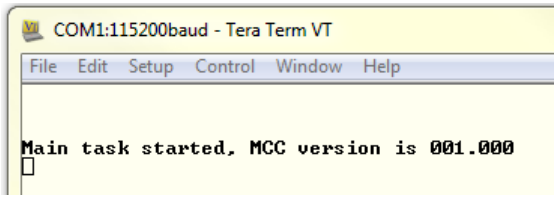
- For secondary core, select “JTAG with multiple targets” and TAP number “3” in the JLINK/J-Trace setting. The number 0 and 3 are the indexes of the CPUs on JTAG chain. Since this can differ on other processors, see your processor Reference Manual for details.



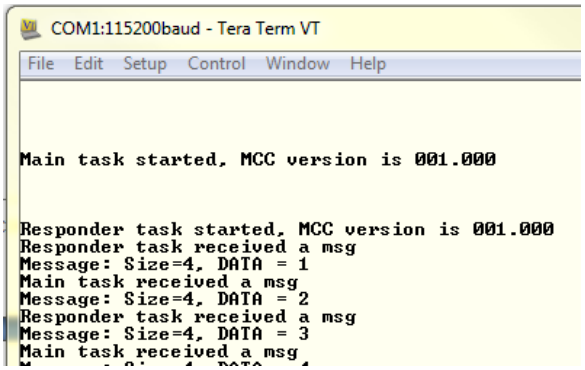
- Set the Reset type to “Core.”



- Start the primary core (CotexA5) application. You should see the following message on the terminal:



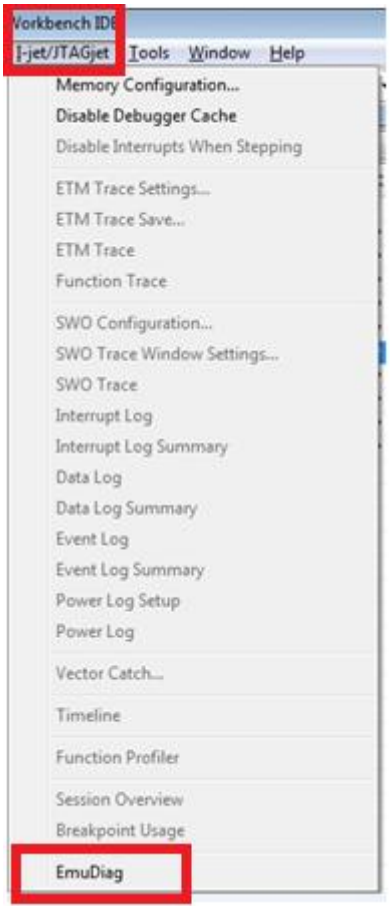
- Start the auxiliary core (CortexM4) application. The responder will be started and message “pingpong” will start between the cores.



