Document Number: MQXRTOS
Rev. 03, 04/2013

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

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	Freescale MQX 4.0.1
DESCRIPTION:	Using ARM® Development Studio 5 (DS-5™) with Freescale MQX™ RTOS
RELEASE DATE:	April 19th, 2013

How to Reach Us:

Home Page:
www.freescale.com

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

Document Number: MQXRTOS
Rev. 03, 04/2013

Table of Contents

1 Read Me First

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

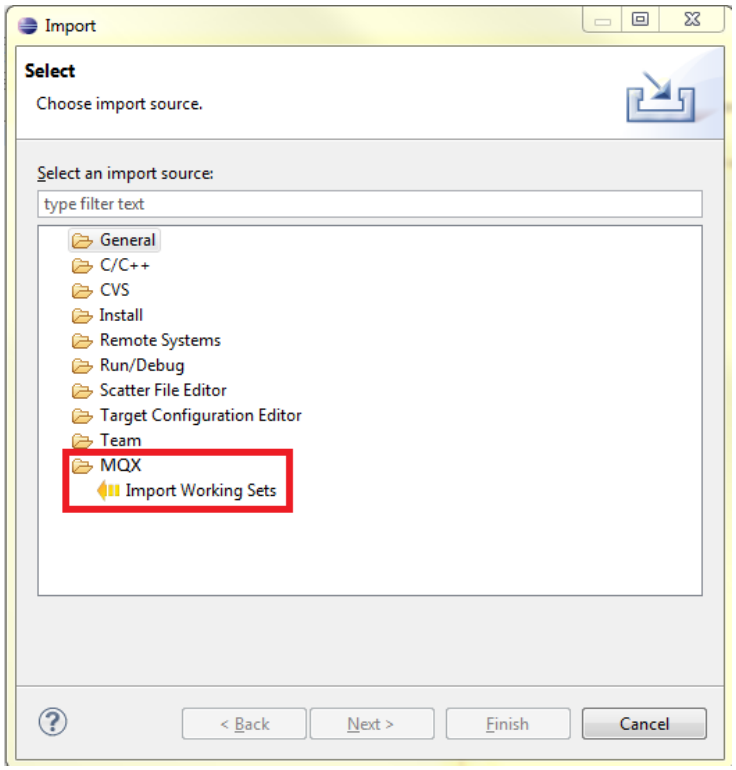
Refer to “Getting Started” and other user documentation included within the latest Freescale MQX™ RTOS distribution for further information on board specific build targets, jumper and HW settings, MQX API documentation, etc.

2 Building the MQX Libraries

This chapter concentrates on steps specific to DS5 tool chain only. For details on generic build process and compile time configuration, Chapter 2 of the MQX Getting Started document.

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>.wsd working set description file using ***File\Import\MQX\Import 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

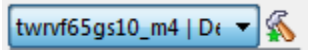
<mqx_install_dir>/mfs/*build/ds5/mfs_<board>*/.project

<mqx_install_dir>/rtcs/*build/ds5/rtcs_<board>*/.project

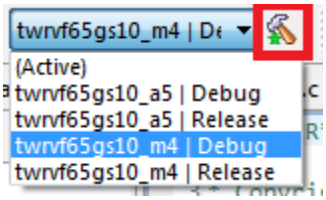
<mqx_install_dir>/usb/*host/build/ds5/usbh_<board>*/.project

<mqx_install_dir>/usb/*device/build/ds5/usbd_<board>*/.project

<mqx_install_dir>/shell/*build/ds5/shell_<board>*/.project

- Select the target and platform and build the libraries - hit the compile all button 

. 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.



3 Running and Debugging MQX application

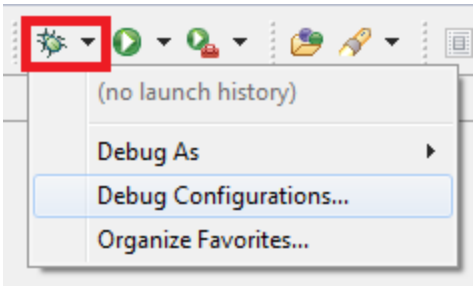
The description bellow is provided for Vybrid microcontrollers BSPs - twrvf65gs10_a5 and twrvf65gs10_m5 and Hello World example application. The twrvf65gs10_a5 BSP runs on primary and twrvf65gs10_m4 runs auxiliary Vybrid core. The same procedure applies for all other BSPs and example applications distributed in the MQX release package.

