

Getting Started with Freescale MQX™ RTOS for Kinetis SDK and Atollic TrueSTUDIO® for ARM®

1 Read Me First

This document describes the steps required to configure Atollic TrueSTUDIO to build, run, and debug MQX™ RTOS demo applications and necessary driver libraries provided in the Kinetis SDK (KSDK framework). The Hello World demo application targeted for the TWR-K64F120M Tower System hardware platform is used as an example in this guide.

Contents

1	Read Me First.....	1
2	Installing MQX RTOS plug-ins.....	2
3	Import project files into workspace	4
4	Build library project files.....	6
5	Build the application project file	8
6	Run a demo application.....	10
6.1	Debugging demo application using MQX RTOS Task Aware Debugger for GDB plug-in	13
7	Revision history	16

2 Installing MQX RTOS plug-ins

There are two plug-ins available for MQX RTOS Task Aware Debugger for GDB and Project of Projects (allows import multiple project to the Working set). Follow the guide below to install them to Atollic TrueSTUDIO IDE.

1. Select “Help” then “Install New Software...” in the toolbar menu .

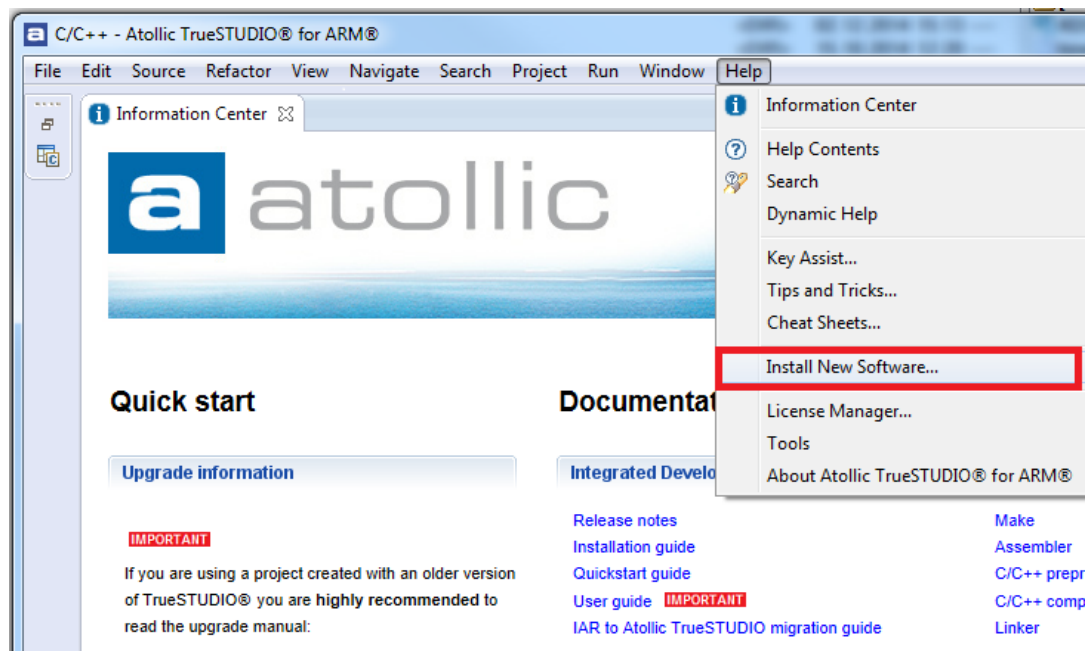


Figure 1: Install New Software

2. In the Install New Software dialog box, click the “Add” button and in the Add Repository pop up window enter KDS Update Site URL (<http://www.freescale.com/lgfiles/updates/Eclipse/KDS/>) to the Location field. Click the “OK” button.

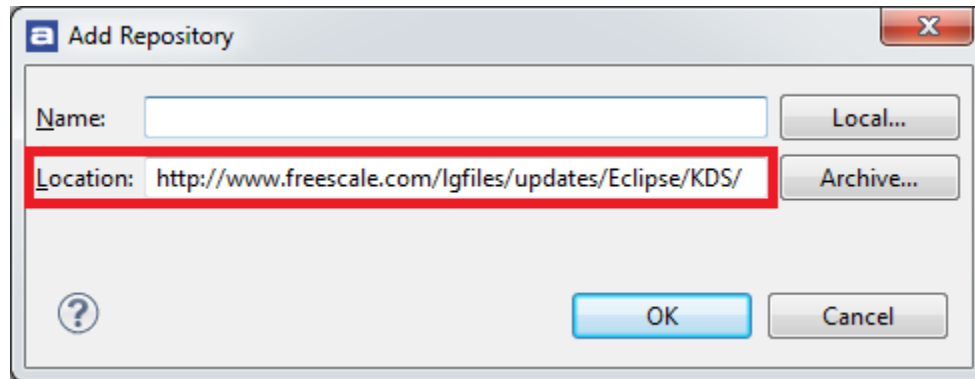


Figure 2: Add KDS Update Site as new repository

3. Three categories from update site are displayed. Select MQX RTOS Plug-ins category, which installs both MQX RTOS TAD and Project of Projects plug-in. Click the “Next” button in lower right corner.