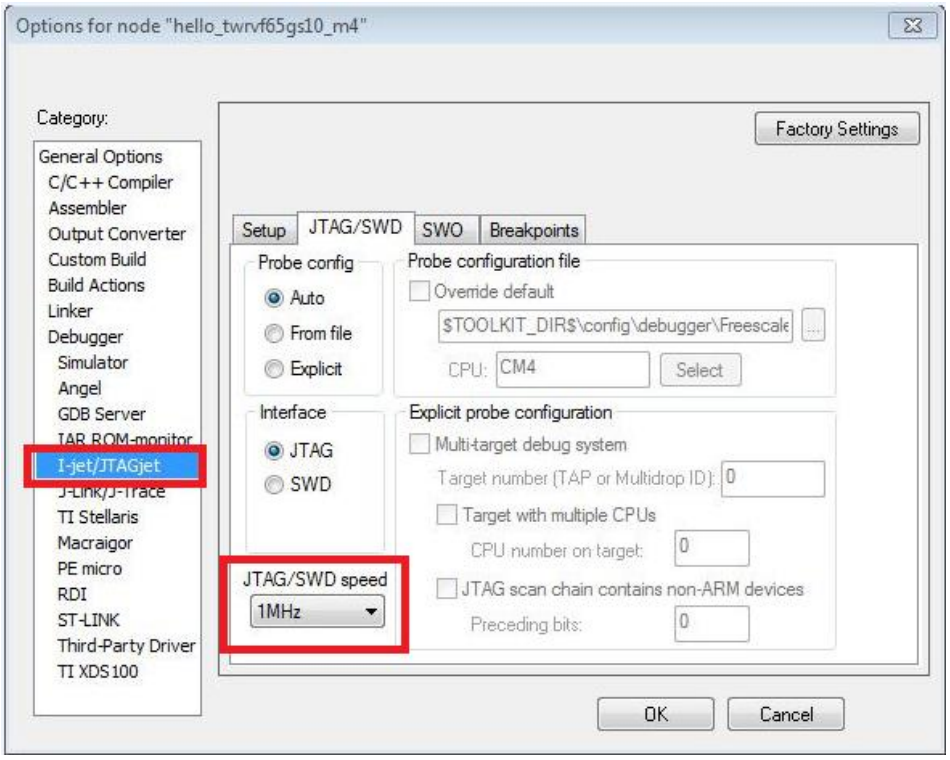
3.2.2 Debugging with I-Jet

Debugging the multicore with I-Jet probe does not offer a possibility to run two debug instances at once. Only one core can be under the debugger at any one time. Special settings are needed to change this. See the steps bellow:

- First, the debugger needs to switch to I-jet got you I-Jet/JTAG-jet menu and update Firmware in EmuDialog Menu.



- Choose "arm\bin\jet\firmware\i-jet\v2\I-JET.upg"
- The frequency needs to be set to 1Mhz. The autodetect does not work correctly. This will be solved in future IAR versions.



3.3 Debugging (attach) the Application loaded by MQX Boot Loader

This chapter describes debugging of the application loaded to the processor memory by MQX Boot Loader. The similar approach can be used for debugging an application loaded by a different boot loader e.g. U-Boot. This chapter also briefly describes the steps required for preparing bootable SD Card image and application images in IAR tool set. For detailed information on Vybrid Boot Loader usage, see Readme.txt located in the MQX Boot Loader application folder.
(*<mqx_install_dir>/mqx/examples/bootloader_vybrid/Readme.txt)*

Building Boot Loader and creating bootable SD card

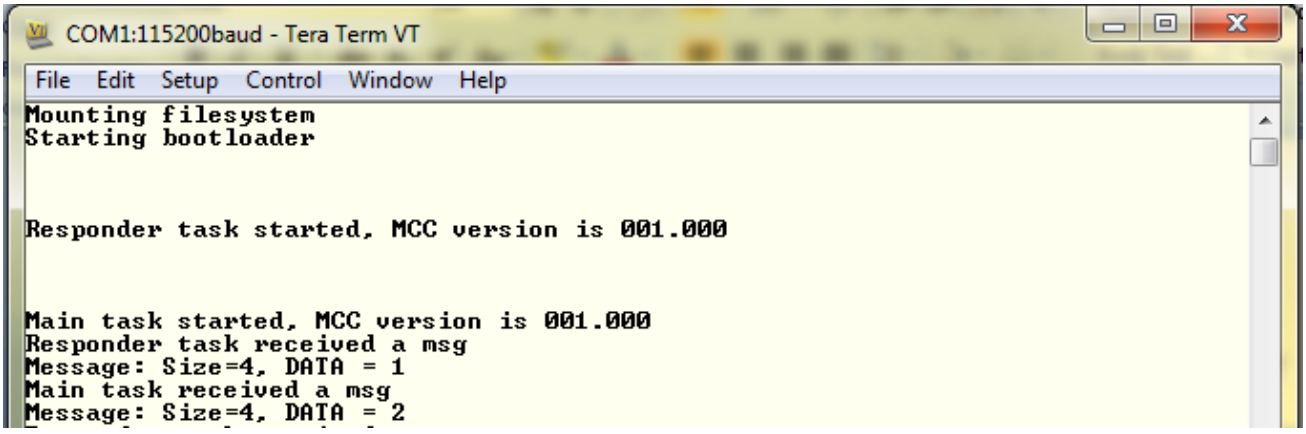
- Import the MQX Boot Loader project to your workspace by using ***Project/Add Existing Project*** menu.
- Select the bootloader_vybrid from your MQX installation directory.

