Document Number: MQXGSDS5 Rev. 04, 8/2013

Getting Started with ARM[®] Development Studio 5 (DS-5™) with Freescale MQX™ RTOS

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	Freescale MQX 4.0.2
DESCRIPTION:	Using ARM [®] Development Studio 5 (DS-5™) with Freescale MQX™ RTOS
RELEASE DATE:	August, 2013



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. Vybrid and Tower are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM is the registered trademark of ARM Limited. ARM Development Studio 5 (DS-5), Cortex-M4, and Cortex-A5 are trademarks of ARM Limited.

© 2008-2013 Freescale Semiconductor, Inc.

Document Number: MQXGSDS5

Rev. 04, 08/2013





Table of Contents

Getting Started with ARM [®] Development Studio 5 (DS-5™) with Freescale MQX™ RTOS	j
1 Read Me First	2
2 Building the MQX Libraries	3
2.1 Vybrid-kit special licenses	5
3 Running and Debugging MQX application	6
3.1 Debugging Primary Core - MQX Hello World program 3.2 Run MQX Hello World program on auxiliary core on dual core system (Vybrid ARM® Cortex M core)	.®_
3.3 Multi-core debugging	.11
4 MOX Task Aware Debugging in Development Studio 5 (DS-5) IDF	16

1 Read Me First

This document describes how to build, debug, and run Freescale MQX™ RTOS programs in the ARM[®] Development Studio 5 (DS-5™) development suite.

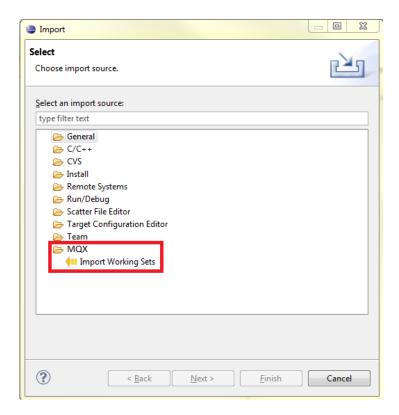
See *Getting Started with Freescale MQX*TM *RTOS* and other user documentation included within the latest Freescale MQX RTOS distribution for further information on board specific build targets, jumper and HW settings, MQX API documentation, etc.

2 Building the MQX Libraries

This chapter concentrates on steps specific to Development Studio 5 (DS-5) tool chain only. For details on generic build process and compile time configuration, see Chapter 2 of the *Getting Started with Freescale MQX™ RTOS*.

First, install the MQX Eclipse plugin using *Help\Install New Software\Add\Archive...* menu. Then, select the following archive: <mqx install dir>/tools/ds5/ds5 update site.zip

To rebuild the MQX libraries, import the <mqx_install_dir>/config/<board>/ds5/<board>.wsd working set description file using *FileVmportWQXVmport Working Sets* menu. The MQX library projects will be imported to DS-5 working space together with build configurations settings.



The following projects will be imported to your workspace

```
<mqx_install_dir>/mqx/build/ds5/bsp_<board>/.project
<mqx_install_dir>/mqx/build/ds5/psp_<board>/.project
<mqx_install_dir>/mfs/build/ds5/mfs_<board>/.project
<mqx_install_dir>/rtcs/build/ds5/rtcs_<board>/.project
<mqx_install_dir>/usb/host/build/ds5/usbh_<board>/.project
<mqx_install_dir>/usb/device/build/ds5/usbd_<board>/.project
<mqx_install_dir>/usb/device/build/ds5/shell_<board>/.project
```

Getting Started with Freescale MQX™ RTOS and ARM® Development Studio 5 (DS-5™) Rev. 04

• Select the target and platform and build the libraries - hit the compile all button twnvf65gs10_m4 | Dt
. All projects will be built in the selected configuration. The "Debug" configuration is dedicated for easy application debugging while the "Release" target has compiler and linker optimization set to maximum.



2.1 Vybrid-kit special licenses

In addition to the regular DS5 licenses there are two special Vybrid licenses available.