3.1 Debugging Primary Core - MQX Hello World program

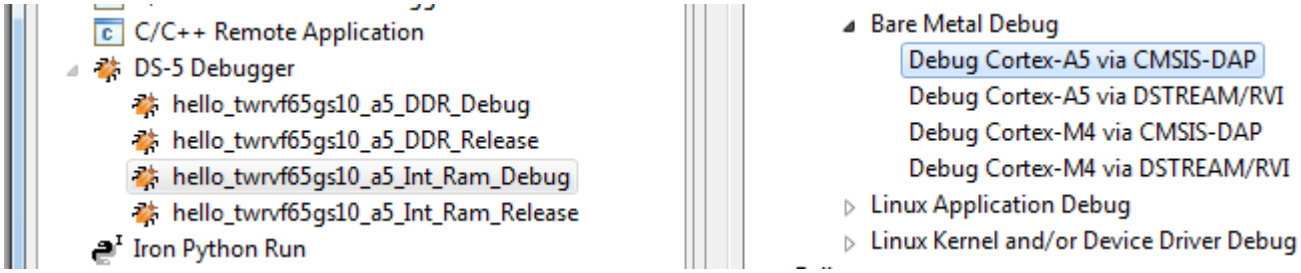
- Connect a serial cable to the TWR-SER or TWR-SER2 board DB9 connector. Set the communication speed to 115200.
- Select menu ***File/Import/General/Existing Projects into Workspace*** and import Hello World example application.

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

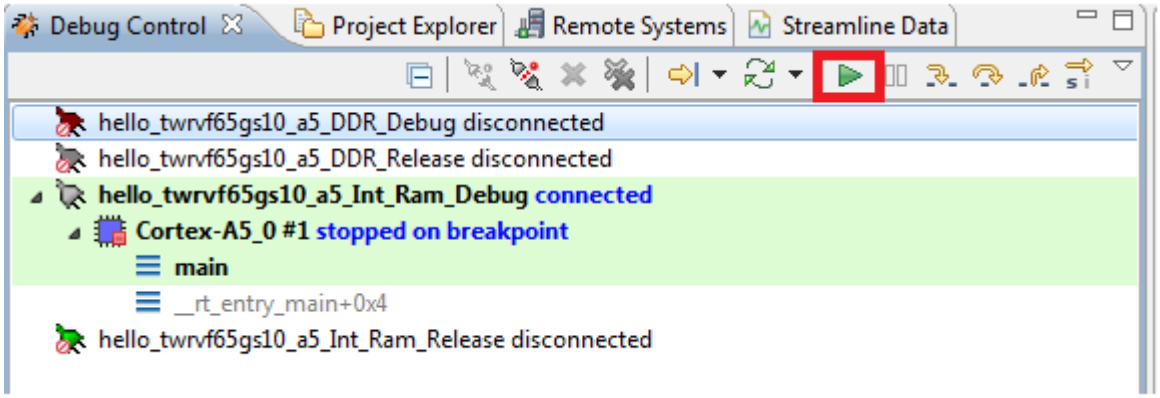
- Hit the compile button  to build application ***Int. RAM Debug target***.
- Click the arrow next to the Debug button and select ***Debug Configurations***.



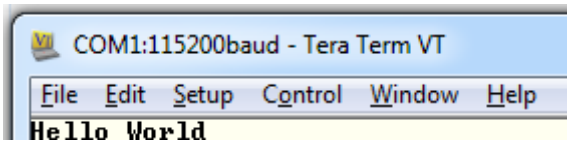
- A dialog box will come up. Select the ***hello_twrvf65gs10_a5_Int_Ram_Debug*** configuration in the ***Vybrid Cortex-A5 CMSIS-DAP*** debug connection. Then hit the **Debug** button in the lower right corner.



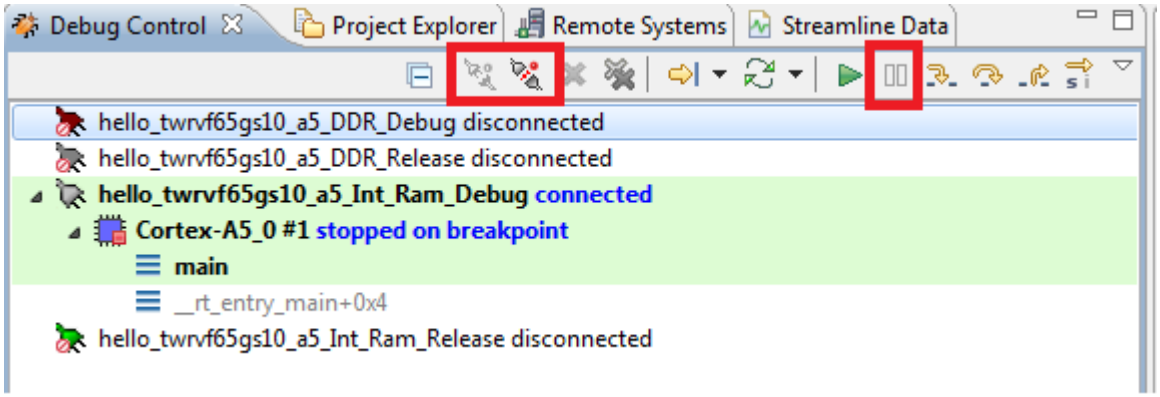
- The DS5 will switch to Debug Perspective automatically. Then, the project will be loaded to the device and execution will stop in the main() function.



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



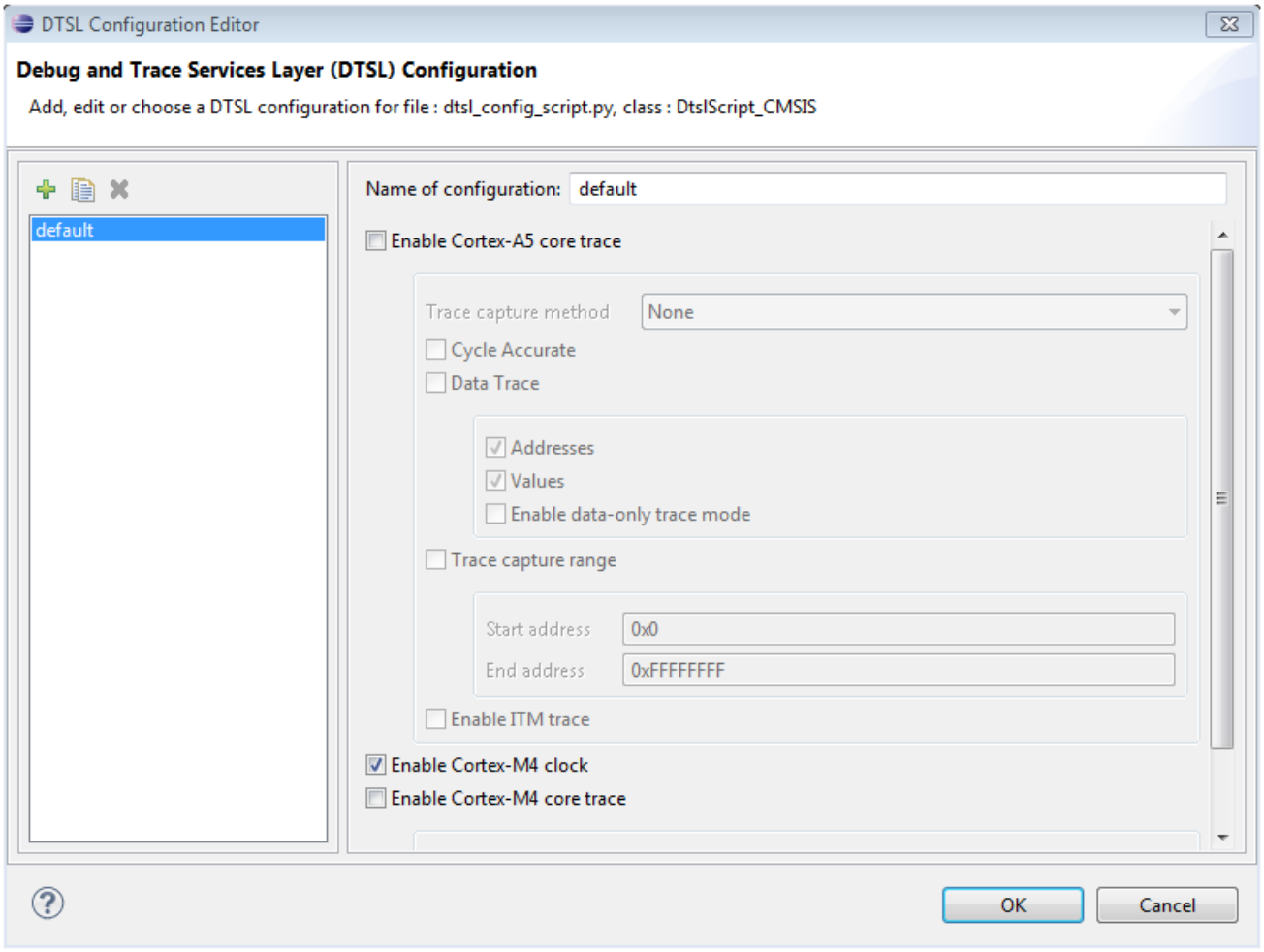
- To debug the application again, push ***Interrupt*** button to stop program execution**.** Then press ***Disconnect from Target*** and ***Connect from Target*** buttons in ***Debug Control*** Menu.



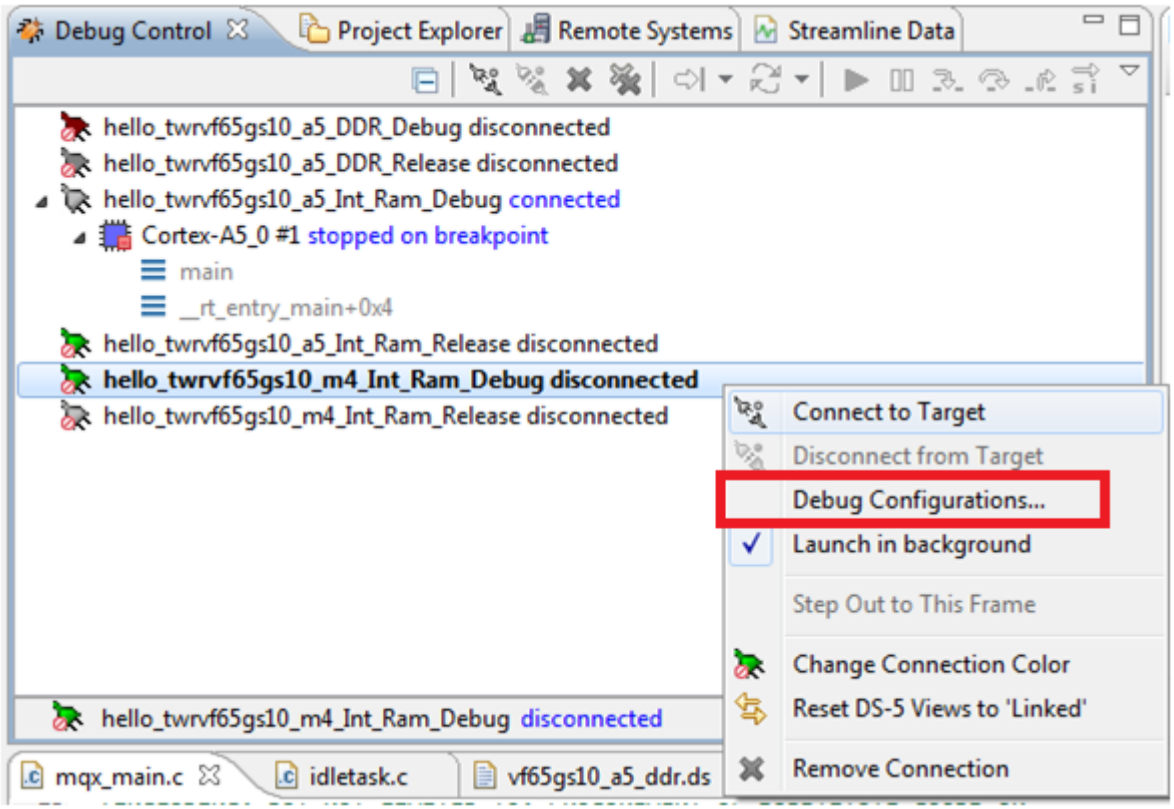
Note: If subsequent connections to the target fail, it is recommended to reset the board by using the Reset button on TWR or by using PWR down/up sequence on the TWR elevator.

3.2 Run MQX Hello World program on auxiliary core on dual core system (Vybrid CortexM core)

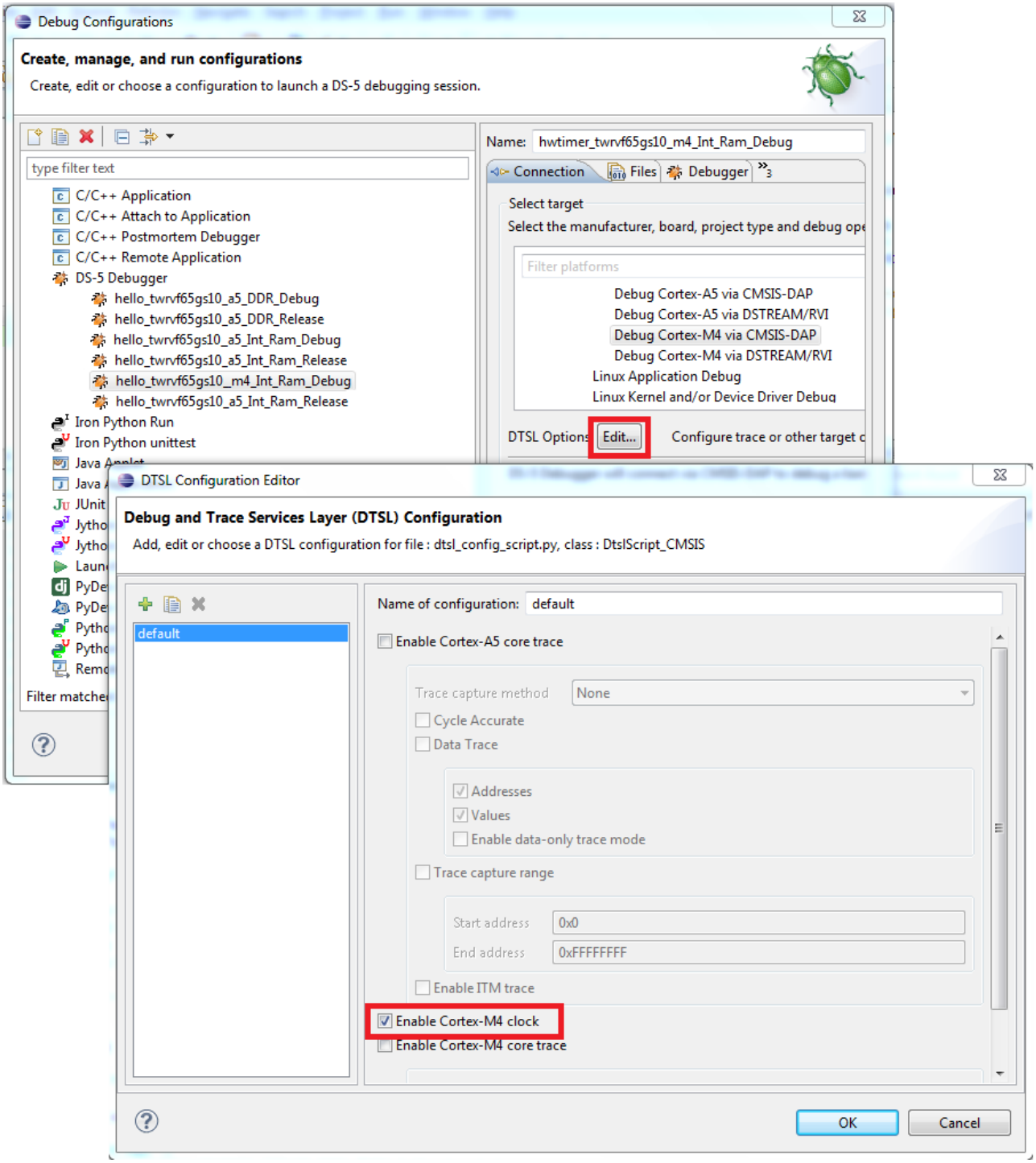
- Before loading the application to the primary core as described in the previous chapter, it is important to check the **Enable Cortex-M4 clock** in the **DTSL Options** menu (accessible from the **Debug Configuration** dialogue).



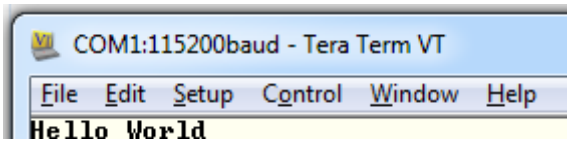
- Run the primary core application and stop the program execution using the ***Interrupt*** button .
- Switch to **C/C++ Perspective** using button in right top corner .
- Select menu ***File/Import/General/Existing Projects into Workspace*** and import Hello World example application for CortexM4 auxiliary 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.
- Switch back to **Debug Perspective** using button in right top corner .
- 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 CortexM4 core will print “Hello World” on the serial console.



Note: It is always necessary to execute MQX application on primary core before. The MQX primary core application startup sequence contains settings required by CortexM4 core (clock setup etc).

3.3 Multi-core debugging

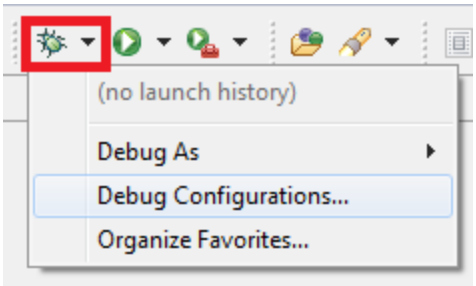
This chapter describes the basics of multi-core debugging with MQX RTOS. Description is provided for Vybrid microcontroller BSPs - twrvf65gs10_a5/twrvf65gs10_m4 and Multicore Communication (MCC) “pingpong” example application. The CortexA5 is primary core in this setup while the CortexM4 is set to auxiliary core.

- Select menu ***File/Import/General/Existing Projects into Workspace*** and import CortexA5 and CortexM4 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 CortexA5 and CortexM4 core as follows:

<mqx_install_dir>/mcc/examples/pingpong/ds5/pingpong_example_twrvf65gs10_a5/.project

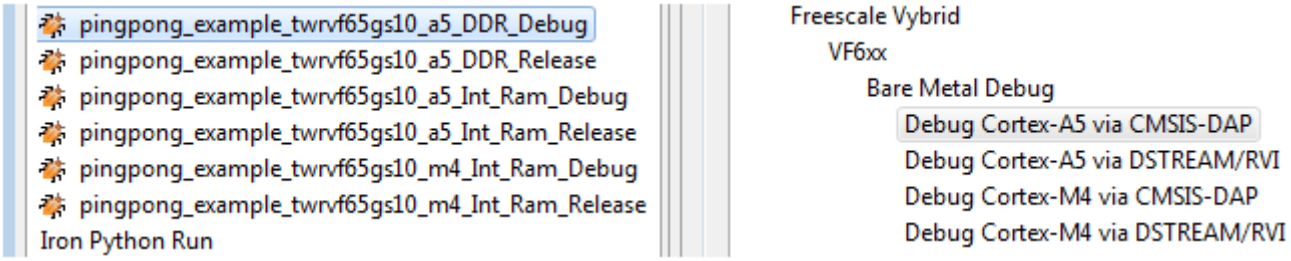
<mqx_install_dir>/mcc/examples/pingpong/ds5/pingpong_example_twrvf65gs10_m4/.project

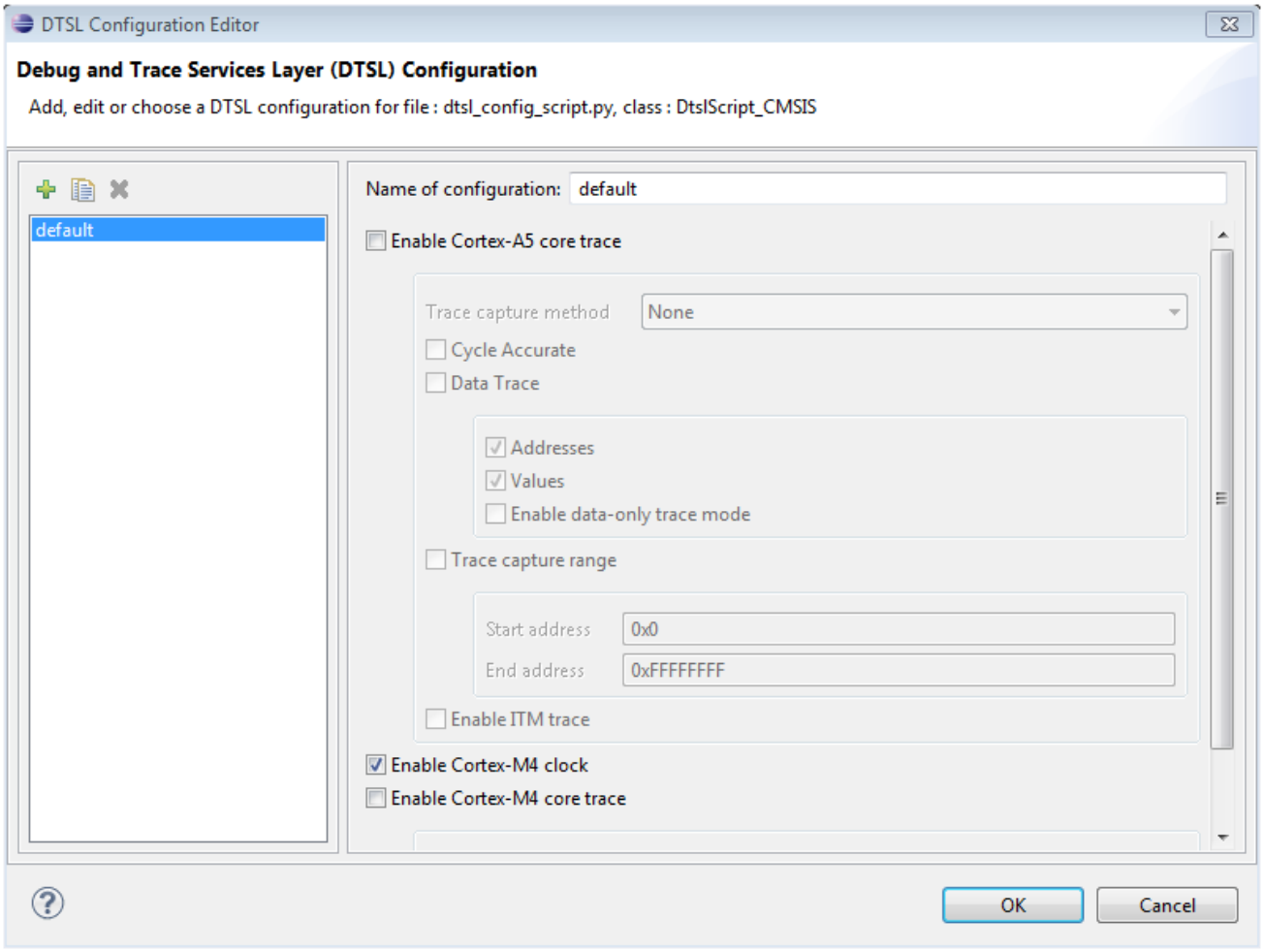
- Select the *pingpong_example_twrvf65gs10_a5* project in the Project Explorer View and then hit the compile button  to build the ***DDR Debug target***.
- Select the *pingpong_example_twrvf65gs10_m4* project in the Project Explorer View and then hit the compile button  to build the ***Int Ram Debug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