<mqx_install_dir>/mqx/examples/bootloader_vybrid/iar/bootloader_vybrid_twrvf65gs10_a5

- Select Int Ram Debug target and compile the application using ***Project/Make menu***.
- Follow *<mqx_install_dir>\mqx\examples\bootloader_vybrid\Readme.txt* description and use prepare binary image to prepare bootable SD Card.

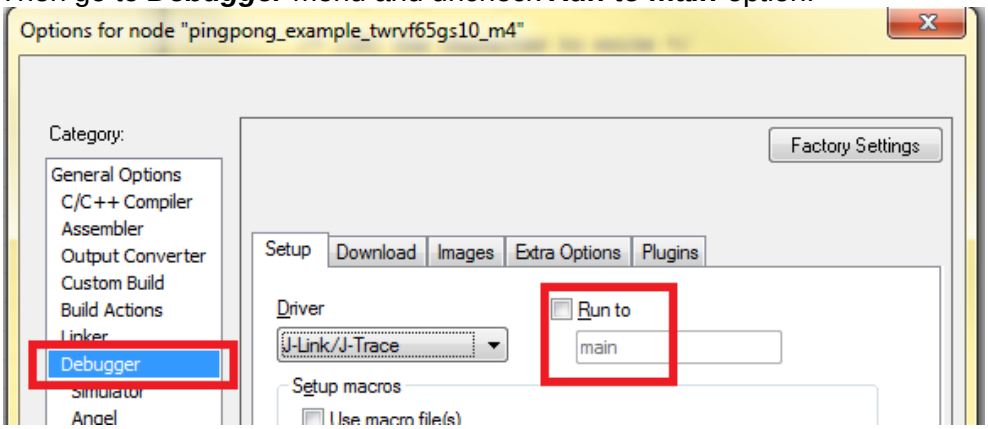
Building and Debugging the Application images

- Build the applications you want to run on A5 and M4 cores the usual way.
- Store the binary images to root directory on bootable SD card.
- Copy setup.ini to the SD Card and modify according to Readme.txt description.
- Remove the SD Card from the PC and plug it into Micro SD Card slot on your Vybrid board.
- Power up the Vybrid board. MQX Boot Loader will print out the following message on default console (RS232 TWR-SER) and start execution of M4 and A5 applications.

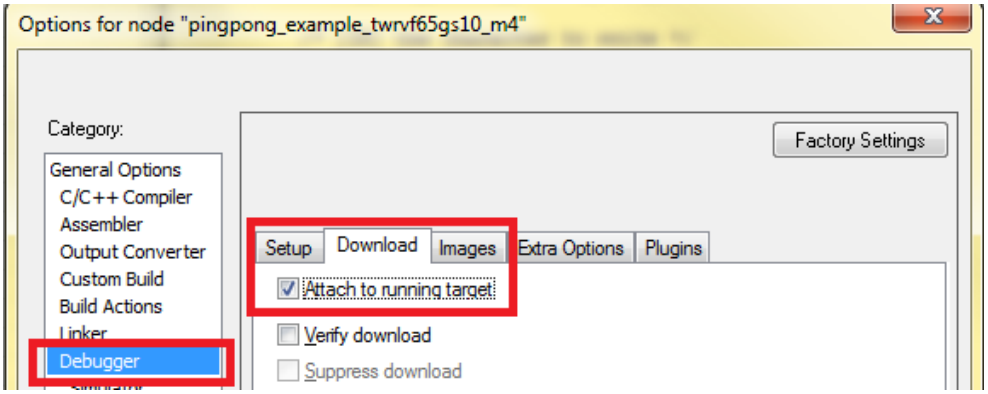


- To debug the running application, go to project properties and setup ***JLINK/JTRACE*** settings as described in the previous chapter.

- Then go to ***Debugger*** menu and uncheck ***Run to main*** option.



- Select Attach to running target.



- Use the Debug without downloading button . The debugger will connect to the selected application. You can then stop the selected core and debug the booted image.