- Vybrid edition license (ds.arm.com/vybrid/vybrid-edition/)
- Vybrid-tower-starter-kit license (ds.arm.com/vybrid/vybrid-tower-starter-kit/)

If you are using any of these licenses please be aware of the following limitations:

- The licenses are "node locked" and the tool cannot be accessed with the Windows Remote Desktop.
- Code size limitation (1 KB to Vybrid-edition, 256 MB for Vybrid-tower-starter-kit)
- Vectorization this feature is not supported in older DS5.14 with the special Vybrid license.
 Users have to turn off this feature manually (uncheck checkbox in IDE or use "--vectorize" option). This results in a less optimized code. The DS5.15 users are not affected.
- To use the command line build tools (with make), an additional compiler option is needed and the scatter file needs to be updated. Add the following options to the command line and update the scatter file by adding the same option to the end of the first line.
- Vybrid edition: "--tool_variant=vf6xx_tk"
- Vybrid-tower-starter-kit: "--tool_variant=vf6xx_sk"

3 Running and Debugging MQX application

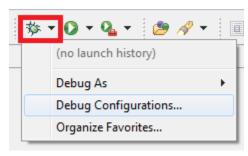
The description bellow is provided for Vybrid microcontrollers BSPs - twrvf65gs10_a5 and twrvf65gs10_m4 and Hello World example application. The twrvf65gs10_a5 BSP runs on primary and twrvf65gs10_m4 runs auxiliary Vybrid core. The same procedure applies for all other BSPs and example applications distributed in the MQX release package.

3.1 Debugging Primary Core - MQX Hello World program

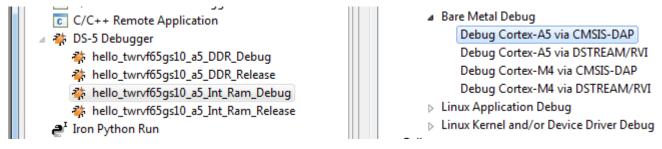
- Connect a serial cable to the TWR-SER or TWR-SER2 board DB9 connector. Set the communication speed to 115200.
- Select menu *File/Import/General/Existing Projects into Workspace* and import Hello World example application.

<mqx_install_dir>/mqx/examples/hello/ds5/hello_twrvf65gs10_a5/.project

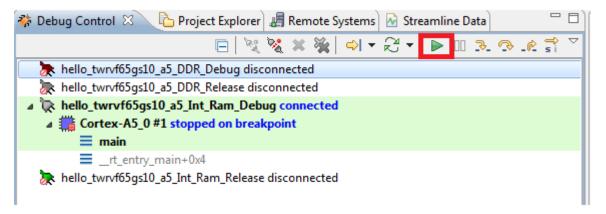
- Hit the compile button strength
 to build application Int. RAM Debug target.
- Click the arrow next to the Debug button and select Debug Configurations.



A dialog box will come up. Select the hello_twrvf65gs10_a5_Int_Ram_Debug
configuration in the Vybrid ARM® Cortex®-A5 CMSIS-DAP debug connection. Then hit the
Debug button in the lower right corner.



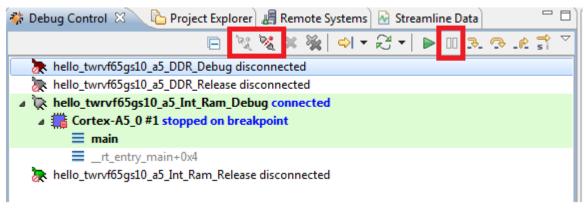
• The Development Studio 5 (DS-5) will switch to Debug Perspective automatically. Then, the project will be loaded to the device and execution will stop in the main() function.



- Press the Run button to continue in the Hello World program execution.
- The program will print Hello World on serial console terminal.



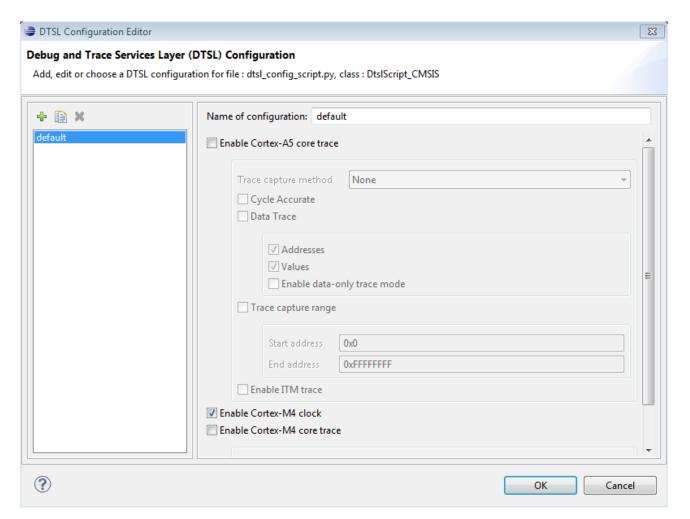
• To debug the application again, push *Interrupt* button to stop program execution. Then press *Disconnect from Target* and *Connect from Target* buttons in *Debug Control* Menu.