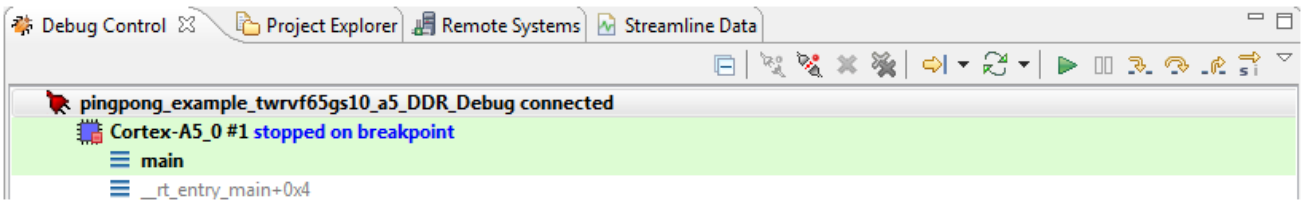
- A dialog box will come up. Select the ***pingpong_example_twrvf65gs10_a5_DDR_Debug*** configuration and the ***Vybrid Cortex-A5 CMSIS-DAP*** debug connection. Before hitting the

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

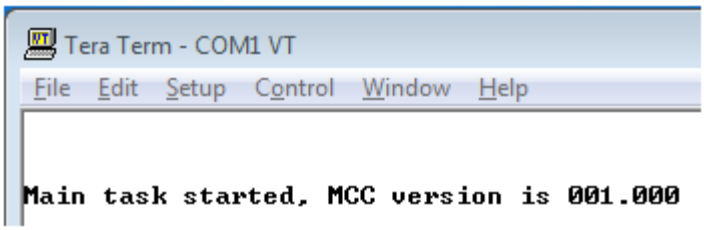




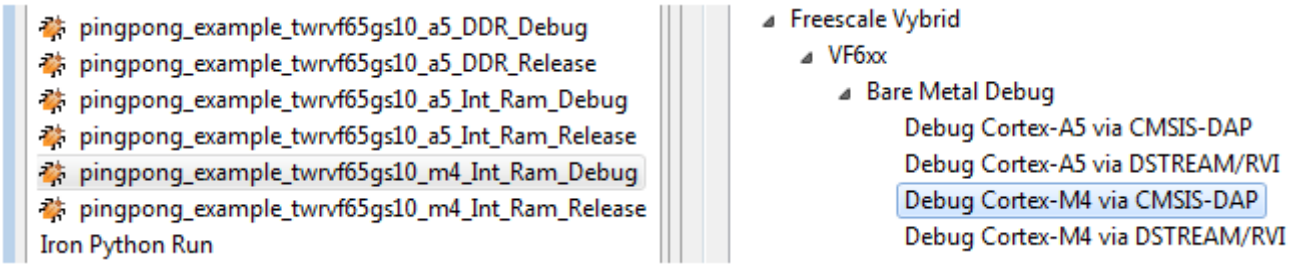
- The DS5 will switch to Debug Perspective automatically. Then, the project will be loaded to the device and execution will stop in the main() function.
- Press the **Run** button to continue in the *pingpong_example_twrvf65gs10_a5* program execution.



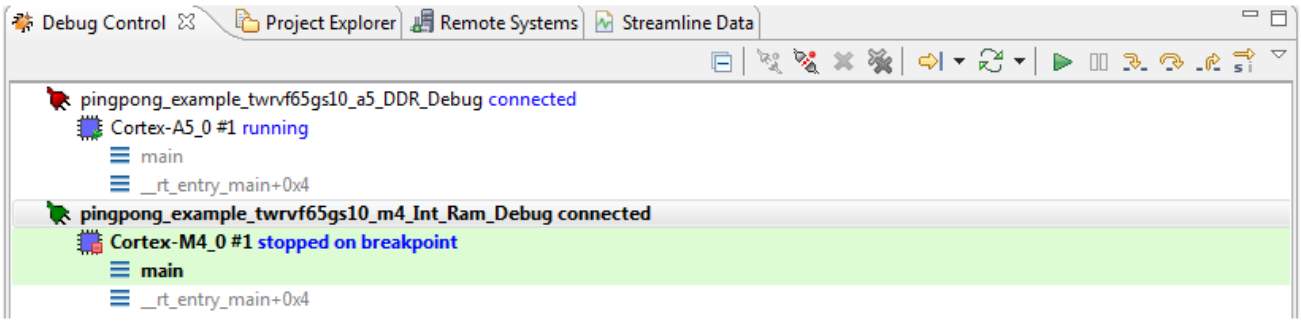
- The program will print “Main task started, MCC version is ___.___” on the serial console terminal.



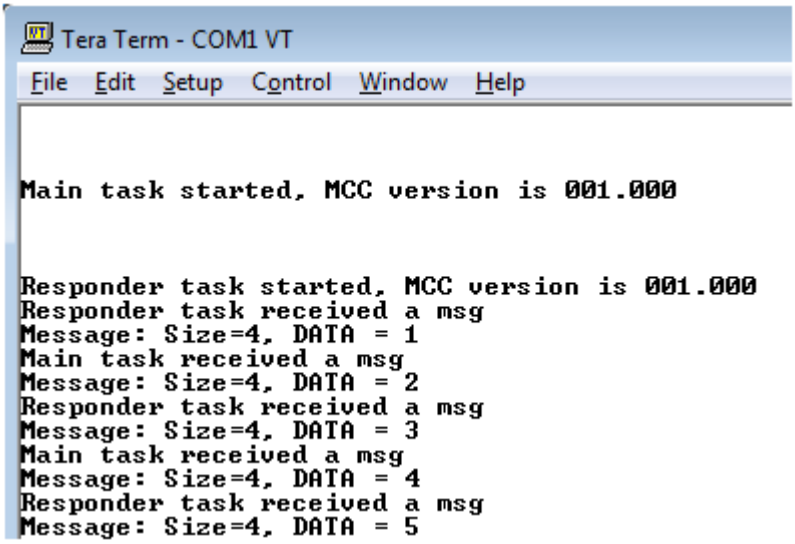
- Once the CortexA5 code application is running, start the execution of the auxiliary core (CortexM4).
- Select menu ***Run/Debug Configuration***s.
- When the Debug Configuration dialogue occurs, select the ***pingpong_example_twrvf65gs10_m4_Int_Ram_Debug*** configuration and the ***Vybrid Cortex-M4 CMSIS-DAP*** debug connection.
- Then hit the **Debug** button in the lower right corner.



- The CortexM4 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.



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:

<mqx_install_dir>/mqx/examples/bootloader_vybrid/ds5/bootloader_vybrid_twrvf65gs10_a5

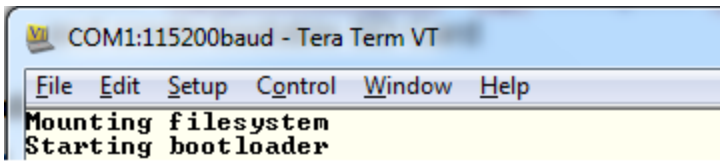
- Select Int Ram Debug target and hit the compile button  .
- Use DS5 "*C:\Program Files\DS-5\bin\fromelf*" utility to create the binary image:

```
fromelf.exe --bin --output=bootloader_vybrid_twrvf65gs10_a5.bin
bootloader_vybrid_twrvf65gs10_a5.axf
```

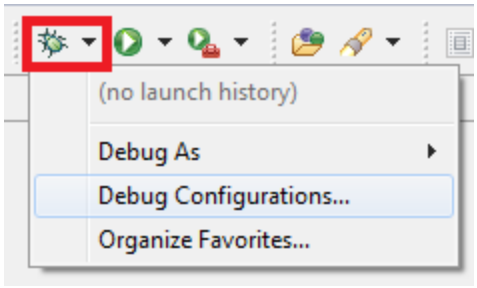
- Follow *<mqx_install_dir>\mqx\examples\bootloader_vybrid\Readme.txt* description and use prepare binary image to prepare the bootable SD Card.

Building and Debugging the Application images

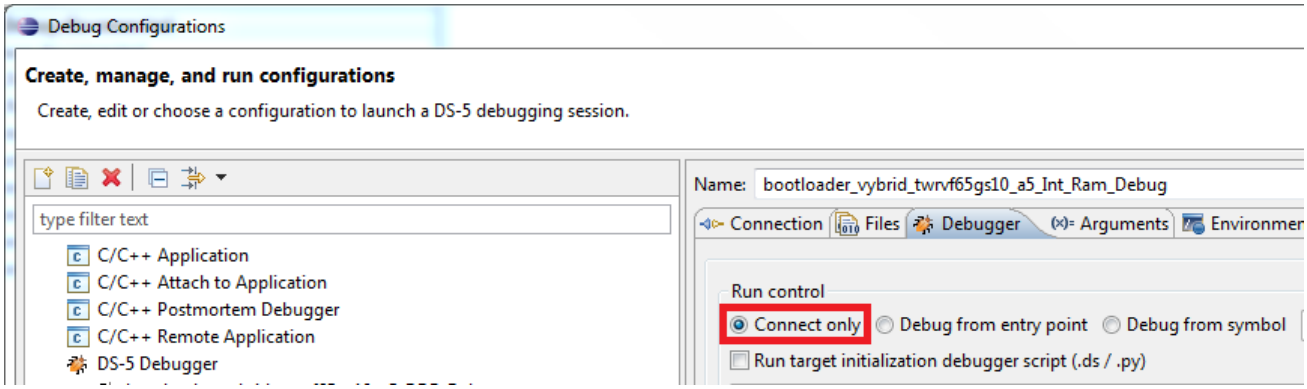
- Build the applications you want to run on A5 and M4 cores and convert them to binary format (.bin) by using *fromelf* DS-5 utility.
- Store the binary images on the root directory on bootable SD card.
- Copy setup.ini to the SD Card and modify according to Readme.txt description.
- Remove the SD Card 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.



- 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 DS5 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 DS5 TAD menu: