

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

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	4.2.0
DESCRIPTION:	Using IAR Embedded Workbench Tools with Freescale MQX™ RTOS
RELEASE DATE:	April, 2015

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, the Freescale logo, and CodeWarrior are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Vybrid and Tower are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM, ARM Powered logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.
© 2008-2015 Freescale Semiconductor, Inc.

Table of Contents

Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench®	i
1 Read Me First	2
2 Building the MQX RTOS Libraries and example application	3
2.1 Batch Build in IAR Embedded Workbench IDE	3
2.2 Build example/demo application project file	4
3 Running and Debugging the MQX RTOS application	5
3.1 Run MQX RTOS Hello World program	5
3.2 Multi-core debugging	6
3.3 Debugging (attach) the Application loaded by MQX RTOS Bootloader.....	11
3.4 MQX RTOS Task Aware Debugging	13
4 Using the MQX RTOS Debug/I/O Driver with EWARM IDE	19

1 Read Me First

This document describes the 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. See *Getting Started with Freescale MQX™ RTOS* and other user documentation included within the latest Freescale MQX RTOS installation for more details not specifically related to the IAR Embedded Workbench tools.

Get the latest Freescale MQX RTOS at freescale.com/mqx.

2 Building the MQX RTOS Libraries and example application

See Chapter 2 of the *Getting Started with Freescale MQX™ RTOS* (document MQXGSRTOS) for details on generic build process and compile time configuration. This document concentrates on steps specific to IAR tool chain only.

For MQX RTOS v4.2.0, each example/demo application comes with one workspace file, which contains the path to the example/demo project file and all dependent MQX RTOS library project files. Import the workspace file in the folder. For example, with MQX RTOS:

```
<mqx_install_dir>/mqx/examples/<example>/build/iar/<example>_<board_name>/<example>_<board_name>.eww
```

For Hello World example of MQX RTOS for FRDMK64F120M, the workspace file is located in folder:

```
<mqx_install_dir>/mqx/examples/hello/build/iar/hello_frdmk64f/hello_frdmk64f.eww
```

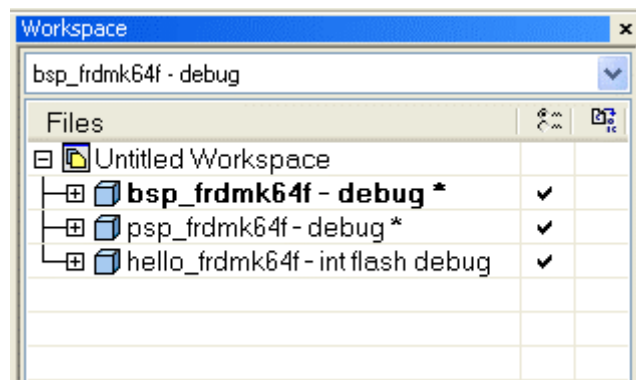


Figure 1- Import projects into IAR workspace

2.1 Batch Build in IAR Embedded Workbench IDE

The IAR IDE supports build of multiple project files in one go. Follow these steps to build all the MQX RTOS library project files within a workspace at once.

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

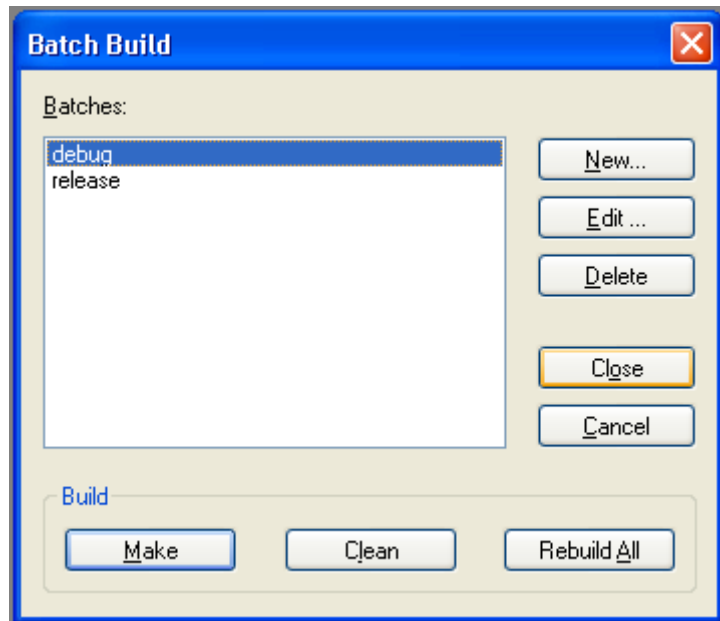


Figure 2- Batch build

2.2 Build example/demo application project file

- Select the build target and memory configuration for the example/demo application. With the Hello World example application of MQX RTOS for FRDMK64120M, it is **Int Flash Debug/Int Flash Release**

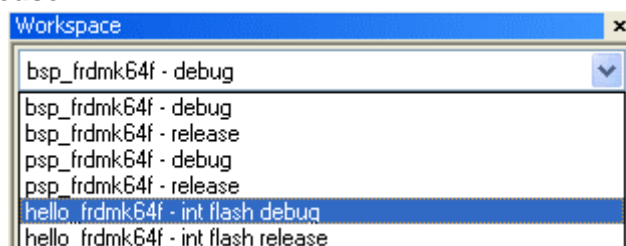


Figure 3- Build application

- Go to menu Project/Make, press the F7 key, or click the icon in the IAR IDE to build the application.

3 Running and Debugging the MQX RTOS application

The description bellow is provided for FRDMK64F120M BSP and Hello World example application. The same procedure applies for all other BSPs and examples.

3.1 Run MQX RTOS Hello World program

Loading and debugging MQX RTOS 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 the USB cable to the OpenJTAG connector on TWR-K60N512 board. Open the Terminal Window application by using Port USB COM, Baud 115200, Parity None, Bits 8.
- In project/Options-Debugger check debugger Driver settings. Default is J-Link. When a MQX RTOS application is compiled and linked to all MQX RTOS 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.

```
mqx_main.c
int main
(
    void
)
{ /* Body */

    extern const MQX_INITIALIZATION_STRUCT MQX_init_struct;

    /* Start MQX */
    mqx( (MQX_INITIALIZATION_STRUCT_PTR) &MQX_init_struct );
    return 0;
} /* Endbody */
```

Figure 4- mqx_main.c

- Press the “Run” button to start the application.



Figure 5- Run button

- Hello World is printed on serial console terminal.

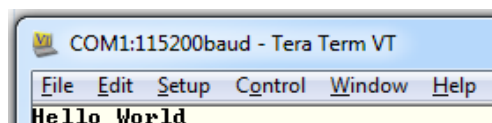


Figure 6 – Console terminal

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 ARM[®]Cortex[®]-A5 is the primary core while the ARM[®]Cortex[®]-M4 is set up as an auxiliary core.

3.2.1 Debugging with J-Link

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

- Open two instances of IAR IDE. In the first IDE instance, open the Cortex-A5 example application. In the second IDE instance, open the application for Cortex-M4 core.

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

And

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

- Ensure that both MQX RTOS library and MCC (<mqx_install_dir>/mcc/build) library are compiled for each core prior to the compilation of the example application. Set the following parameters before debugging:
- For primary core (Cortex-A5), set in project *Options*:

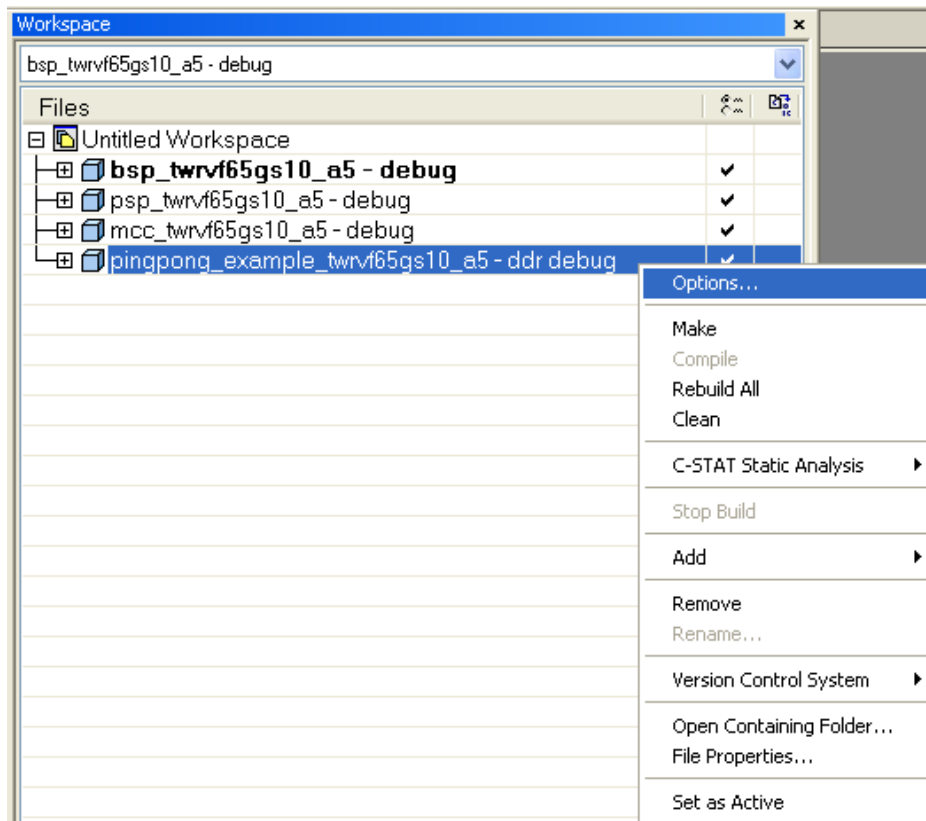


Figure 7- Options

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

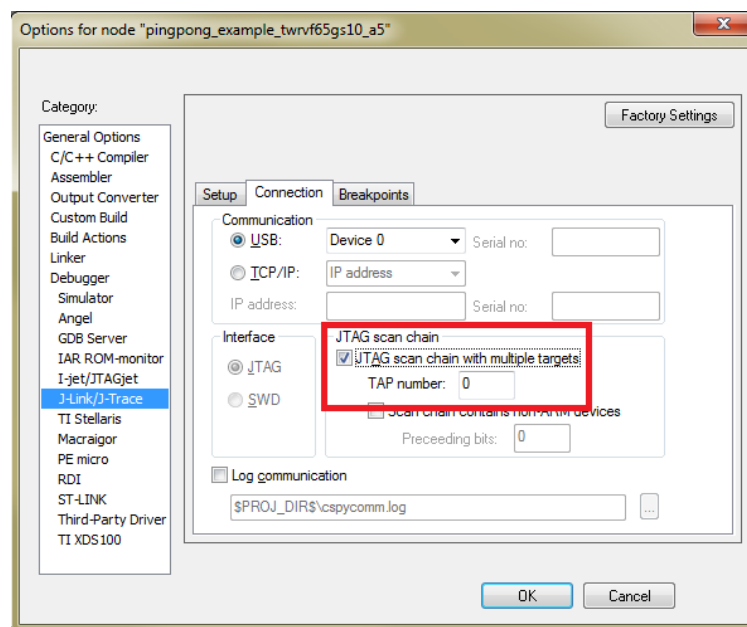


Figure 8-Options for node

- For secondary core, select “JTAG scan chain with multiple targets” and TAP number “3” in the J-Link/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.

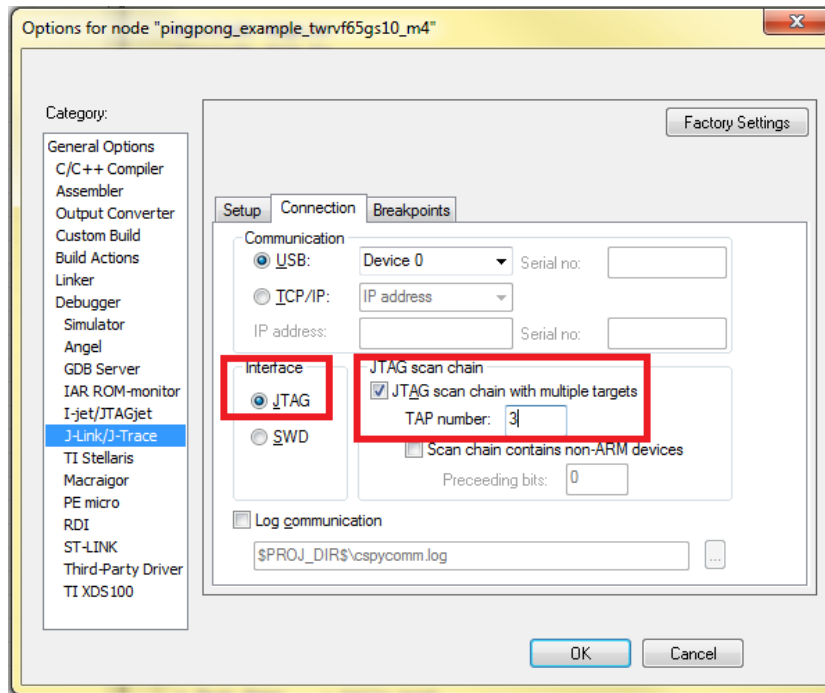


Figure 9- Options for node

- Set the Reset type to “Core.”

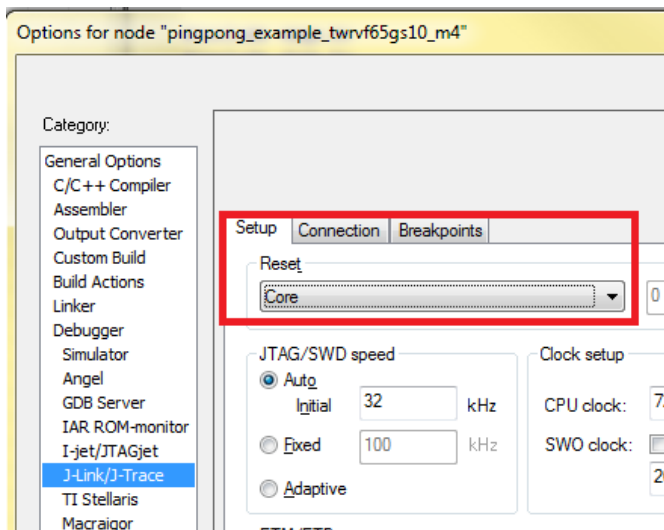


Figure 10- Options for node - setup

- Start the primary core (Cortex-A5) application. You should see the following message on the terminal:

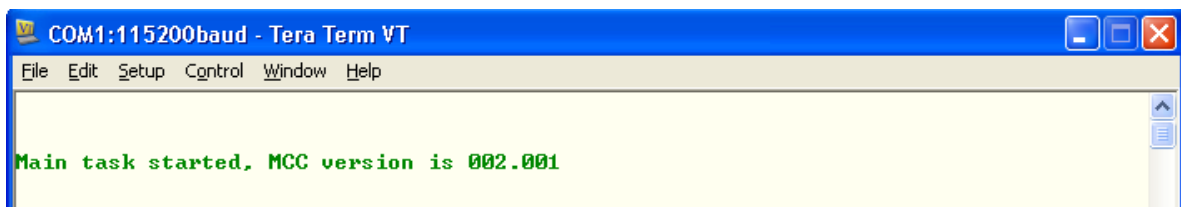


Figure 11- Terminal

- Start the auxiliary core (Cortex-M4) application. The responder starts and the message “pingpong” starts between the cores.

```
COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

Main task started, MCC version is 002.001

Responder task started, MCC version is 002.001
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 1
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 2
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 3
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 4
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 5
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 6
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 7
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 8
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 9
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = a
```

Figure 12- Terminal

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 below:

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

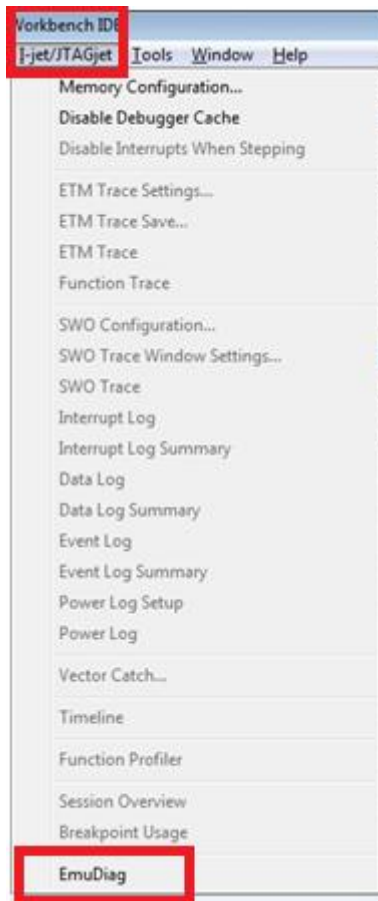


Figure 13- I-Jet

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

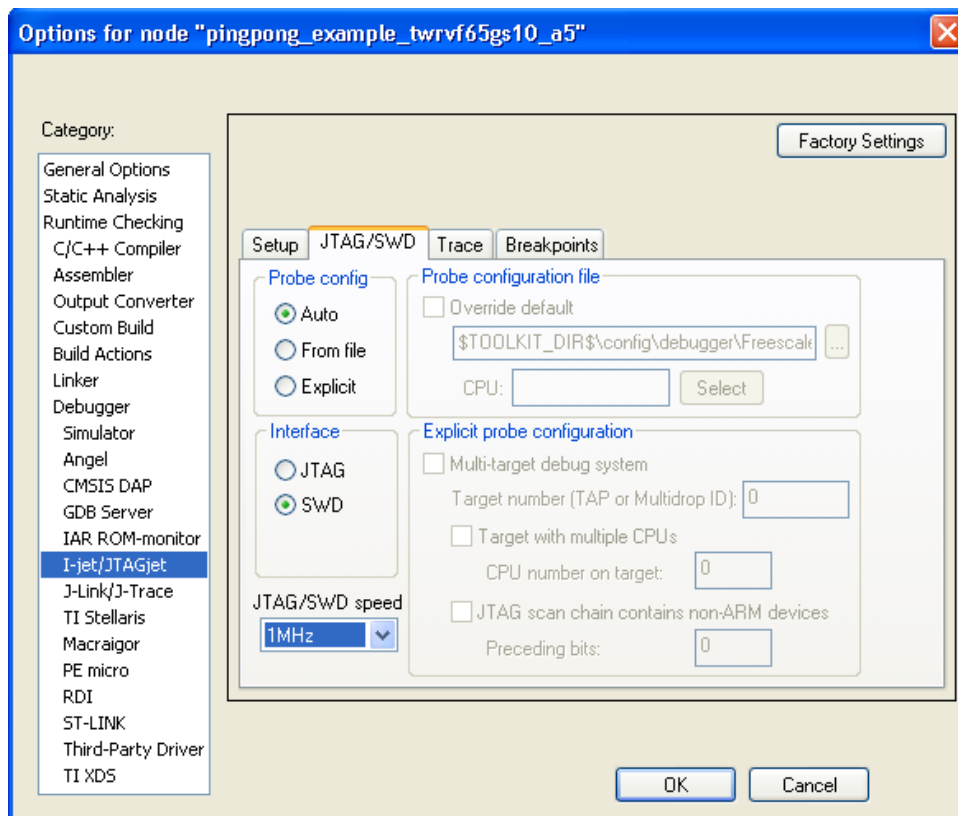


Figure 14- Factory settings

- Change the interface from JTAG to SWD

3.3 Debugging (attach) the Application loaded by MQX RTOS Bootloader

This chapter describes debugging of the application loaded to the processor memory by MQX RTOS Bootloader. The similar approach can be used for debugging an application loaded by a different bootloader 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 Bootloader usage, see Readme.txt located in the MQX RTOS Bootloader application folder.

(`<mqx_install_dir>/mqx/examples/bootloader_vybrid/Readme.txt`)

Building Bootloader and creating bootable SD card

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

`<mqx_install_dir>/mqx/examples/bootloader_vybrid/build/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 RTOS Bootloader prints out the following message on default console (RS232 TWR-SER) and start execution of M4 and A5 applications. The pingpong example image files are copied to the root directory on the bootable SD card.

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
Bootloader from A5 as primary core, dual core
=====
Mounting filesystem
Starting bootloader

Responder task started, MCC version is 002.001

Main task started, MCC version is 002.001
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 1
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 2
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 3
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 4
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 5
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 6
Responder task received a msg from [0,0,1] endpoint
Message: Size=4, DATA = 7
Main task received a msg from [1,0,2] endpoint
Message: Size=4, DATA = 8
  
```

Figure 15 – Terminal

- To debug the running application, go to project properties and setup **J-Link/J-Trace** settings as described in the previous chapter.
 - Go to **Debugger** menu and uncheck **Run to main** option.

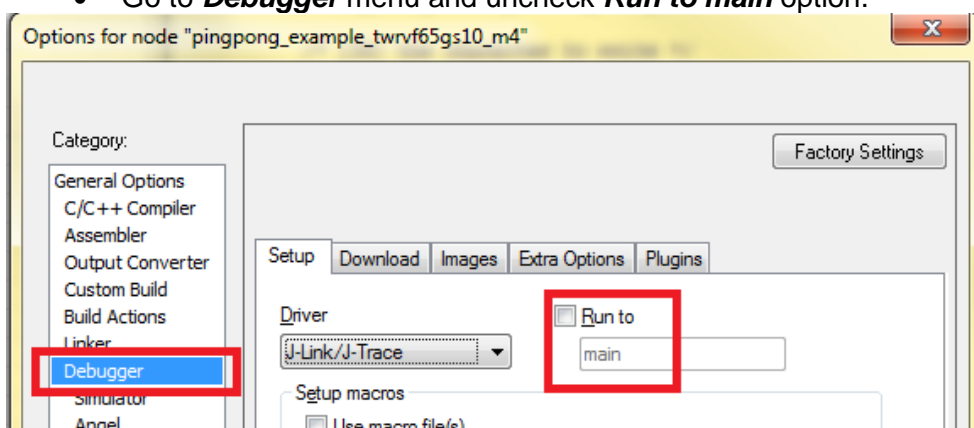


Figure 16- Debugger menu

- Select Attach to running target.

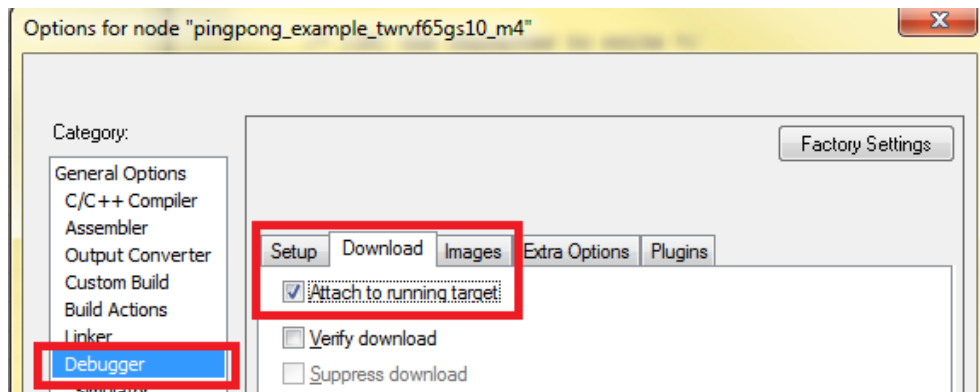



Figure 17- Download

- Use the Debug without downloading button . The debugger connects to the selected application. Stop the selected core and debug the booted image.

3.4 MQX RTOS Task Aware Debugging

MQX RTOS 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 RTOS data structures, task-specific information, I/O device drivers, and other MQX RTOS context data.

3.4.1 Installing IAR Embedded Workbench TAD

TAD plug-in DLL is pre-installed in IAR Embedded Workbench automatically. To update the plug-in to a new version included with the latest MQX RTOS 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 RTOS TAD by selecting "MQX RTOS" in the "Plugins" tab of the "Debugger" panel of project settings. All example applications included with Freescale MQX RTOS are already configured this way.

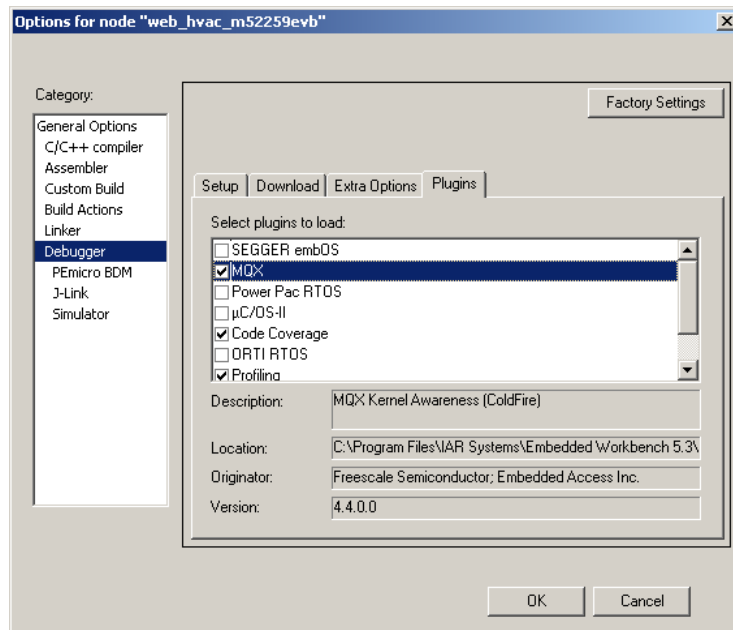


Figure 18-Plugins

3.4.2 Using MQX RTOS TAD Screens

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

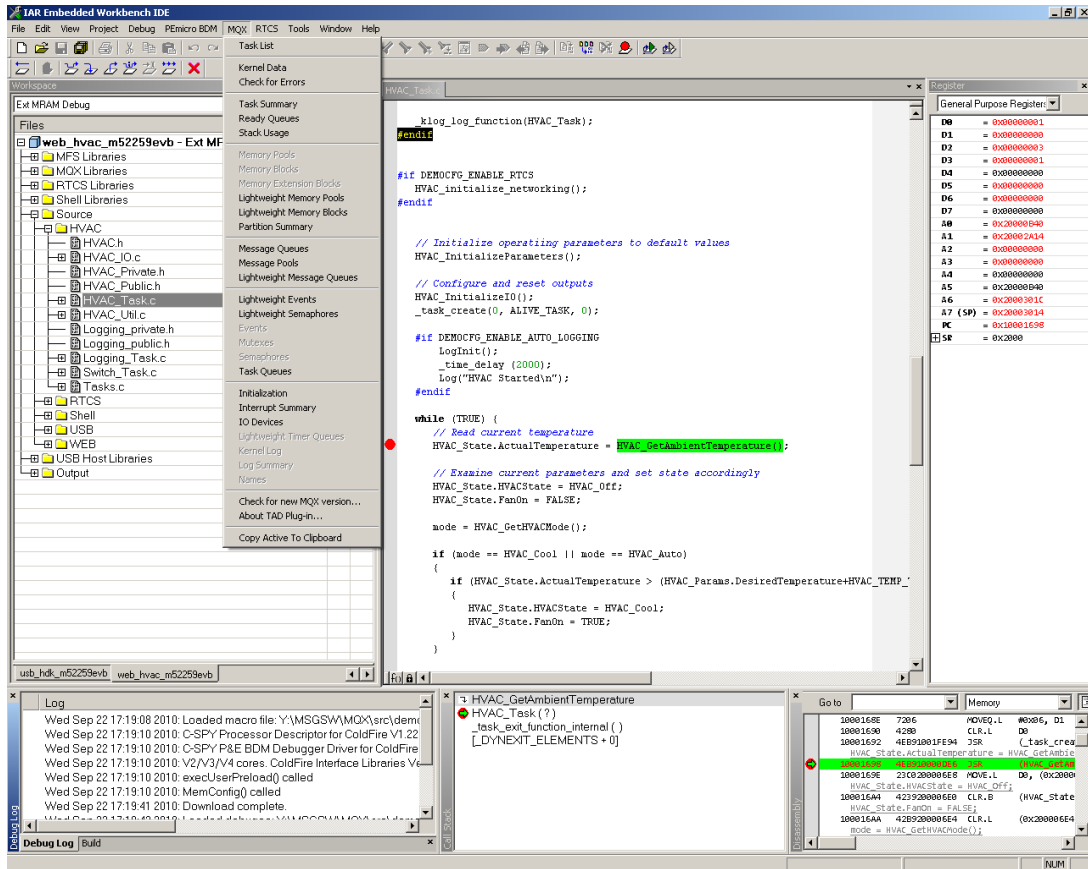


Figure 19- MQX RTOS TAD screens

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

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

Task Name	Task ID	TD	Priority	State	Task Error Code
HVAC	0x10001	0x20002a14	9	Active	OK (0x0000)
Shell	0x10002	0x20003054	12	Ready	OK (0x0000)
USB	0x10003	0x20003c70	8	LW Event Blocked	OK (0x0000)
KHCI Task	0x10004	0x200047bc	8	LW Event Blocked	OK (0x0000)
TCP/IP	0x10005	0x20005658	6	Rx Msg Blocked, timeout	OK (0x0000)
httpd server	0x10006	0x2000a450	8	Msg Send Blocked	OK (0x0000)
HeartBeat	0x10007	0x2000b014	10	Ready	OK (0x0000)

Figure 20- Task summary

- **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 RTOS user applications.

Task	Stack Base	Stack Limit	Stack Used	% Used	Overflow?
HVAC	0x20003044	0x20002acc	0x20002eb8	28 %	No
Shell	0x20003c60	0x2000310c	0x20003b24	10 %	No
USB	0x200045c0	0x20003d28	0x2000447c	14 %	No
KHCI Task	0x20004ed4	0x20004894	0x20004e04	13 %	No
TCP/IP	0x200062e8	0x20005730	0x2000617c	12 %	No
httpd server	0x2000aeec	0x2000a528	0x2000ab98	34 %	No
HeartBeat	0x2000b6a8	0x2000b0cc	0x2000b638	7 %	No
Interrupt	0x20001eb4	0x20000e90	0x20001e58	2 %	No

Figure 21 – Stack usage summary

- **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 RTOS system or applications. Additional memory pools (if used) may be displayed by using the “Memory Pools” screen.

Address	Size	Offset	Type	Name
0x200047b0	0x728	1832	0x10004	Task + Stack
0x20004ed8	0x1c	28	System	Interrupt Vector
0x20004ef4	0x18	24	System	USB Host service struct
0x20004f0c	0x3c	60	System	RTCS Data
0x20004f48	0x40	64	System	Socket Configuration
0x20004f88	0x2c	44	System	Message Component
0x20004fb4	0x44	68	System	Message Pools
0x20004ff8	0x22c	556	System	Message Queues
0x20005224	0xb0	176	System	Message Pool Blocks
0x200052d4	0x20	32	System	RTCS Partition
0x200052f4	0x33c	828	System	Partition
0x20005630	0x1c	28	System	Partition Component
0x2000564c	0xca0	3232	0x10005	Task + Stack
0x200062ec	0x20	32	System	RTCS Partition
0x2000630c	0x62c	1580	System	Partition
0x20006938	0xe8	232	0x10005	IP Configuration
0x20006a20	0x20	32	System	RTCS Partition
0x20006a40	0x94	148	System	Partition
0x20006ad4	0x20	32	System	RTCS Partition
0x20006af4	0x70	112	System	Partition
0x20006b64	0x20	32	System	RTCS Partition
0x20006b84	0xb4	180	System	Partition
0x20006c38	0x164	356	0x10005	IP Interface
0x20006d9c	0xd4	212	0x10005	ICMP Configuration

Figure 22- Memory block summary

- **Semaphores, Events (or Lightweight Semaphores, Lightweight Events)** – displays address and status of synchronization objects created by the MQX RTOS 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.



LWSem	Valid	Value	Waiting#	Symbol
0x20000d14	Yes	1	0	
0x20000c88	Yes	1	0	
0x20000db0	Yes	1	0	
0x200028e0	Yes	1	0	
0x20000764	Yes	0	0	USB_Stick
0x200071fc	Yes	1	0	
0x20000820	Yes	1	0	ipcfg_data[0].control_semaphore
0x2000083c	Yes	1	0	ipcfg_data[0].request_semaphore
0x20000858	Yes	0	0	ipcfg_data[0].dhcp_response_semaphore

Figure 23- Lightweight semaphore summary

3.4.3 Task-aware Debugging

The TAD plug-in also provides native debugger support for multi-tasking MQX RTOS 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 RTOS menu, in the IAR IDE main window, select the “Task List” item at the top of the menu. The Task List view opens at the top of the window and gives 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.

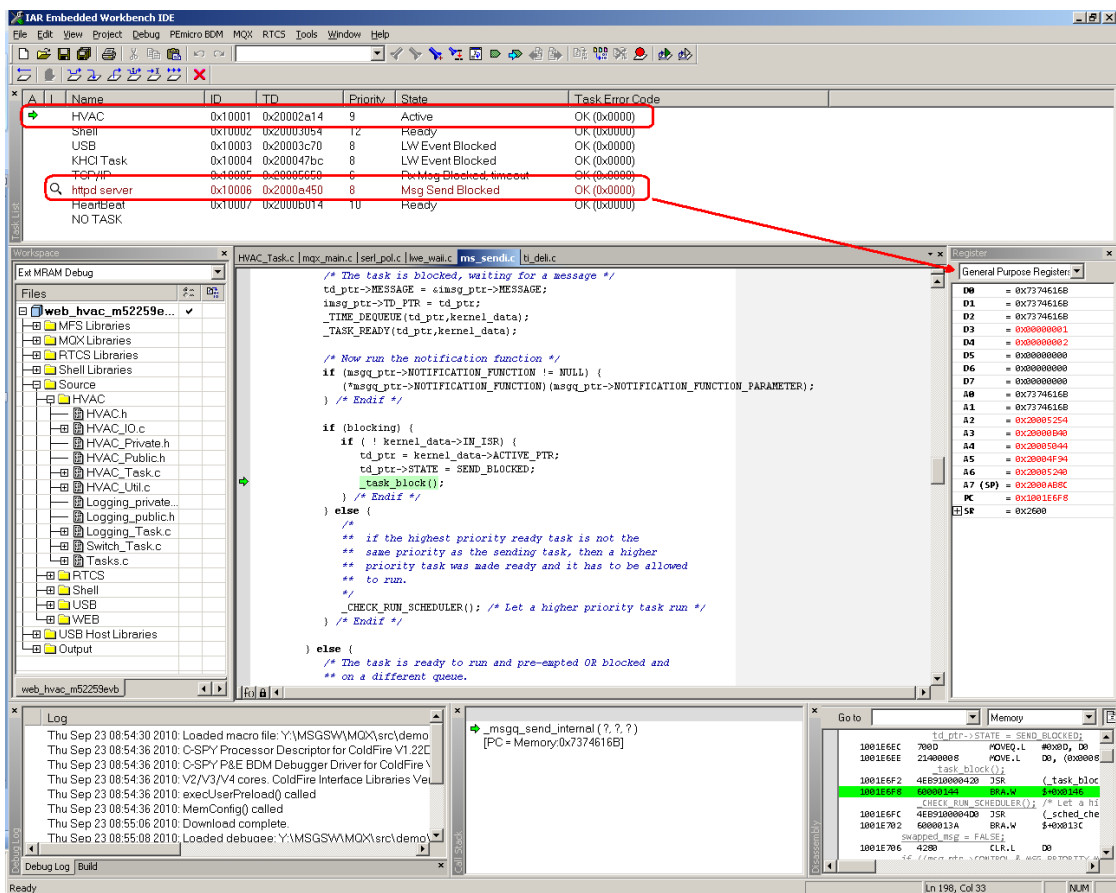


Figure 24- IAR

4 Using the MQX RTOS Debug/I/O Driver with EWARM IDE

The MQX RTOS 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® Cortex®-M 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:

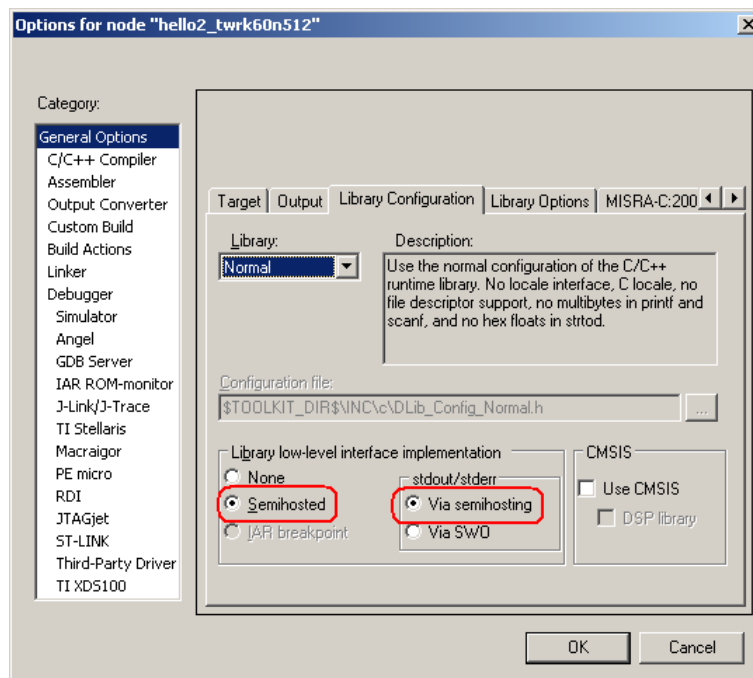


Figure 25- Library configuration

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

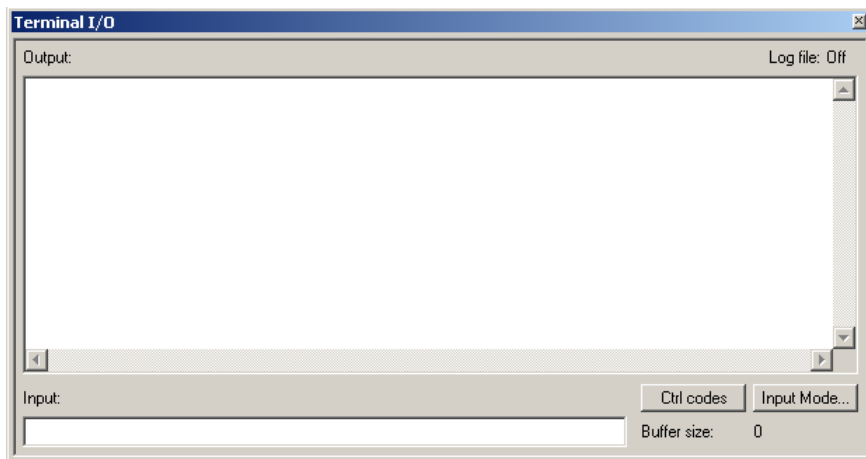


Figure 26- Terminal I/O