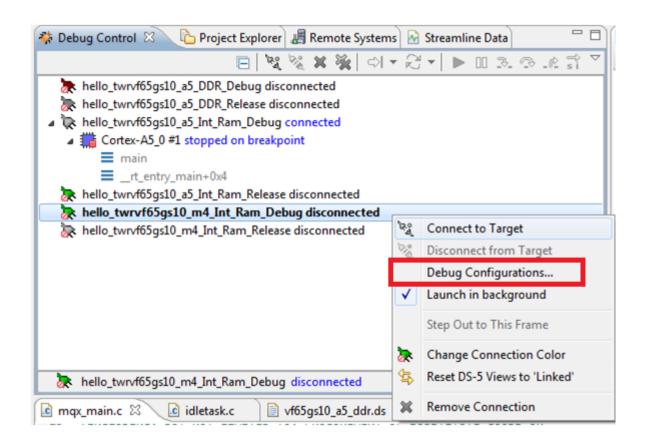
Note: If subsequent connections to the target fail, it is recommended to reset the board by using the Reset button on TWR or by using PWR down/up sequence on the TWR elevator.

3.2 Run MQX Hello World program on auxiliary core on dual core system (Vybrid ARM® Cortex®-M core)

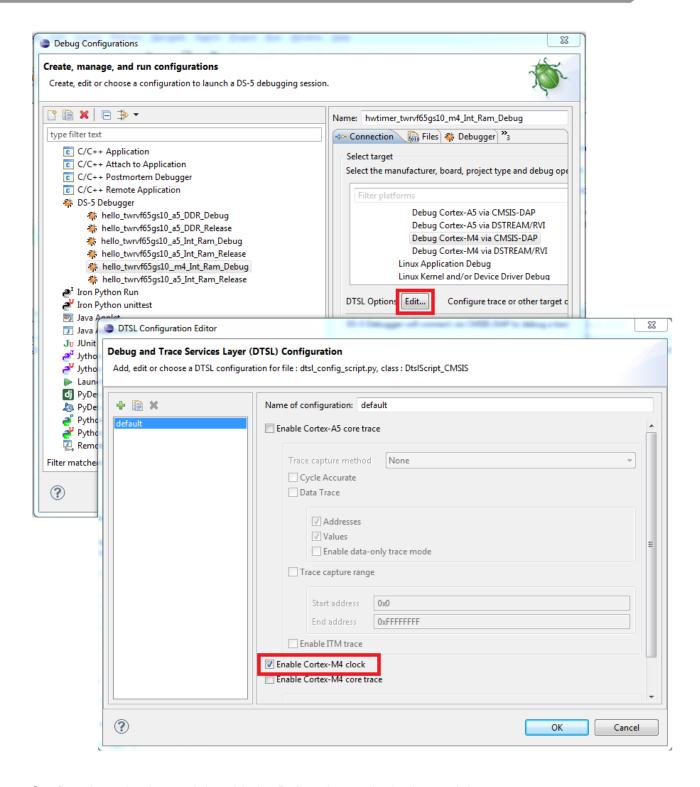
Before loading the application to the primary core as described in the previous chapter, it is
important to check the Enable Cortex-M4 clock in the DTSL Options menu (accessible
from the Debug Configuration dialogue).



- Run the primary core application and stop the program execution using the *Interrupt* button
- Switch to C/C++ Perspective using the button in right top corner
- Select menu *File/Import/General/Existing Projects into Workspace* and import Hello World example application for Cortex-M4 auxiliary core to your workspace:
- <mqx install dir>/mqx/examples/hello/ds5/hello twrvf65gs10 m4/.project
- Hit the compile button and build application in the selected target. By default, the project is compiled in the *Int Ram Debug* target.
- Highlight the hello_twrvf65gs10_m4_Int_Ram_Debug target.



 Select Debug Configurations and verify that the Enable Cortex-M4 clock option in the DTSL Options menu is still checked.



- Confirm the selection and then hit the **Debug** button in the lower right corner.
- Project will be loaded to device and execution. CorexM4 core execution will stop in the main() function
- Press the run button and the program running on the auxiliary Cortex-M4 core will print "Hello World" on the serial console.



Note: It is always necessary to execute MQX application on primary core first. The MQX primary core application startup sequence contains settings required by Cortex-M4 core (clock setup etc).