3.4 MQX Task Aware Debugging

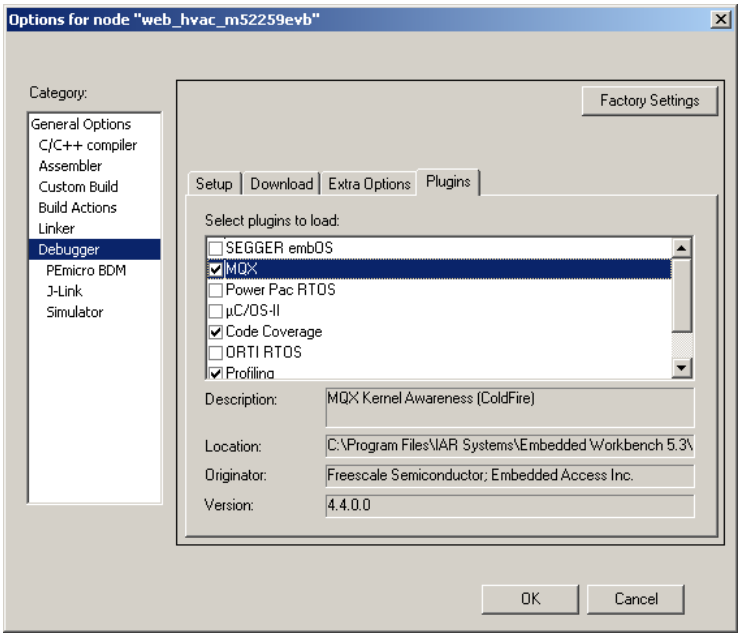
MQX Task Aware Debugging plug-in (TAD) is an optional extension to a debugger tool which enables easy debugging of multi-tasking applications. It helps to visualize internal MQX data structures, task-specific information, I/O device drivers, and other MQX context data.

3.4.1 Installing CodeWarrior TAD

TAD plug-in DLL is pre-installed in IAR Embedded Workbench automatically. If you need to update the plug-in to a new version included with the latest MQX installation, perform the following manual installation steps:

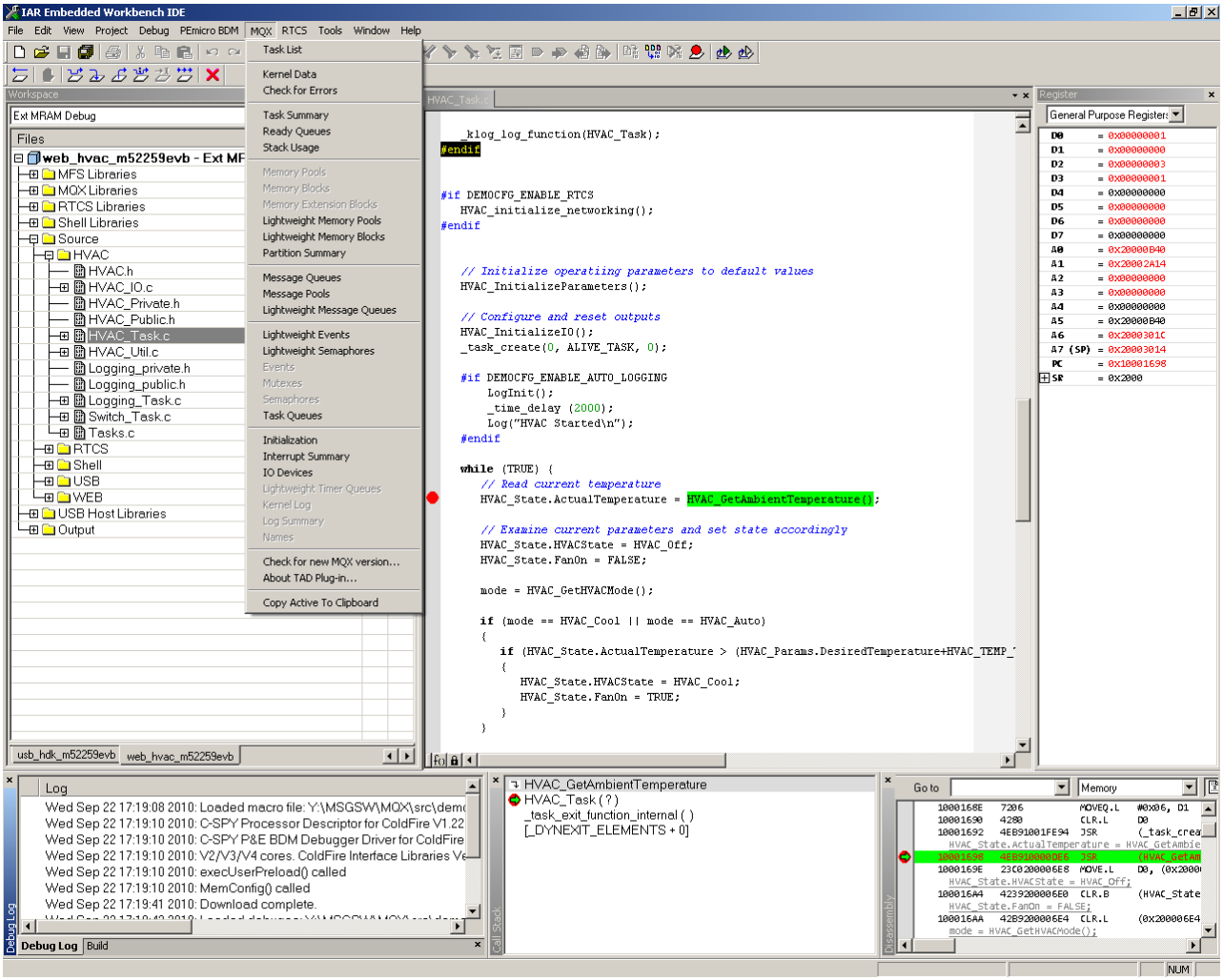
- Close the IAR Embedded Workbench IDE.
  - Locate the `tools\iar_extensions\<platform>` directory in Freescale MQX™ RTOS installation folder (by default `C:\Freescale\Freescale MQX x.y`).
  - Copy the entire content of `tools\iar_extensions\<platform>` directory to the IAR Embedded Workbench installation folder (e.g. `C:\Program Files\IAR Systems\Embedded Workbench 6.4\arm`)