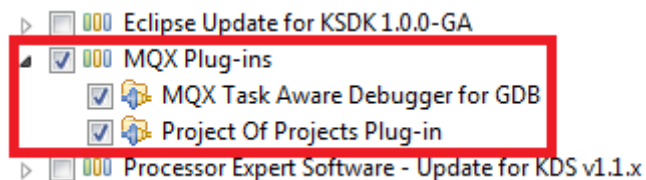


Figure 3: Categories on Update Site

4. Follow the remaining instructions to finish the installation of the update. After the update is applied, restart the Atollic TrueSTUDIO IDE for the changes to take effect.

3 Import project files into workspace

This section shows how to import project files with Project or Projects plug-in using workspace files (.wsd).

1. Click File then Import. In the Import dialog window, select Existing Projects Sets in category Project of Projects and click the “Next” button.

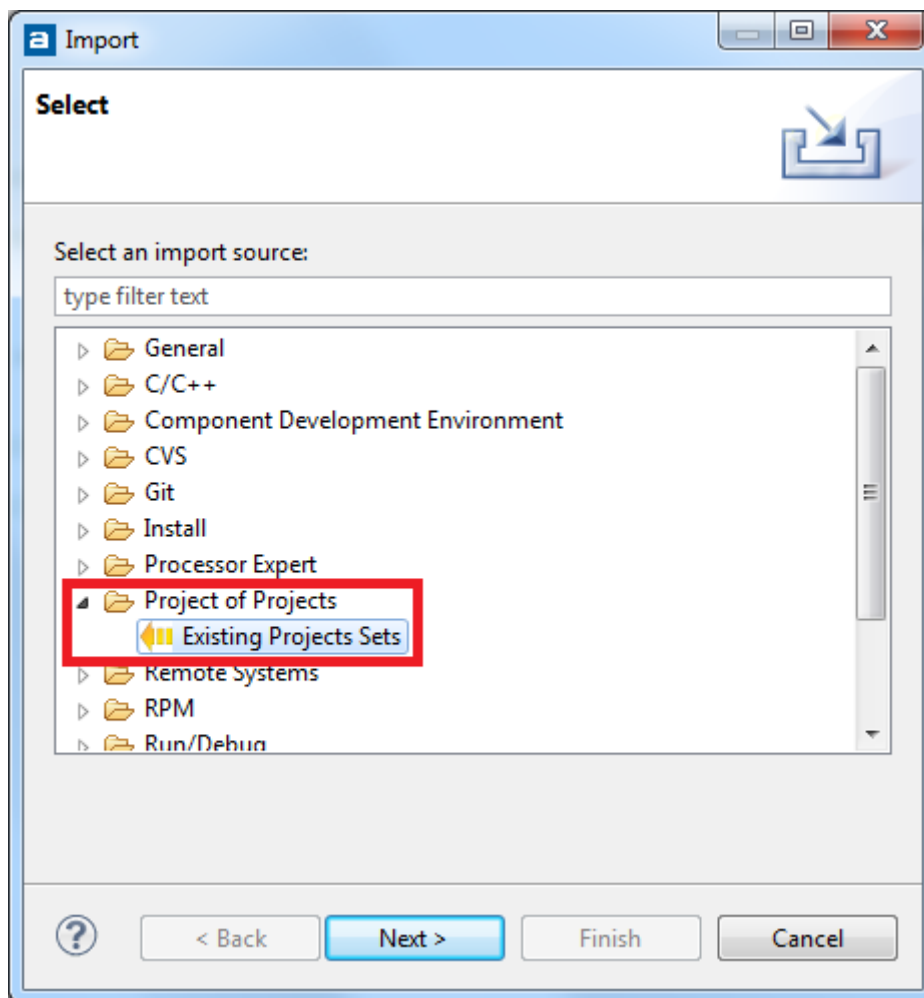


Figure 4: Import dialog

2. Click the “Browse” button to load wanted workspace file. The workspace files for MQX RTOS examples are located in the file:

`<install_dir>/rtos/mqx/mqx/examples/<demo_name>/build/kds/<demo_name>_<board_name>`

3. Click the “Finish” button when you are done. The MQX RTOS example application and associated libraries will be opened in your workspace.