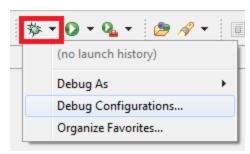
3.3 Multi-core debugging

This chapter describes the basics of multi-core debugging with MQX RTOS. Description is provided for Vybrid microcontroller BSPs - twrvf65gs10_a5/twrvf65gs10_m4 and Multicore Communication (MCC) "pingpong" example application. The Cortex-A5 is primary core in this setup while the Cortex-M4 is set to auxiliary core.

- Select menu *File/Import/General/Existing Projects into Workspace* and import Cortex-A5 and Cortex-M4 MCC library projects as follows:
 - <mqx install dir>/mcc/build/ds5/mcc twrvf65gs10 a5/.project
 - <mqx install dir>/mcc/build/ds5/mcc twrvf65gs10 m4/.project
- Select the *mcc_twrvf65gs10_a5* project in the Project Explorer View and then hit the compile button to build the *Debug target*.
- Select the *mcc_twrvf65gs10_m4* project in the Project Explorer View and then hit the compile button to build the *Debug target*.
- Select menu File/Import/General/Existing Projects into Workspace and import MCC Pingpong example applications for both Cortex-A5 and Cortex-M4 core as follows:

<mqx_install_dir>/mcc/examples/pingpong/ds5/pingpong_example_twrvf65gs10_a5/.project
<mqx install dir>/mcc/examples/pingpong/ds5/pingpong example twrvf65gs10 m4/.project

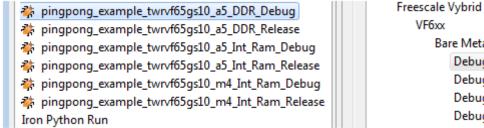
- Select the *pingpong_example_twrvf65gs10_a5* project in the Project Explorer View and then hit the compile button to build the *DDR Debug target*.
- Select the *pingpong_example_twrvf65gs10_m4* project in the Project Explorer View and then hit the compile button to build the *Int Ram Debug target*.
- Click the arrow next to the Debug button and select **Debug Configurations**:



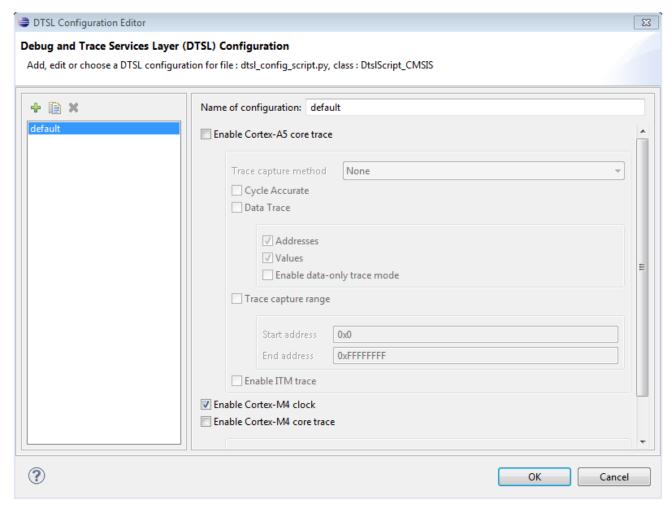
• A dialog box will come up. Select the *pingpong_example_twrvf65gs10_a5_DDR_Debug* configuration and the *Vybrid Cortex-A5 CMSIS-DAP* debug connection. Before hitting the

Getting Started with Freescale MQX™ RTOS and ARM® Development Studio 5 (DS-5™) Rev. 04

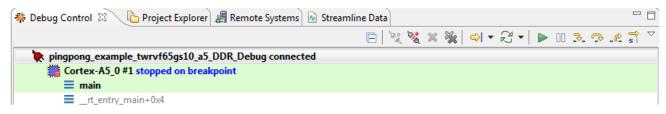
Debug button in the lower right corner, check the **Enable Cortex-M4 clock** in the **DTSL Options** menu.



Freescale Vybrid
VF6xx
Bare Metal Debug
Debug Cortex-A5 via CMSIS-DAP
Debug Cortex-A5 via DSTREAM/RVI
Debug Cortex-M4 via CMSIS-DAP
Debug Cortex-M4 via DSTREAM/RVI

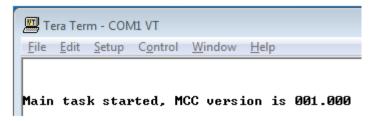


- The Development Studio 5 (DS-5) will switch to Debug Perspective automatically. Then, the project will be loaded to the device and execution will stop in the main() function.
- Press the **Run** button to continue in the *pingpong_example_twrvf65gs10_a5* program execution.