- After the steps above are done, verify that the TAD plugin files exist at the new location:
  `<EW>\<platform>\plugins\rtos\MQX\MQXRtosPlugin.ewplugin`
  `<EW>\<platform>\plugins\rtos\MQX\MQXRtosPlugin<PLATFORM>.dll"`
- Re-start IAR Embedded Workbench IDE.
- In the Embedded Workbench environment, you should now be able to enable MQX TAD by selecting *"MQX"* in the *"Plugins"* tab of the "*Debugger"* panel of project settings. All example applications included with Freescale MQX™ RTOS are already configured this way.



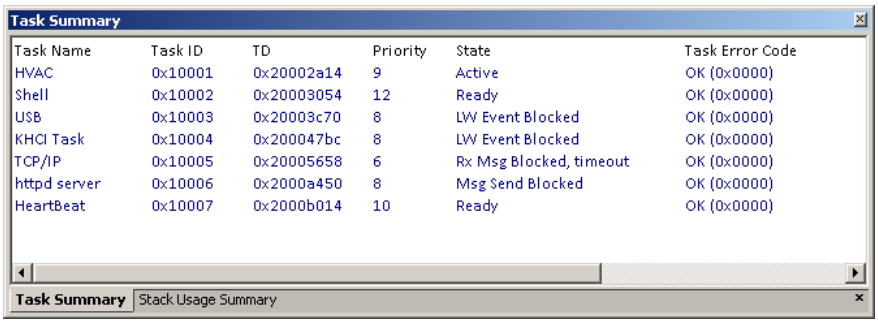
3.4.2 Using MQX TAD Screens

Using the MQX or RTCS menu in IAR IDE main window, several TAD “screens” may be opened during the debugging session.

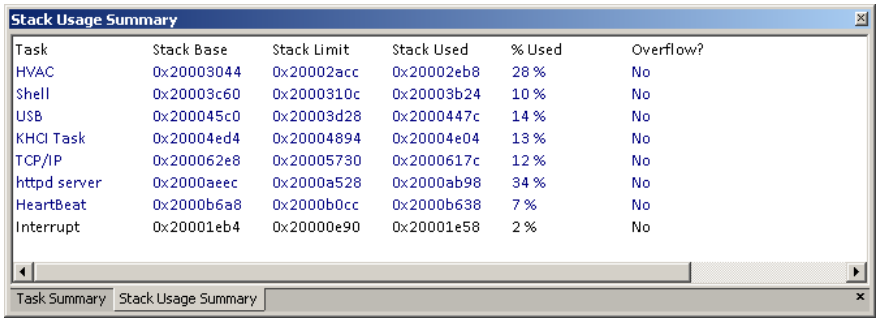


The most helpful and frequently used screens are shown in the images below:

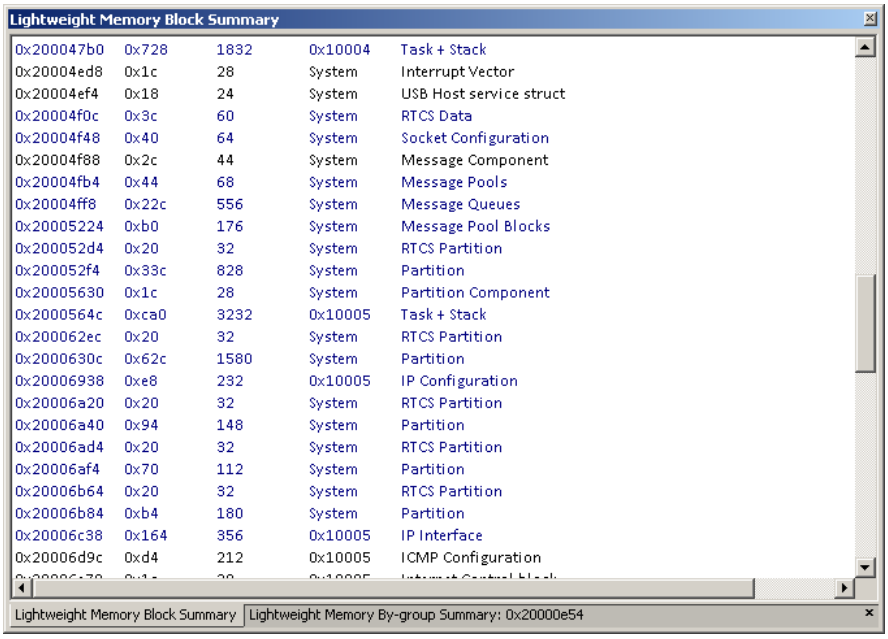
- *Task Summary* – overview about all tasks created in the MQX application.



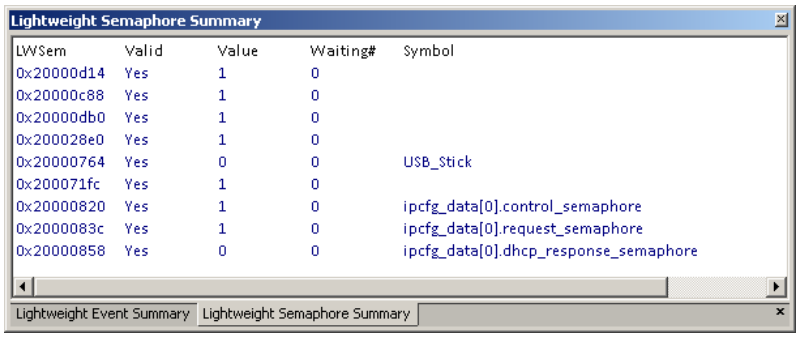
- *Stack Usage Summary* – displays information about interrupt and task stacks. Typically, a stack overflow is a root cause for vast majority of problems in MQX user applications.



- *Memory Block Summary* (or *Lightweight 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 by using the "*Memory Pools*" screen.



- *Semaphores, Events* (or *Lightweight Semaphores, Lightweight Events*) – displays address and status of synchronization objects created by the MQX system or application. When a synchronization object is allocated either as a global or static variable in the system, or as an array element or as a structure member allocated as global or static variable, the TAD plug-in also displays the symbolic name of the object.

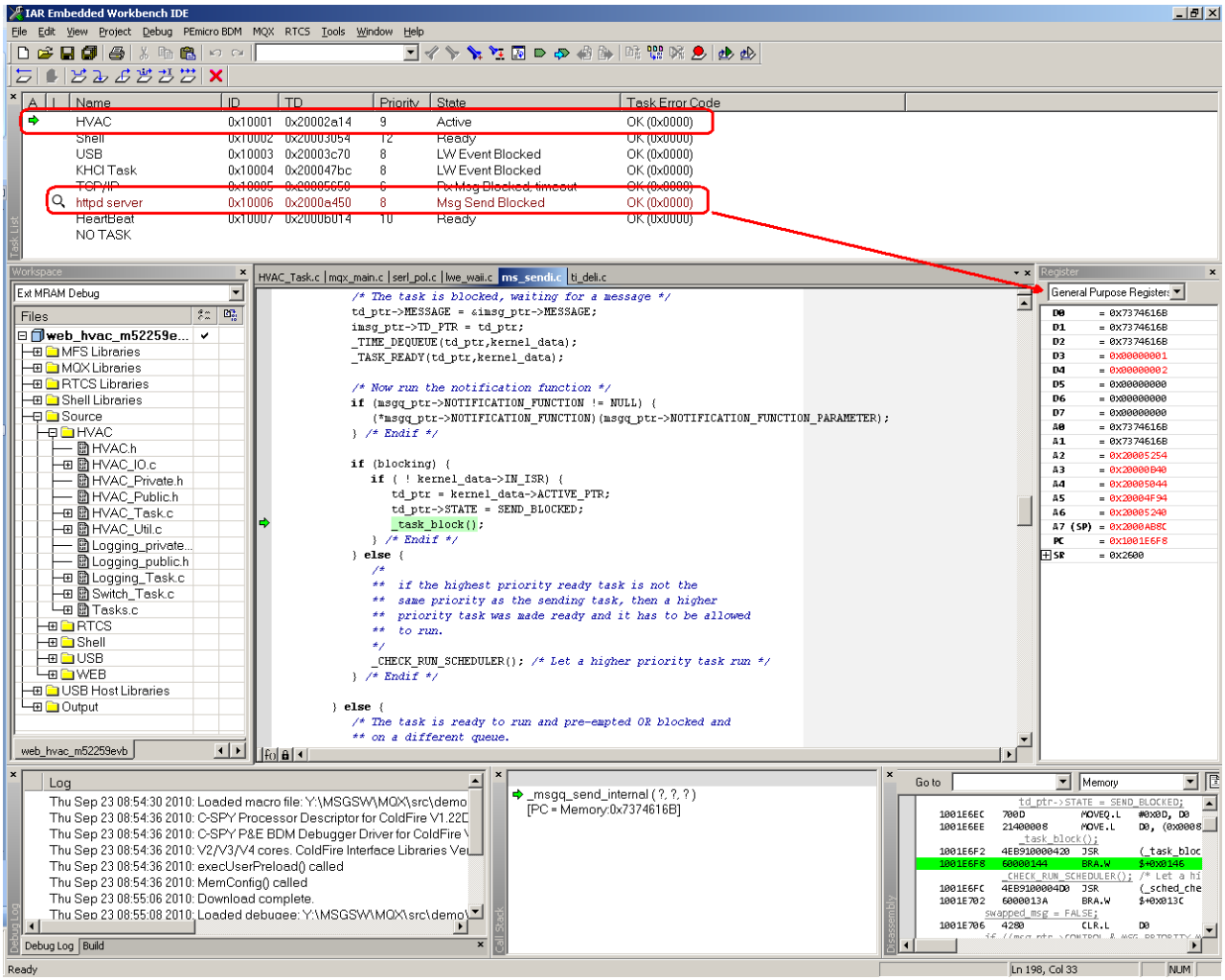


3.4.3 Task-aware Debugging

The TAD plug-in also provides native debugger support for multi-tasking MQX environment. Individual tasks can be examined any time the application stops on breakpoint or when it is stopped manually by pressing the “Break” red-hand toolbar button.

In the MQX menu, in the IAR IDE main window, select the “Task List” item at the top of the menu. The Task List view will open at the top of the window and will give you a list of all running tasks.

- The Green Arrow ➪ icon indicates which task was active at the moment of break.
- The Lens 🔍 icon indicates which task context is currently examined in the debugger in terms of execution point, register values, etc. Double click task items in the “Task List” view to move the lens and examine other tasks.

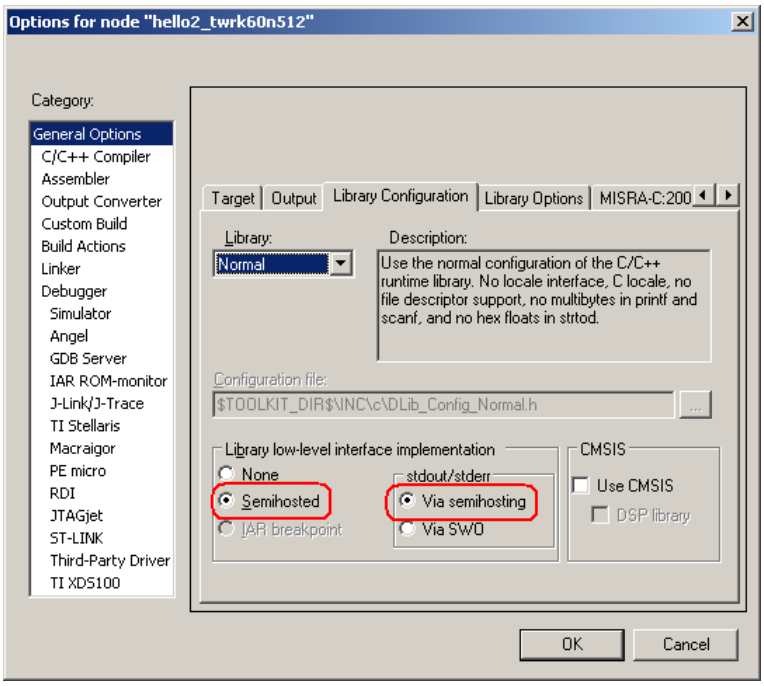
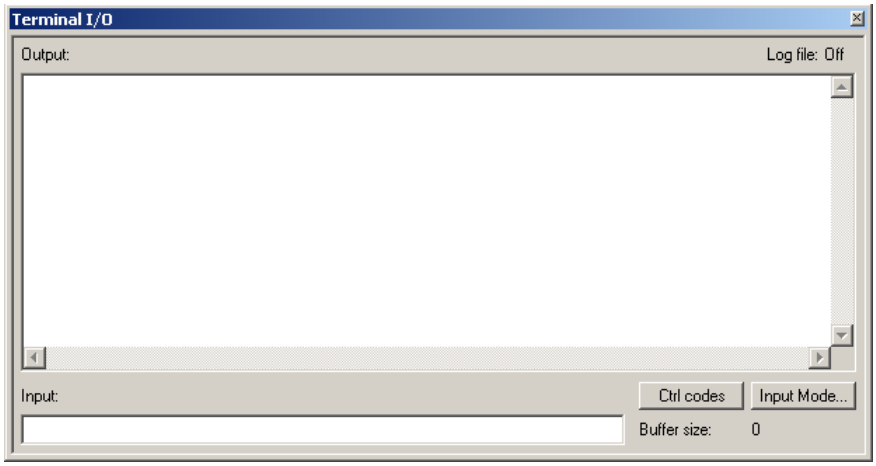


4 Using the MQX DebugIO Driver with EWARM IDE

The MQX provides the DebugIO driver allowing the processor to communicate with PC host computer via a debugger probe. The DebugIO channel can also be used as a default console for standard input and output operations. See more details about this driver in the “Getting Started with Freescale MQX™ RTOS” document.

The MQX RTOS currently supports ARM CortexM Semihost and ITM technologies. The IAR EWARM supports the Semihost communication channel for both input and output direction.

Change the “low-level interface implementation” settings in the project options, the *General Options* group *Library Configuration* tab, to enable debug console in the IDE:



The console can be opened during debugger session using the *View / Terminal I/O* menu in the EWARM IDE.