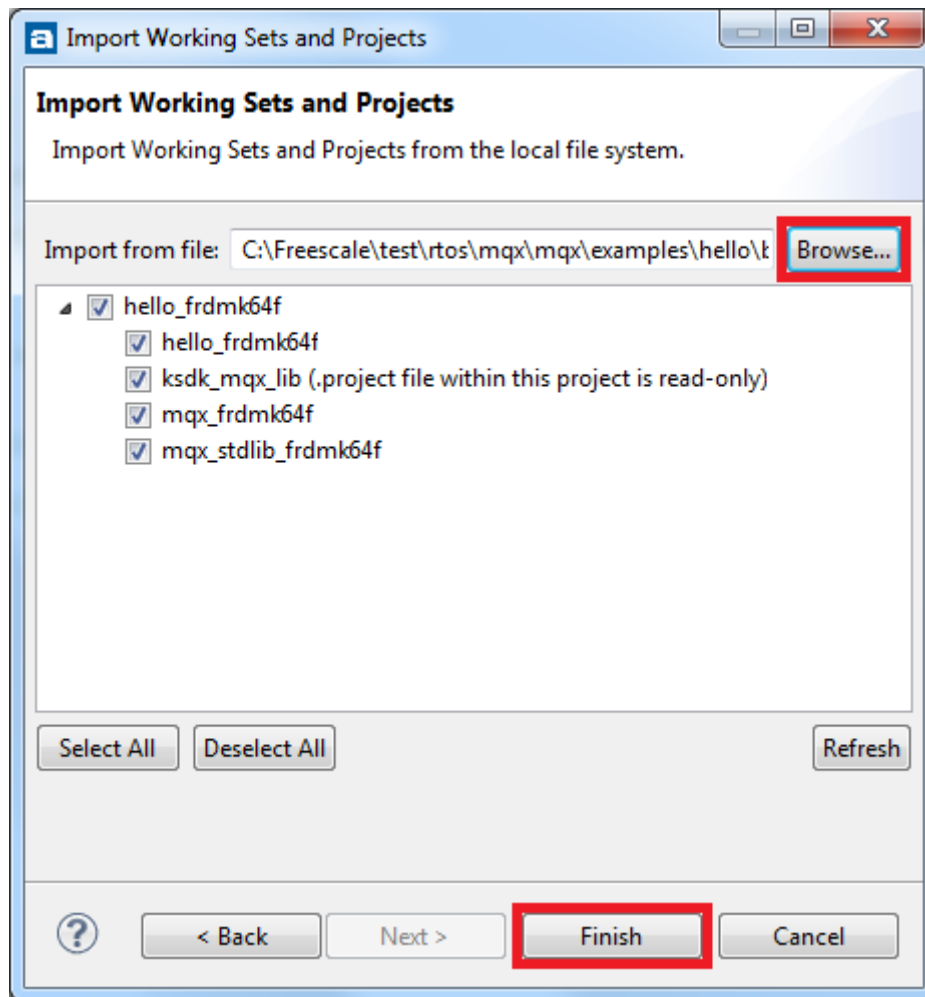


Figure 5: Import workspace file for MQX RTOS example application

4 Build library project files

This section will guide you to build the library project files of the hello example in MQX RTOS for KSDK for TWR-K64F120M. Follow the steps to build multiple libraries with different build targets at the same time.

1. Select the project files that you want to do build by pressing Ctrl and left-clicking the project files in the “Project Explorer” tab view. Right-click any project in the selected group and select “Build Configurations”, then “Build Selected”.

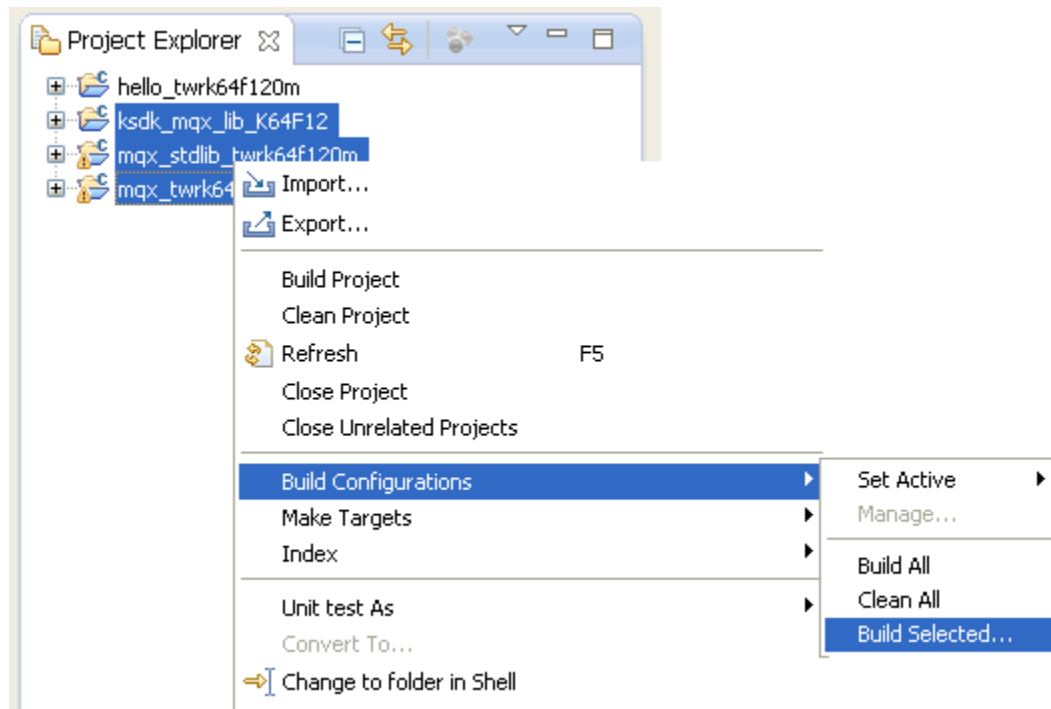


Figure 6: Select projects to build

2. In the Clean and Rebuild Configurations dialog window, select the projects you want to build with the corresponding build target. Click the “OK” button to start the build process.
3. The archive files are generated after a successful build of the library project files. They are located in the following folders for libraries mqx, mqx_stdlib, and ksdk_mqx_lib.

<install_dir>/rtos/mqx/lib/<board_name>.atl/<build_target>/mqx/<lib_mqx>.a

<install_dir>/rtos/mqx/lib/<board_name>.atl/<build_target>/mqx_stdlib/<lib_mqx_stdlib>.a

<install_dir>/lib/ksdk_mqx_lib/atl/<device_name>/<build_target>/<libksdk_platform_mqx>.a

For an example with TWR-K64F120M and the Debug target, the following library files are generated:

`<install_dir>/rtos/mqx/lib/twrk64f120m.atl /debug/mqx/<lib_mqx>.a`

`<install_dir>/rtos/mqx/lib/twrk64f120m.atl/debug/mqx_stdlib/<lib_mqx_stdlib>.a`

`<install_dir>/lib/ksdk_mqx_lib/atl/K64F12/debug/<libksdk_platform_mqx>.a`

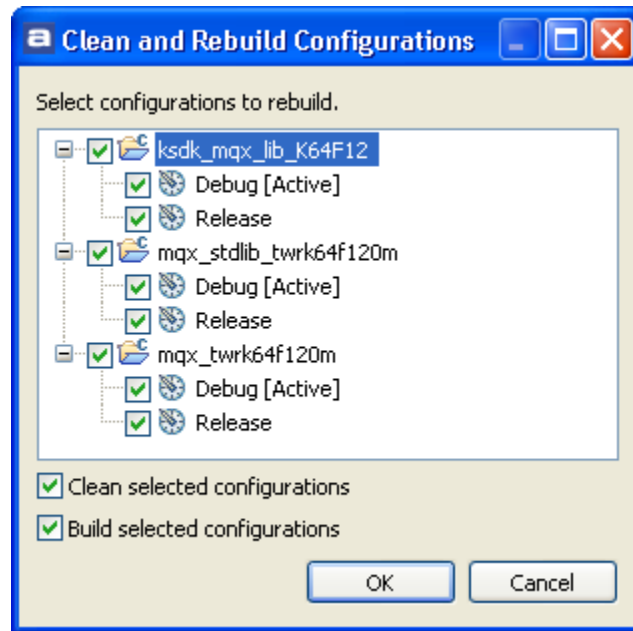


Figure 7: Select libraries to build

5 Build the application project file

After building the libraries, build the application project file. These are steps to build the hello example of MQX RTOS for KSDK for TWR-K64F120M.

1. Select the hello project and click the hammer icon (in red) to select build target.

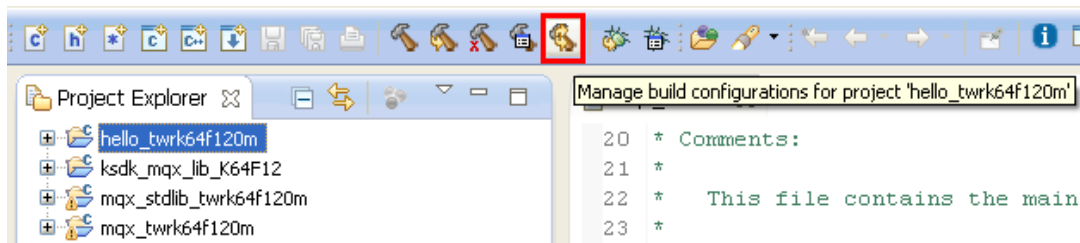


Figure 8: Select build target for hello project – 1

2. In the “Manage Configurations” window, select the build target and set it as the active build target by clicking the “Set Active” button. Click the “OK” button when it is complete.

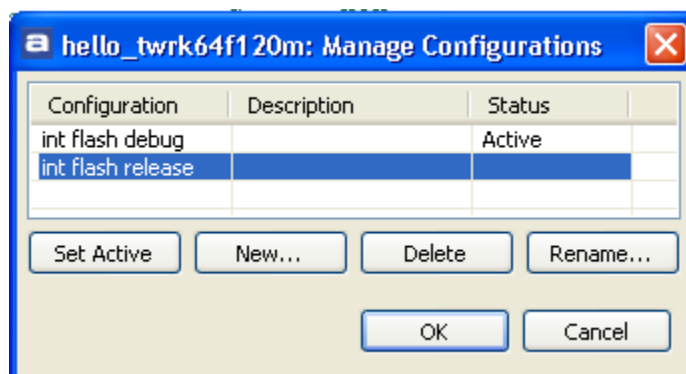


Figure 9: Select build target for hello project – 2

3. Click the “Build” button to build the hello project.

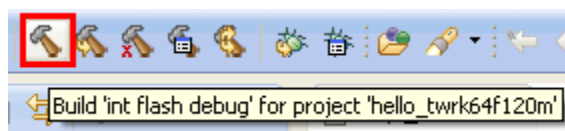


Figure 10: Build hello project

4. The build output message is shown in the console tab view when the build is done.



```
CDT Build Console [hello_twrk64f120m]
arm-atollic-eabi-gcc -c -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv
arm-atollic-eabi-gcc -c -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv
arm-atollic-eabi-gcc -c -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv
arm-atollic-eabi-gcc -c -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv
arm-atollic-eabi-gcc -Wl,--start-group Sources\hello.o KSDK_Files\pin_mux
C:\Program Files\Atollic\TrueSTUDIO for ARM Pro 5.1.1\ide\jre\bin\java -j
Generate build reports...
Print size information
      text      data      bss      dec      hex filename
  49996      1456      3536      54988      d6cc hello_twrk64f120m.elf
Print size information done
Generate build reports done

15:48:27 Build Finished (took 5s.469ms)
```

Figure 11: Build output message

6 Run a demo application

To download and run the application, follow these steps.

1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the OpenSDA serial port number. Configure the terminal with these settings:
 - a) 115200 baud rate
 - b) No parity
 - c) 8 data bits
 - d) 1 stop bit

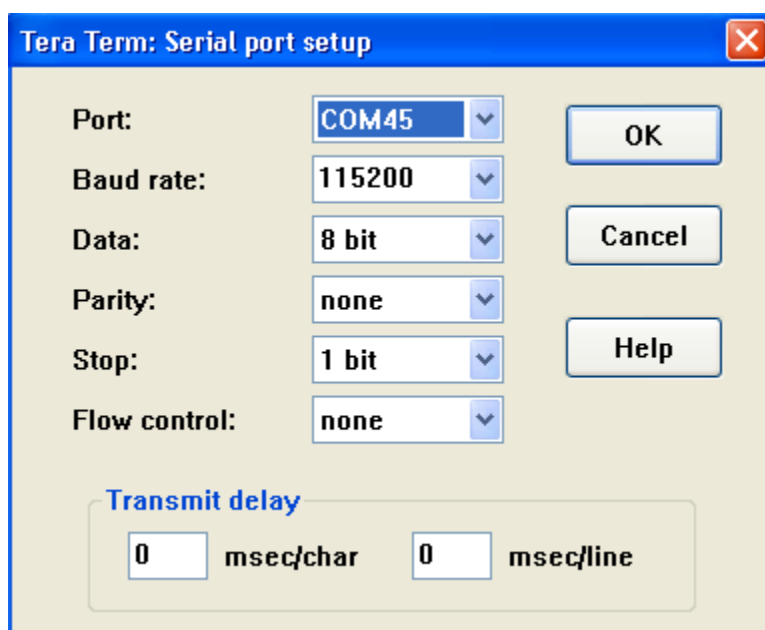


Figure 12: Setup serial terminal

3. Left-click the application project and click the “Configure Debug” button (in red) to launch the debug configurations window.

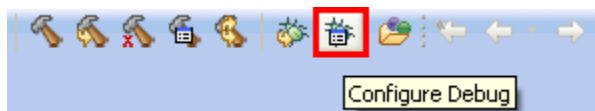


Figure 13: Open Debug configurations

4. In the “Debug Configurations” window, go to Embedded C/C++ Application (in red) in the left and select the appropriate application to load into the platform. In the “Main” tab (in green), select the correct output file to download into platform. In the “Debugger” tab (in purple), select the correct debugger tool, interface, and the connection type to download and debug the application.

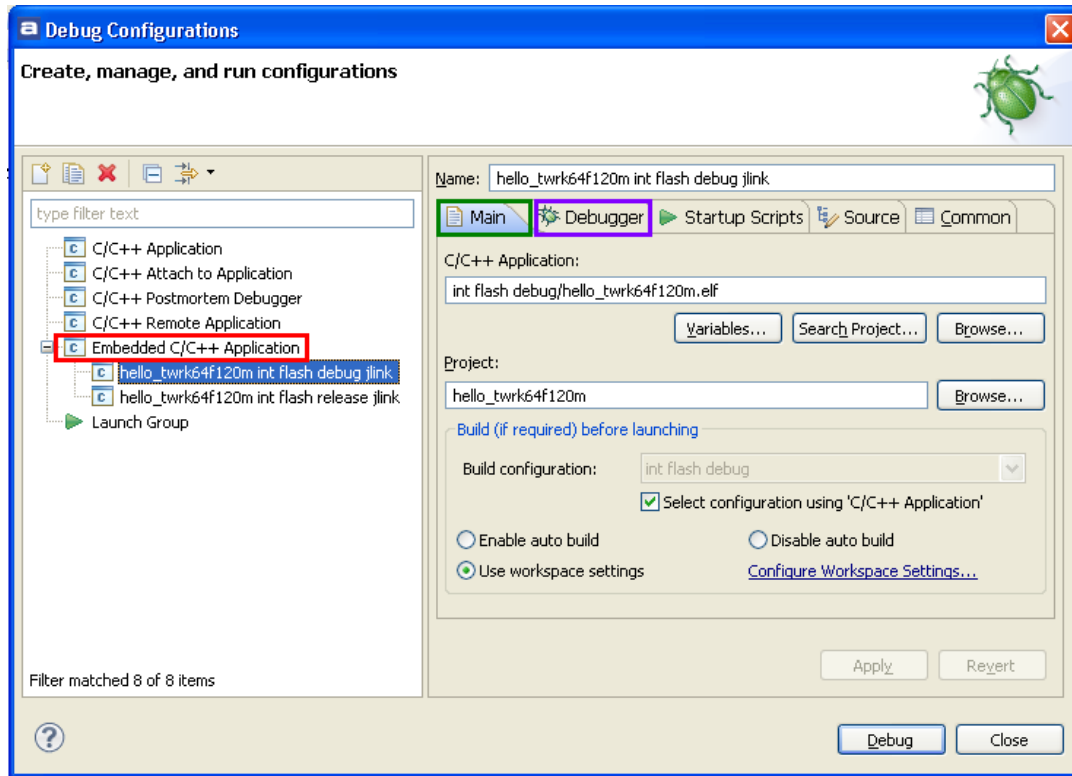


Figure 14: Debug configurations window

5. Click the “Debug” button at the bottom right corner to start loading application image into the platform and to debug.

6. The application runs into the main() function and stops as in shown in the green rectangle. Click the “Resume” button (in red) to run the application.

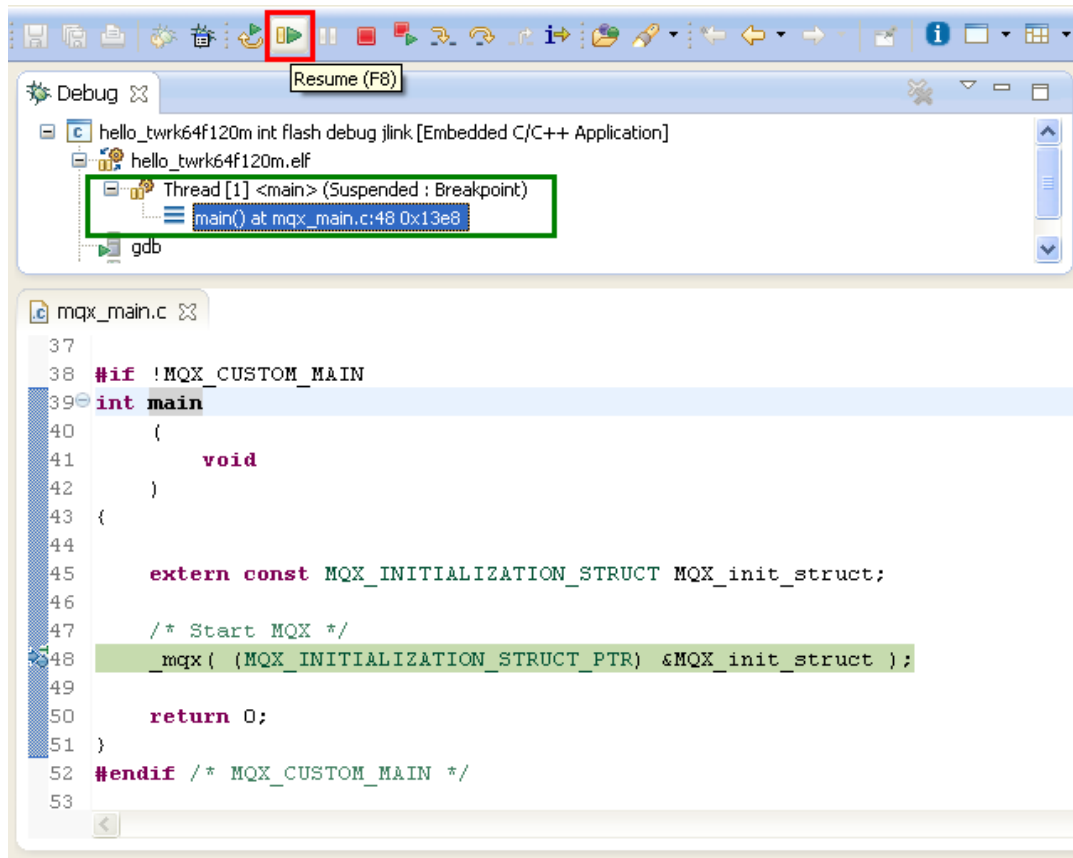


Figure 15: Run hello example

The output of the hello example is similar to this figure.

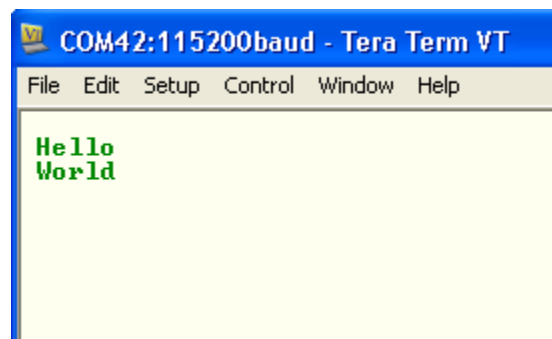


Figure 16: Output of hello example

6.1 Debugging demo application using MQX RTOS Task Aware Debugger for GDB plug-in

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.

MQX RTOS TAD Screens are accessible from MQX RTOS menu which is only displayed during the debug session.

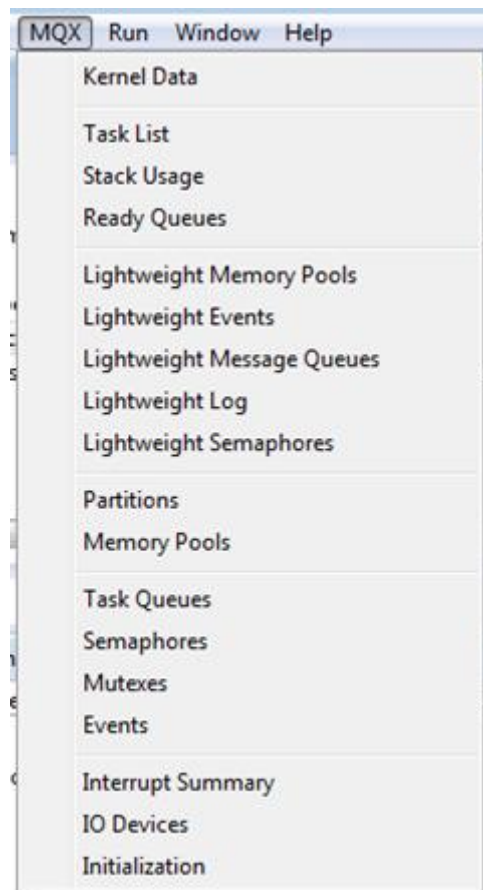
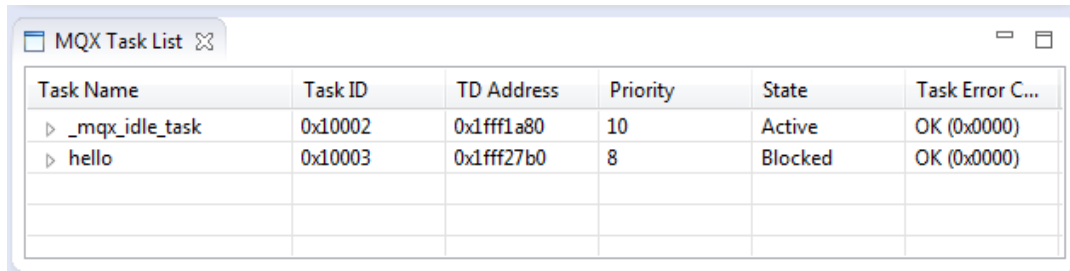


Figure 17: List of available TAD screens

Resume (F8 or the “Run/Resume” button)  the debug session and then suspend  (of Run/Suspend) it again to initialize MQX RTOS structures needed by MQX RTOS TAD.

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

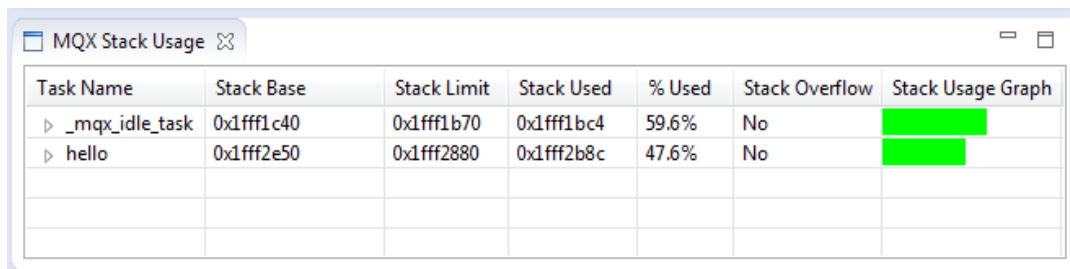
- Task List – Overview about all tasks created in the MQX RTOS application



Task Name	Task ID	TD Address	Priority	State	Task Error C...
▸ _mqx_idle_task	0x10002	0x1fff1a80	10	Active	OK (0x0000)
▸ hello	0x10003	0x1fff27b0	8	Blocked	OK (0x0000)

Figure 18: MQX RTOS Task List screen

- Stack Usage – 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.





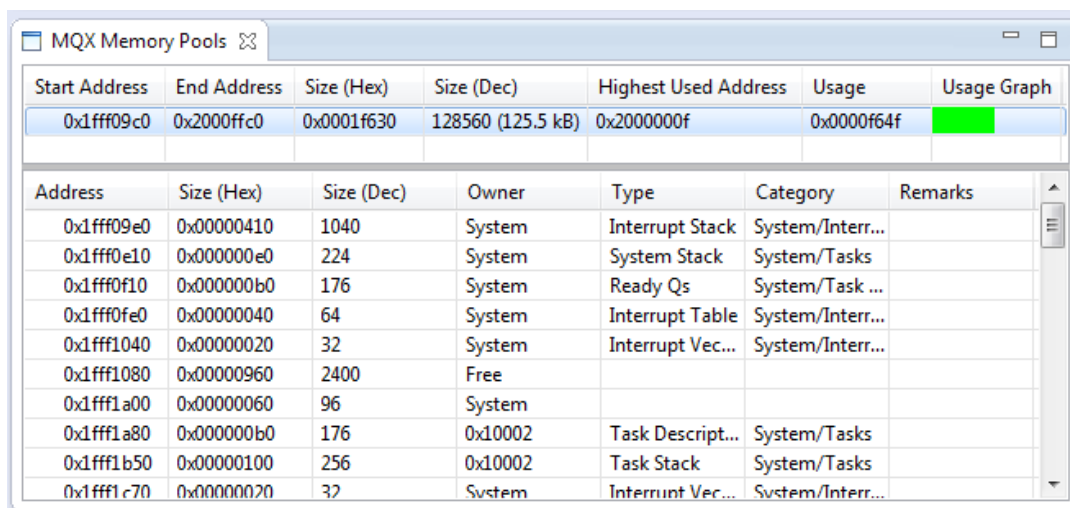

Task Name	Stack Base	Stack Limit	Stack Used	% Used	Stack Overflow	Stack Usage Graph
▸ _mqx_idle_task	0x1fff1c40	0x1fff1b70	0x1fff1bc4	59.6%	No	
▸ hello	0x1fff2e50	0x1fff2880	0x1fff2b8c	47.6%	No	

Figure 19: MQX RTOS Stack Usage screen

- Memory Pools (or Lightweight Memory Pools) – Displays address, size, and type information about each memory block allocated in the selected memory pool by the MQX RTOS system or applications.



Start Address	End Address	Size (Hex)	Size (Dec)	Highest Used Address	Usage	Usage Graph
0x1fff09c0	0x2000ffc0	0x0001f630	128560 (125.5 kB)	0x2000000f	0x0000f64f	

Address	Size (Hex)	Size (Dec)	Owner	Type	Category	Remarks
0x1fff09e0	0x00000410	1040	System	Interrupt Stack	System/Interr...	
0x1fff0e10	0x000000e0	224	System	System Stack	System/Tasks	
0x1fff0f10	0x000000b0	176	System	Ready Qs	System/Task ...	
0x1fff0fe0	0x00000040	64	System	Interrupt Table	System/Interr...	
0x1fff1040	0x00000020	32	System	Interrupt Vec...	System/Interr...	
0x1fff1080	0x000000960	2400	Free			
0x1fff1a00	0x00000060	96	System			
0x1fff1a80	0x000000b0	176	0x10002	Task Descript...	System/Tasks	
0x1fff1b50	0x00000100	256	0x10002	Task Stack	System/Tasks	
0x1fff1c70	0x00000020	32	System	Interrupt Vec...	System/Interr...	

Figure 20: MQX RTOS Memory Pools screen

- Semaphores, Events, Mutexes (or Lighthouse Semaphores, Lighthouse 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.

MQX Semaphores						
Semaphore	Owning #	Waiting #	Count	Destroy	Name	
0x1fff1190	0	2	0	No	sem.write	
0x1fff1200	0	0	2	No	sem.read	
0x1fff1270	0	0	0	No	sem.index	
Property		Value				
Semaphore ID		0x1fff1190				
Name		sem.write				
Valid?		Yes				
Destroy?		No				
Priority queuing?		No				
Priority inherit?		No				
Count		0				
Max count		Unknown				
Strict?		No				
Own/Wait:		Task:				
▸ Wait		write, ID:0x10001, TD:0x1fff12e0				
▸ Wait		write, ID:0x10004, TD:0x1fff13b0				

Figure 21: MQX RTOS Semaphores screen

7 Revision History

This table summarizes revisions to this document.

Table 1 Revision History		
Revision number	Date	Substantial changes
1	04/2015	Kinetis SDK 1.2.0 release
0	12/2014	Kinetis SDK 1.1.0 release

How to Reach Us:**Home Page:**

freescale.com

Web Support:

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 Kinetis 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. ARM, ARM powered logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2015 Freescale Semiconductor, Inc.