Getting Started with Freescale MQX™ RTOS and ARM DS-5 Rev. 04

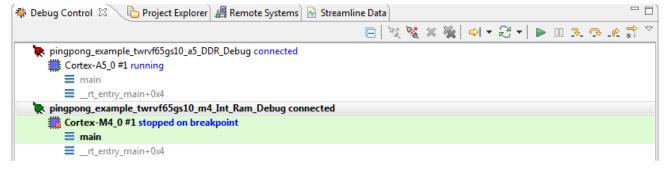
• The program will print "Main task started, MCC version is xxx.xxx" on the serial console terminal.



- Once the Cortex-A5 code application is running, start the execution of the auxiliary core (Cortex-M4).
- Select menu Run/Debug Configurations.
- When the Debug Configuration dialogue occurs, select the pingpong_example_twrvf65gs10_m4_Int_Ram_Debug configuration and the Vybrid Cortex-M4 CMSIS-DAP debug connection.
- Then hit the **Debug** button in the lower right corner.



The Cortex-M4 project will be loaded to the device and execution will stop in the main () function. Press the *Run* button to continue in the *pingpong_example_twrvf65gs10_m4* program execution.



• The responder will be started and message "pingpong" between the cores will be initialized. See the console log.

Getting Started with Freescale MQX™ RTOS and ARM® Development Studio 5 (DS-5™) Rev. 04

```
Tera Term - COM1 VT
 File Edit Setup Control Window
                                           Help
Main task started, MCC version is 001.000
Responder task started, MCC version is 001.000
Responder task received a msg
Message: Size=4, DATA = 1
Main task received a msg
Message: Size=4, DATA = 2
Responder task received a
Message: Size=4, DATA = 3
Main task received a msg
Message: Size=4, DATA = 4
Responder task received a msg
Message: Size=4, DATA = 5
```

3.4 Debugging the Application loaded by MQX Boot Loader

This chapter describes debugging the application which was 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 steps required for preparing bootable SD Card image and application images in DS-5 tool set. For details about the 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

- First, import the MQX Boot Loader project to your workspace by using the File/Import/General/Existing Projects into Workspace menu.
- Select the bootloader_vybrid from your MQX installation directory:

<mgx install dir>/mgx/examples/bootloader vybrid/ds5/bootloader vybrid twrvf65gs1 0 a5

Select Int Ram Debug target and hit the compile button



Use DS5 "C:\Program Files\DS-5\bin\fromelf" utility to create the binary image:

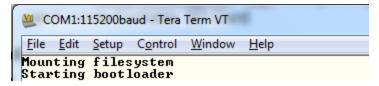
fromelf.exe --bin --output=bootloader vybrid twrvf65qs10 a5.bin bootloader vybrid twrvf65gs10 a5.axf

Follow <mgx install dir>\mgx\examples\bootloader vybrid\Readme.txt description and use prepare binary image to prepare the bootable SD Card.

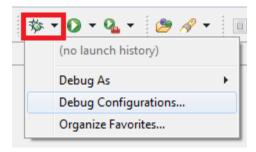
Building and Debugging the Application images

- Build the applications you want to run on A5 and M4 cores and convert them to binary format (.bin) by using frome1f DS-5 utility.
- Store the binary images on the 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 the default console (RS232 TWR-SER) and start execution of M4 and A5 applications.

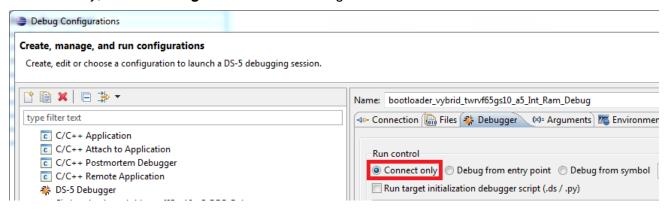
Getting Started with Freescale MQX™ RTOS and ARM DS-5 Rev. 04



To debug the running application, click the arrow next to the Debug button and select **Debug** Configurations.



- Then, select the application and target you want to debug and select Connect only.
- Finally, hit the **Debug** button in the lower right corner.



 The debugger will connect to the selected application. You can stop the selected core and debug the booted image.

4 MQX Task Aware Debugging in Development Studio 5 (DS-5) IDE

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

The TAD plug-in is distributed separately from MQX release and directly from ARM[®] Ltd. For detailed documentation, contact your ARM distributor.

Example of available DS-5 TAD menu:

