Getting Started with Runtime TAD Tool for Freescale MQX™ RTOS

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	3.8.1
DESCRIPTION:	Getting Started with Runtime TAD Tool for Freescale MQX™ RTOS
RELEASE DATE:	June 26th, 2012

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan**:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale
semiconductor

Rev. 2
06/2012

Table of Contents

Getting Started with Runtime TAD Tool

1 Read Me First

This document describes steps to configure and use the *MQX Runtime TAD* Tool to analyze, debug and fine tune the applications of the Freescale MQX™ RTOS operating system.

The MQX Runtime TAD Tool is an addition to the MQX Task-aware Debugger plug-in (TAD) with the similar features but using a different way of connecting to the target application. The Runtime TAD displays the same information as the "standard" MQX TAD without the need of a debugger connection. It can be connected to a running application via serial line, CAN, BDM or JTAG or it can analyze a memory dump file created with a regular MQX TAD during a debug session.

Note that the current version of MQX Runtime TAD Tool is a plug-in for CodeWarrior 10.x IDE. It can only be used within this IDE. The Runtime TAD shares resources with a regular CodeWarrior 10.x TAD so this plug-in needs also to be installed. Except the "from a file" communication mode, the Runtime TAD uses FreeMASTER communication drivers for communicating with the target boards. Although it is not required, installing (and understanding) FreeMASTER tool features will be helpful in most cases.

The FreeMASTER is a lightweight data visualization tool available without any additional cost to Freescale customers and covering most of the Freescale microcontroller and microprocessor platforms. Read more about the FreeMASTER tool at http://www.freescale.com/freemaster.

2 What is the MQX Runtime TAD Tool?

The MQX Runtime TAD Tool is a plug-in for CodeWarrior 10.x IDE which can be used to display various run-time information about an MQX application. There are the same views available in the CodeWarrior IDE as when using the standard "debugger-based" MQX TAD. The most useful TAD views are supported, for example the Task Summary view, Stack Usage and Memory Blocks views. The information is retrieved independently of the CW debugger from a running application, without the need of suspending or stopping the application at a breakpoint. The information can also be retrieved from an offline memory dump data file saved from Task-aware Debugger plug-in during any debugger session.

See more information about TAD in the *"Getting Started with Freescale MQX^{TM} RTOS"* document distributed within the MQX installation.

The first version of the MQX Runtime TAD Tool distributed with the MQX version 3.8.1 includes the following features:

- Connection to a target via serial line and using the "native" FreeMASTER communication protocol.

- Connection to a target via any of the FreeMASTER communication plug-ins. The communication plug-ins are available for OSBDM, P&E BDM and other interfaces. In order to make full use of the plug-ins, please install the FreeMASTER application.

- Displaying information from a memory dump file.

The MQX Runtime TAD Tool is available in the *MQX Tools* menu in the CodeWarrior 10.x IDE, together with other menu items of the *MQX Performance Tool* plug-in. The *MQX Performance Tool* is a separate plug-in with additional MQX analysis views documented separately in the *Getting Started with Performance Tool for Freescale MQX™ RTOS* document.
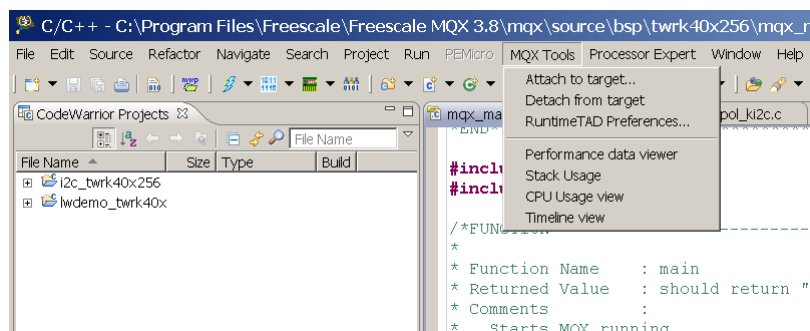
The following commands are available after installing the *MQX Runtime TAD* plug-in:

- **Attach to target…** - This command opens dialog box and enables to select a communication interface.

- **Detach from target…**This command closes the communication interface with the Runtime TAD.

- **RuntimeTAD Preferences…** This command opens dialog box with Runtime TAD preference. This dialog also displays the current connection status.

Getting Started with Runtime TAD Tool

3 Using MQX Runtime TAD Tool

This section describes how to use MQX Runtime TAD tool for analyzing your MQX application. The Runtime TAD uses the same views as MQX TAD for presenting and visualizing the data. For more information about MQX TAD and also about building and debugging MQX applications please refer to the *"Getting Started with Freescale MQX^{TM} RTOS"* document distributed within the MQX installation.

More information about the FreeMASTER application and a communication protocol is available in documentation distributed within the FreeMASTER installer package.

3.1 Usage

Before describing use of the Runtime TAD step-by-step in more details, this section outlines the basic usage principles.

3.1.1 Working within IDE

The Runtime TAD requires CodeWarrior 10.2 and MQX 3.8.1 (or later) installed and used. Unlike when using common MQX TAD, the Runtime TAD does not require a debugger session to be active. It is even not desired the application is running under a debugger connection because in this case the MQX TAD and Runtime TAD would compete for the CodeWarrior visualization views.

The use case for Runtime TAD is an analysis of an MQX application running autonomously and standalone out of Flash.

3.1.2 Target Application

Any running MQX application can be analyzed with Runtime TAD. Generally there are three ways of using the Runtime TAD:

- Use the "native" FreeMASTER serial or CAN communication protocol between Host PC and target application. In this case a FreeMASTER driver needs to be part of the target application code. See more in section 3.2 below.

- Use a non-intrusive BDM or JTAG mode of communication. The FreeMASTER contains plug-ins which may access the target application memory without any additional code on the target side. There is a potential risk of inconsistent data being obtained from if Host PC accesses the data while it is being modified. See more in section 3.3 below.

- Use an offline mode and "connect" the Runtime TAD to an application memory dump file.

3.1.3 Attaching to Target

In the CodeWarrior IDE, go to the *MQX Tools* menu and select *Attach to target…* The dialog box opens and you will be able to select communication mode.

Getting Started with Runtime TAD Tool
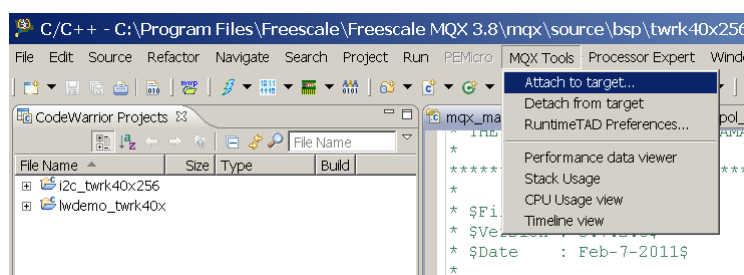
As the target is not running under a debugger session, the plugin needs to obtain variable addresses directly from within the ELF file. Before pressing OK in the dialog, make sure the *ELF File* path is set correctly and points to a binary being currently executed on the target board.
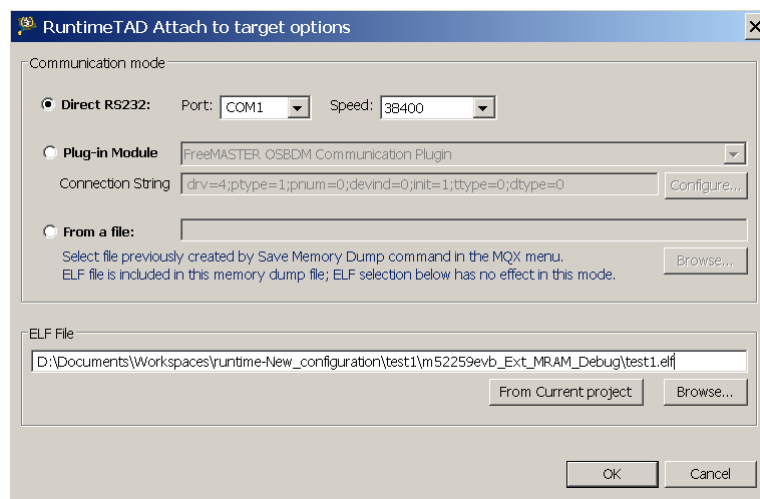
Pressing the OK button, the Runtime TAD will attempt to connect to the target application using selected communication interface and will inform you about the result.

3.1.4 Using TAD Views

When a connection is successfully established, the CodeWarrior IDE switches to the *Debug* perspective and the *Task summary* view gets activated. You should be able to see the MQX tasks running on the connected target.

Switching to the *Debug* perspective may be confusing as there is no debugger session active; on the other side it is a convenient way how to bring up the TAD views and menus in the same layout as when debugging with common MQX TAD.

Getting Started with Runtime TAD Tool
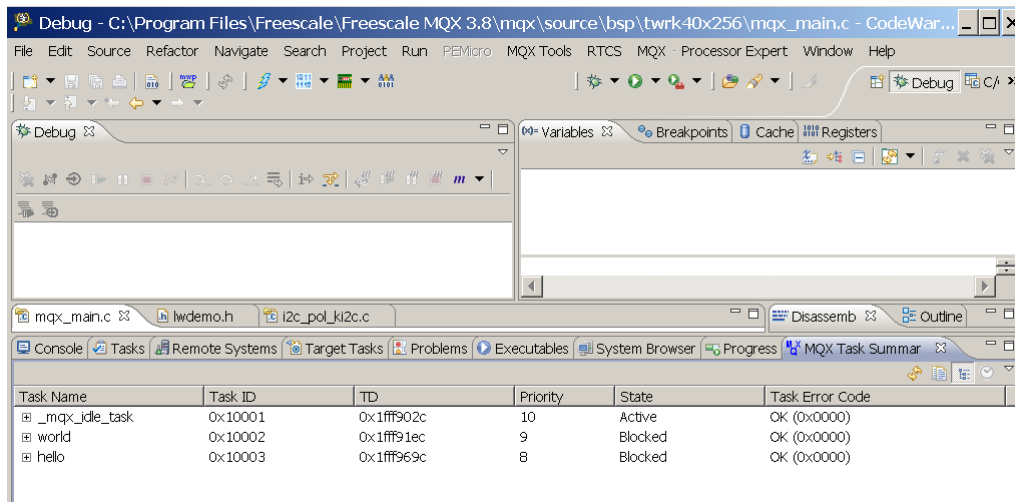
3.1.5 Using TAD Menus

When target is attached, the MQX and RTCS menus are available to open various views. Note that availability of individual menu items depends on the configuration and features used in the MQX target application.
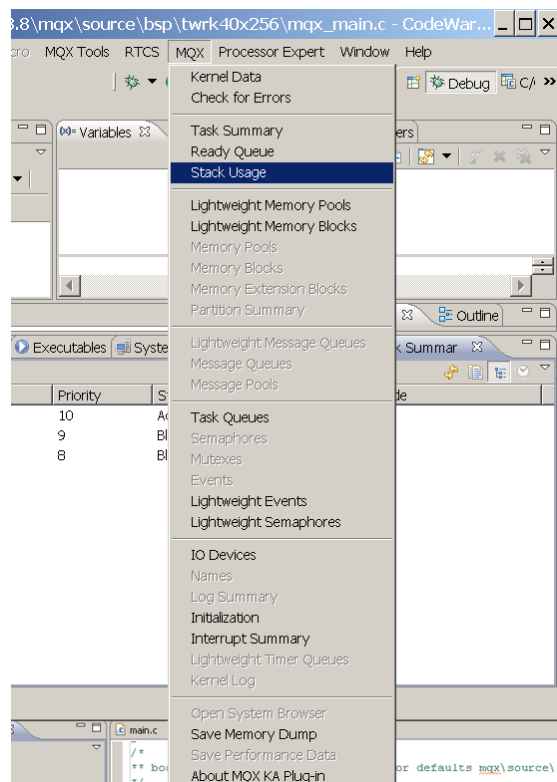
3.1.6 Closing the connection

When you are done analyzing the application disconnect the target using *Detach from target* command in the *MQX Tools* menu.

3.1.7 Limitations and Risks

Important Note: Both and advantage and a limitation of the Runtime TAD use is that it runs asynchronously and independently on the target application. It is a limitation in a way that the consistency of data memory cannot be granted as it may be changed at the moment of being read. It is recommended to refresh Runtime TAD screens when the application runs in an "idle" mode without intensive dynamic memory or MQX objects manipulations. Try to avoid cases when the Runtime TAD screens are refreshed during memory allocations, task creation/destruction, semaphore or other synchronization object creation/destruction performed by the target application.

With the direct serial communication mode, the integrity on one memory block being transferred to a host PC can be assured, however a consistency of dynamic structure queues and chains (e.g. list of semaphore objects) cannot be principally assured.

With the BDM communication mode, even an access to a single memory block happens asynchronously to target CPU. The data may be read and displayed by the Runtime TAD in an incomplete or inconsistent state.

3.2 Using Direct RS232 Communication

The *Runtime TAD* natively communicates with the target board using a serial interface. The FreeMASTER serial communication protocol has been selected as it enables also other features like variable graphing, visualization, etc. to be used independently on the Runtime TAD. Use of serial line protocol **requires a FreeMASTER Serial Driver task running in the target MQX application** which handles the communication. This section provides more details on setting up the application in order to make use of this communication mode.

For more information and application examples, refer to FreeMASTER Serial Communication Driver available at www.freescale.com/freemaster. The Driver package contains an MQX-enabled application which will be used in the further text as an example.

Steps to prepare MQX application with FreeMASTER communication driver:

1. Download and install the FreeMASTER Communication Driver.

2. Run the CodeWarrior 10 IDE and import the example project from FreeMASTER Serial Communication Driver directory to your workspace. The example projects for MQX are located (by default) in: *C:\Program Files\Freescale\FreeMASTER Serial Communication\examples\MQX_IO* folder.

3. Configure the application as required for your board and serial interface you will be using. Pay attention especially to the `FMSTR_MQX_IO_CHANNEL` in the *freemaster_cfg.h* file which defines the MQX IO channel used for the communication, for example the "ittyb:".

4. Make sure the driver for selected communication channel is enabled in the user_config.h file. For example for the "ittyb:" channel, you should have a line

```
#define BSPCFG_ENABLE_ITTYB     1
```

5. In case you make any changes to the user_config.h, you need to rebuild the MQX BSP and other libraries. Refer to the *Getting Started with Freescale MQX™ RTOS* document for a build procedure.

6. Compile and run the application.

Getting Started with Runtime TAD Tool

7. Disconnect the debugger, or restart (reset) the application without a debugger connected.

8. Continue with the steps described below. If you encounter problems with establishing the communication, restart the application in the debugger and make sure the `FMSTR_Init()` function returns a non-zero value. If not, the MQX serial driver could not be opened – most probably because it has not been enabled in the *user_config.h* file.

To attach to the application with Runtime TAD:

1. In CodeWarrior IDE go to *MQX Tools* menu and select *Attach to target…*
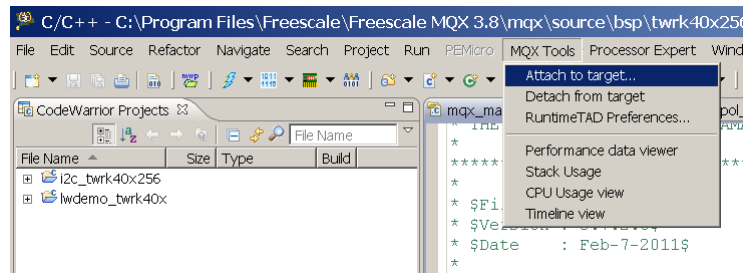
2. In the *Attach to target* dialog, select the **Direct RS232** communication mode and specify the Port and Speed used. The FreeMASTER MQX example is configured for 115200 bps by default.

3. In the *ELF File* area select the ELF file which corresponds to the application running on the target board. You can either browse to this file using the *Browse…* button or click the *From Current Project* button to use the output file of a project currently selected in CodeWarrior IDE.
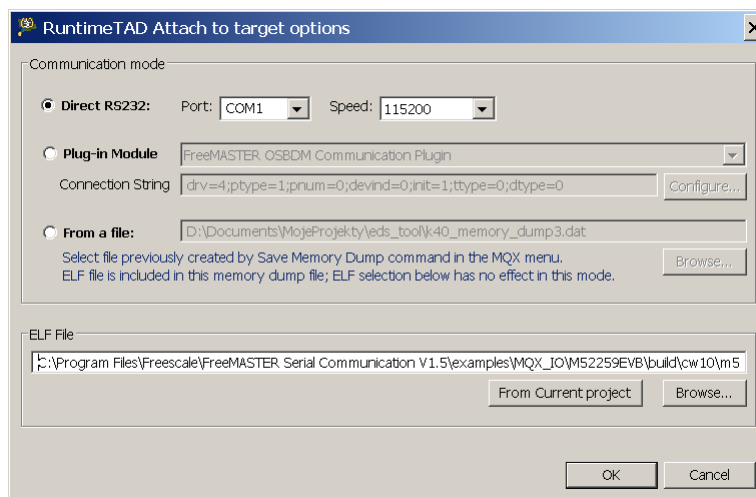

4. Click *OK* to close this dialog and connect to the target.

5. Runtime TAD will attempt to connect to the target application and inform you about the result.

Getting Started with Runtime TAD Tool
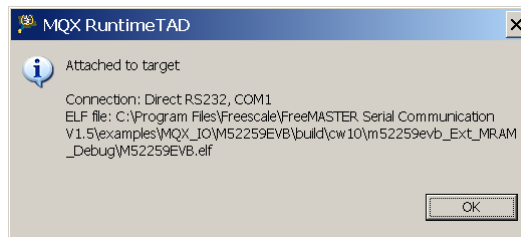
6. If the connection is successfully established, CodeWarrior IDE will be switched to the Debug perspective and *Task summary* view will be opened. You should be able to see the MQX tasks running on the connected target.

Task Name	Task ID	TD	Priority	State	Task Error Code
⊞ main	0x10001	0x2000140c	9	Time delay blocked	OK (0x0000)
⊞ FreeMASTER	0x10002	0x20001ab8	10	Active	OK (0x0000)

7. You can now use the *MQX* or *RTCS* menu to open the various TAD views. Note that availability of individual menu items depends on the features used by the MQX application.
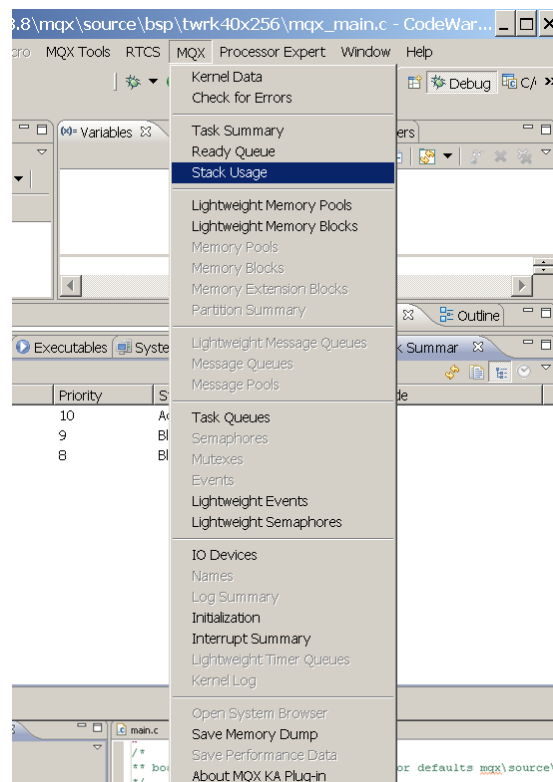

Note: Depending on the FreeMASTER communication task priority and other processes running in the target application, you may experience communication time-out errors. The screens may then display errors or incomplete data. Use the refresh button in the TAD view to try to reload the data.

Getting Started with Runtime TAD Tool

3.3 Using BDM or JTAG Communication Mode

The RuntimeTAD can use any FreeMASTER communication plug-ins installed on your computer to connect to the target. There are BDM and JTAG communication plug-ins which do not require any driver to be running in the MQX application. This section describes use of Runtime TAD with FreeMASTER BDM plug-in which enables to read target CPU memory directly over a debugger interface cable.

Follow the steps below to connect Runtime TAD using a FreeMASTER OSBDM communication plug-in mode. Depending on the target CPU and debug cable options, you can also use other BDM or JTAG plug-in to communicate.

1. Use the CodeWarrior 10 IDE to build and run any MQX application on your target board.

2. Terminate the debugger session. While keeping the application running and the OSBDM/OSJTAG USB cable connected.

3. Go to MQX Tools menu and select *Attach to target…*
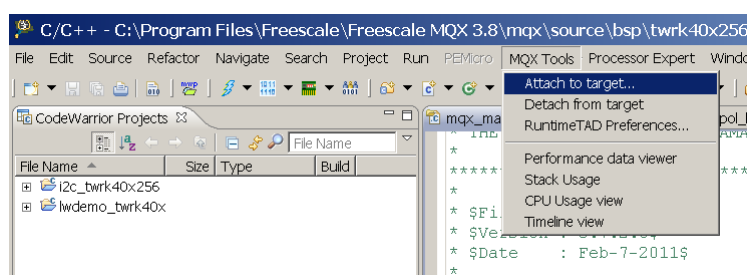

4. *In the Attach to target* dialog, click the *Plug-in Module* mode and select the OSBDM Communication Plug-in from the list.
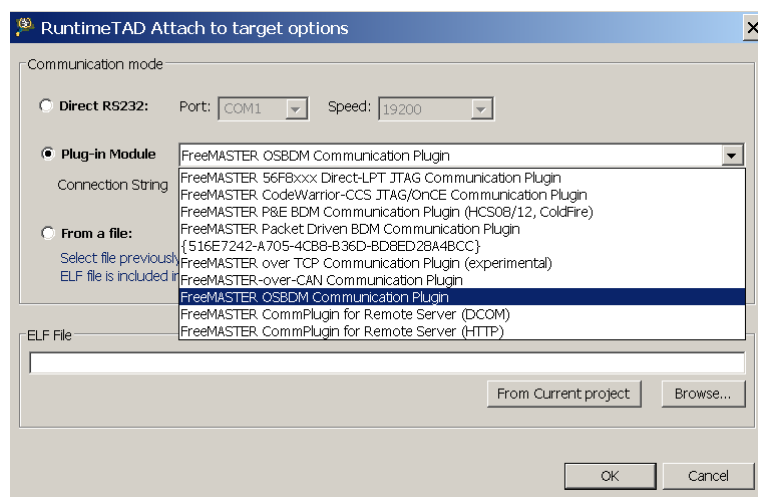

5. Click the *Configure…* button below the drop down list box to configure the selected communication plug-in. A Configuration dialog box appears. For the OSBDM plug-in, the configuration dialog should look like the one in the picture below:
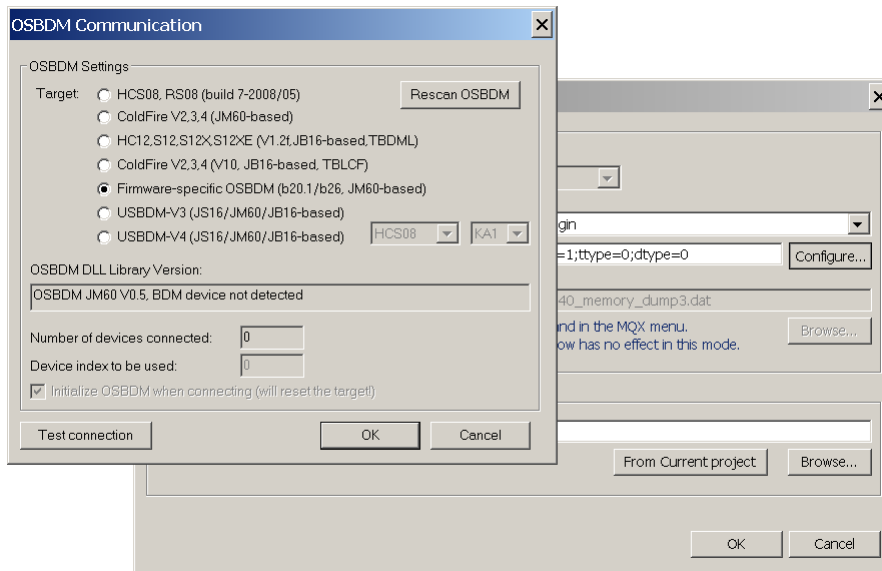
6. Set up the communications options as shown in the picture. Use the Test Connection button to verify that the communication with the target CPU can be established without problems.

7. Click OK to close the dialog. The options get transformed into so-called Connection string, which is displayed below the list of available plug-ins. You can also edit the connection string directly.

8. In the *ELF File* area select the ELF file which corresponds to the application running on the target board. You can either browse to this file using the *Browse…* button or click the *From current project* button to use the output file of the project currently selected in CodeWarrior IDE.
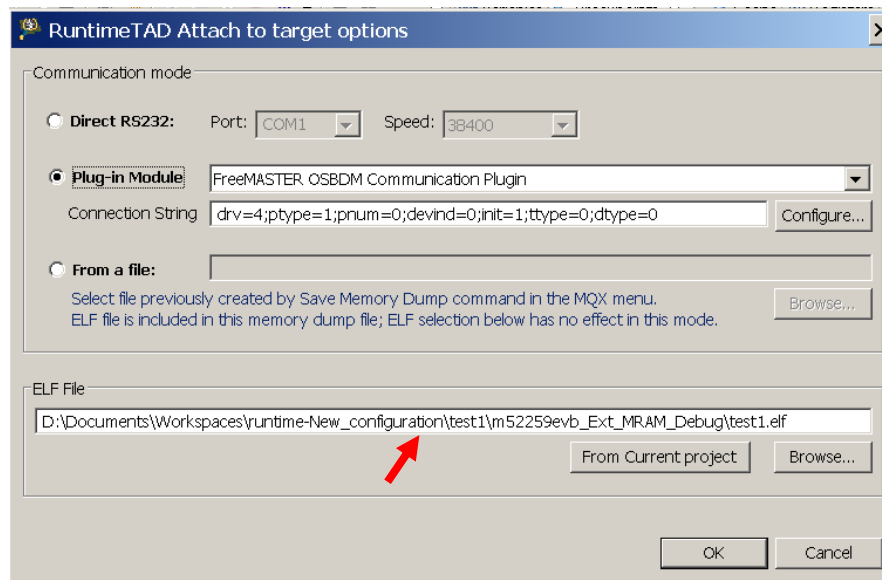

9. Click OK to close the dialog and connect to the target.

8. The Runtime TAD will attempt to connect to the target application using the selected communication plug-in and will inform you about the result.

Getting Started with Runtime TAD Tool

 or
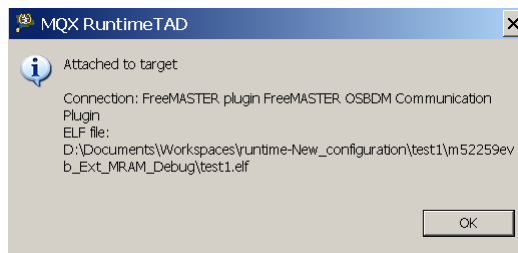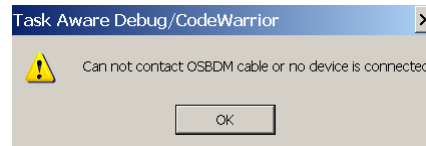
9. If the connection is established successfully the CodeWarrior IDE will be switched to the *Debug* perspective and *Task summary* view will be opened. You should be able to see the MQX tasks running on the connected target.
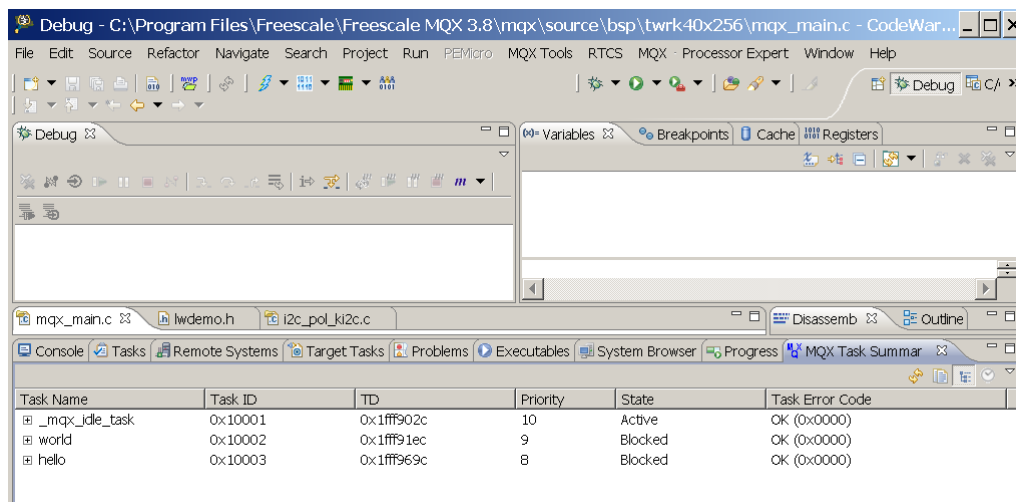
10. You can now use the *MQX* or *RTCS* menu to open the various TAD views. Note that availability of individual menu items depends on the features used by the MQX application.

3.4 Offline Memory Dump Analysis

The Runtime TAD can be used to analyze memory dump file of an MQX application saved from a common MQX TAD debugging session. In the MQX TAD, use the *Save Memory Dump* command which is available in the *MQX* menu. The output (.dat) file can be then selected as an input for Runtime TAD. The MQX TAD takes care to save all context information about the MQX application as well as a copy of the whole application ELF file. This makes the data file the only input needed for the Runtime TAD to display data.

**Note:** the memory dump file can also be saved from development environments other than CodeWarrior and can be analyzed in the CodeWarrior-based Runtime TAD.

Follow the steps below to create memory dump file with MQX TAD:

1. Start a debugger session in a CodeWarrior or other IDE which is supported by MQX TAD. The TAD must be enabled for the debugged application.

2. In the MQX menu select the *Save Memory Dump* command.
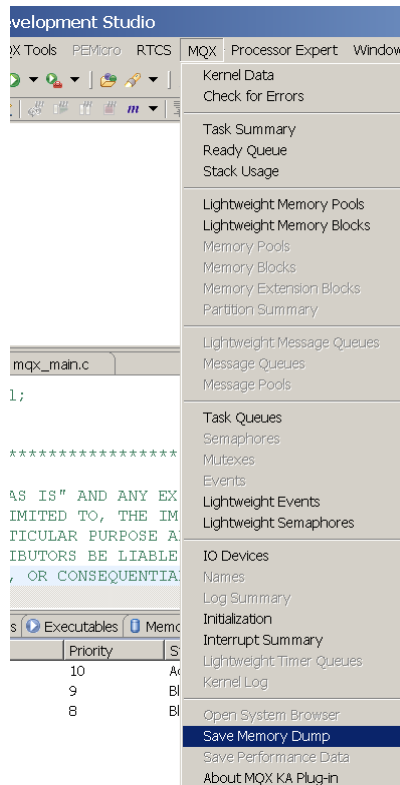
3. In the *Save Memory Dump As* dialog, select the location and name of the dump file and click the *Save* button.

4. The memory dump file will be created. You can now analyze it with Runtime TAD.
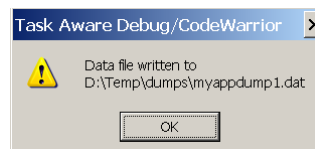
To analyze memory dump file with the Runtime TAD:

1. With CodeWarrior 10.x and MQX 3.8.1 (or later) installed, run the CodeWarrior IDE.

2. Go to MQX Tools menu and select *Attach to target…*

Getting Started with Runtime TAD Tool
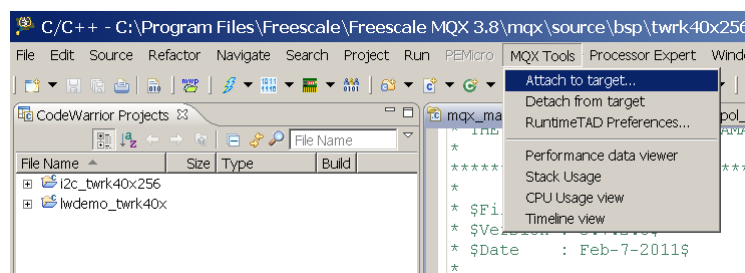
3. In the *Attach to target* dialog, select the *From a file* option and click the *Browse…* button to select the input file.
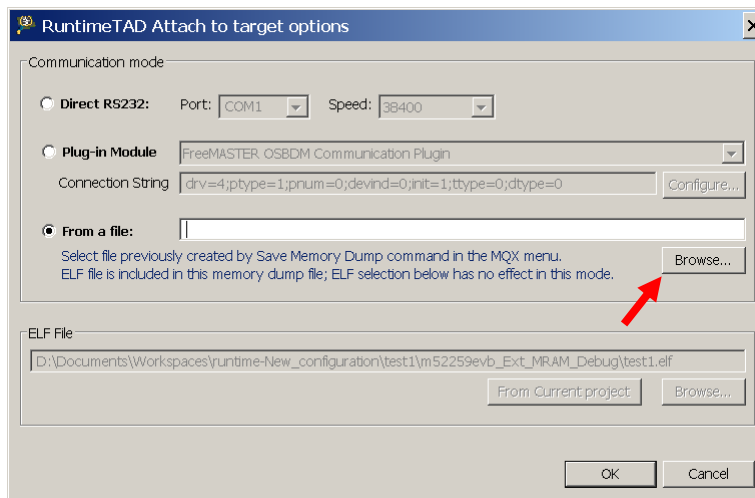
4. Locate the memory dump file you wish to analyze and click *Open*.
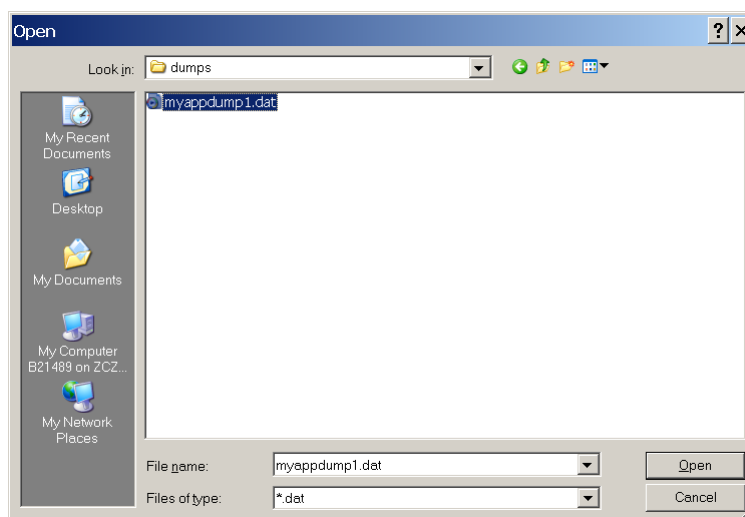
5. Click the *OK* button to attach the Runtime TAD to the selected file.

   Note: You do not need to select ELF file in this communication mode. The ELF file is part of the memory dump file.
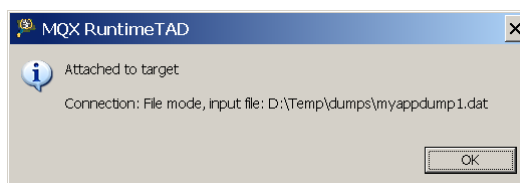
6. If the selected input file is successfully loaded the CodeWarrior IDE will be switched to the *Debug* perspective and *Task summary* view will be opened. You should be able to see the MQX tasks which were running at the time of saving the memory dump file.
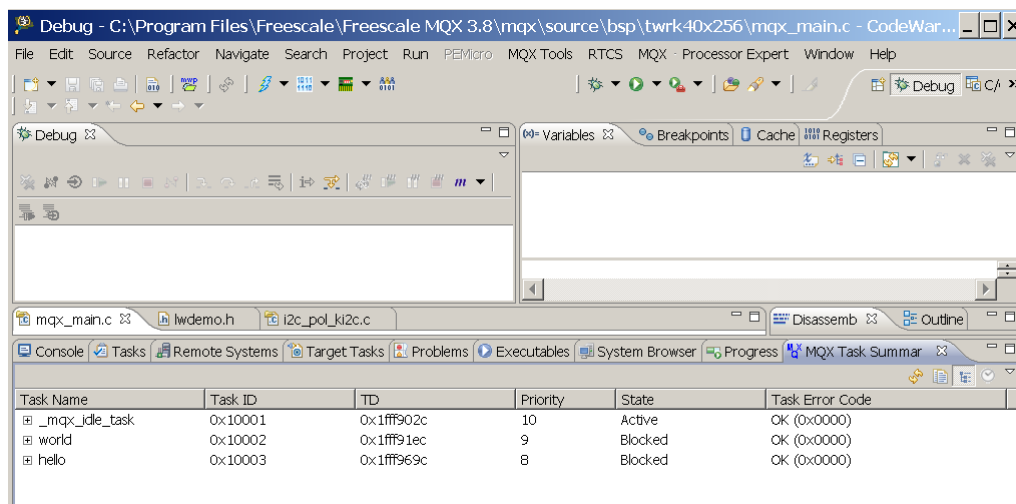
7. You can now use the *MQX* or *RTCS* menu to open the various TAD views. Note that availability of individual menu items depends on the features used by the MQX application.

   Note: Keep in mind that what you are viewing is not "live" picture of a running application. Rather it is a "snapshot" of the application at the time the memory dump was saved.
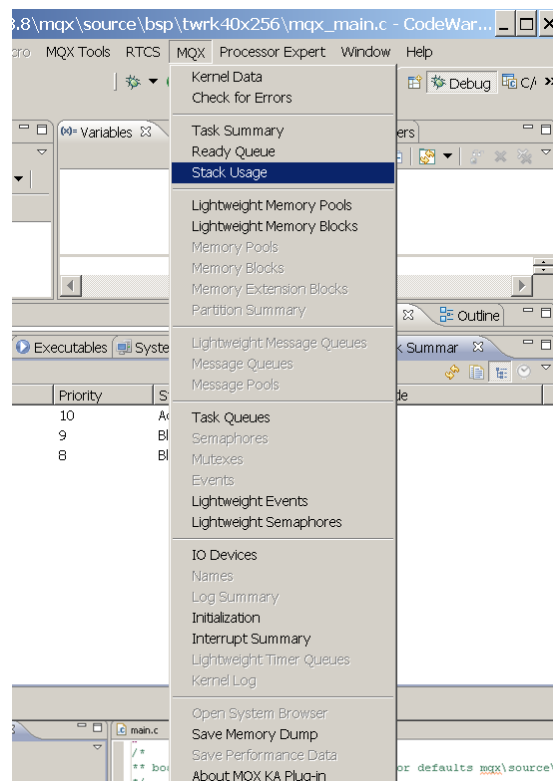
4 Additional Runtime TAD Features

4.1 Automatic Refresh of TAD views

The Runtime TAD can be set to automatically request new data and refresh selected views periodically. This may be useful in situations where manual refresh using the *Refresh* button (available in each view) is not comfortable. Note that in standard MQX TAD all TAD views are automatically refreshed whenever the debugged application is suspended or hits a breakpoint. There is no such event which would naturally refresh the views in the Runtime TAD which is why the periodic refresh may be helpful.

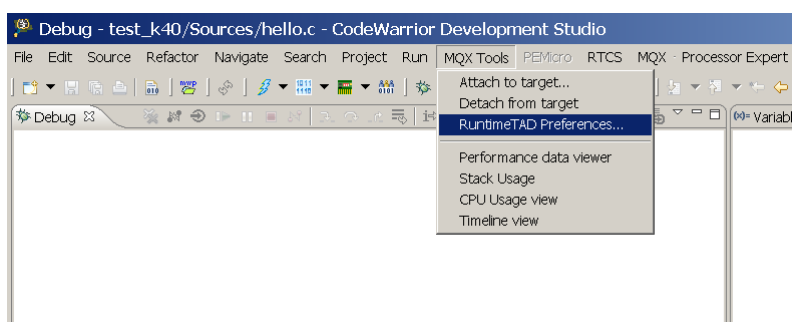The automatic timer-based refresh can be enabled or disabled independently for each view. However, the refresh period is set globally in the *Runtime TAD Preferences* dialog.

Follow the steps below to enable automatic refresh of a view:

1. In the *MQX Tools* menu select *Runtime TAD Preferences…* command

2. In the *RuntimeTAD Preferences* window, check the *Timer enabled* box and enter the refresh period in seconds into the *Timer interval* text box.

    Note: setting up the timer here does not enable periodic refresh of TAD views yet. Each view has its own enable button which needs to be used to actually enable the refreshing.

    Also be aware that depending of the communication speed and TAD view type, reading a data can take long time. Enabling periodic refreshing for several views or choosing short refresh period may degrade overall responsiveness of Runtime TAD GUI.

Getting Started with Runtime TAD Tool
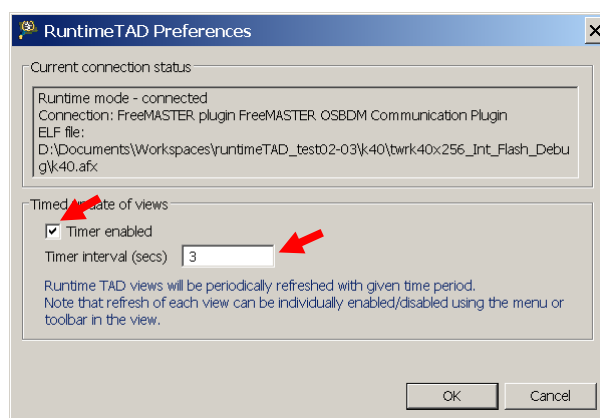
3. In the TAD view to be periodically updated click the *Enable/Disable* button in the view's toolbar or click the *Enable timed refresh* menu item in the view's local menu.
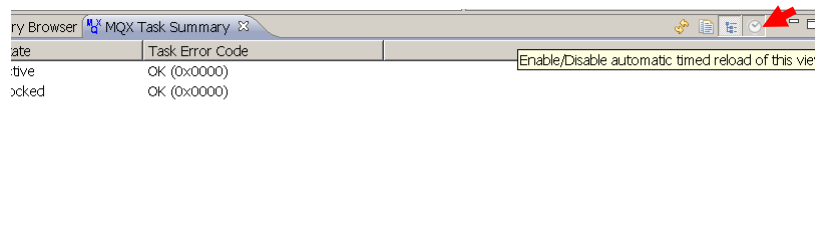
or

4.2 Using Runtime TAD and Standard MQX TAD Together

The MQX Runtime TAD tool uses the same views and tabs in the CodeWarrior IDE as the standard MQX TAD does to display data. Actually the Runtime TAD can be considered an extension of the MQX TAD that displays MQX data obtained from a source other than the debugger engine.

This way of implementation presents a limitation that the Runtime TAD and MQX TAD cannot be used at the same time. The TAD views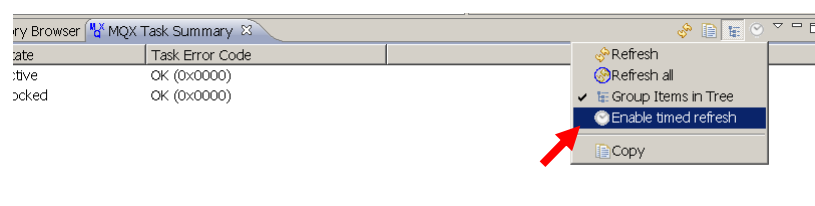 in CodeWarrior IDE can display information either for the TAD or for Runtime TAD. Displaying the standard debugger TAD data is referred to as "**Debugger mode**"; displaying data for Runtime TAD is referred to as "**Runtime mode**".

There are two scenarios in which the standard TAD and Runtime TAD come into conflict:

1. If you start Runtime TAD session while having a debugger session active.

2. If you start a debugger session while connected to the target with Runtime TAD.

4.2.1 Starting Runtime TAD Session while Debugging

If you attempt to use Runtime TAD while a debugger session is active, you get the following message:
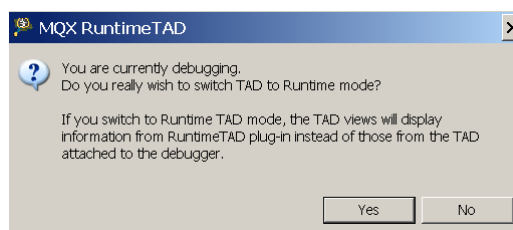
There are two options how to proceed:

1. **"Yes" to Switch to Runtime TAD mode**. In this case the TAD views in the CodeWarrior IDE will be reset and the Runtime TAD will attempt to attach to target application as

Getting Started with Runtime TAD Tool

configured in the *Attach to target* dialog. The debugger session in CodeWarrior is not affected by this choice, only the TAD views are affected. The debugger remains connected to the target.

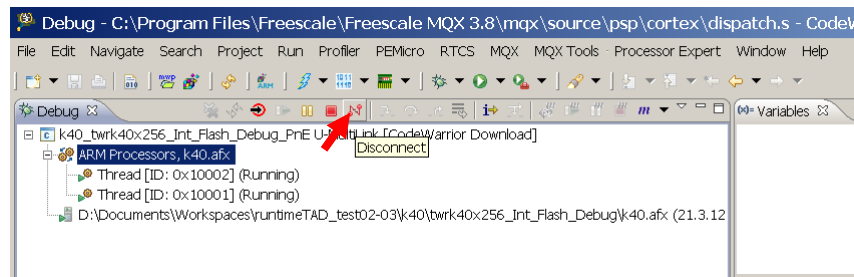If you wish to use the same communication interface in the Runtime TAD as the one used for debugging (e.g. OSBDM), you need to disconnect the debugger first using the *Disconnect* button in CodeWarrior IDE:

2. **"No" to Stay in the Debugger mode.** This option cancels the *Attach to target* command for the Runtime TAD. The IDE stays in the debugger mode and TAD views continue to display information obtained through a debugger engine as before.

4.2.2 Starting Debugger while Runtime TAD is Connected

If you attempt to start a debugger session in CodeWarrior IDE at the time when Runtime TAD is attached to target, the following message appears:
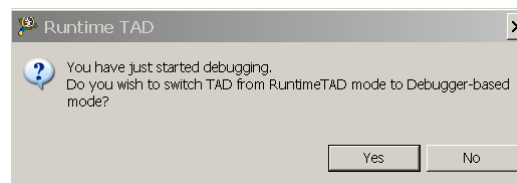
- **"Yes" to Leave Runtime TAD mode**. In this case, the Runtime TAD will be disconnected from the target and the TAD views in CodeWarrior IDE will switch to display information obtained from standard TAD.

- **"No" to Stay in Runtime TAD mode**. In this case the debugger session will start normally, but the TAD views in CodeWarrior IDE will not display information from the debugger. The views will continue to display information from the Runtime TAD. This situation can lead to confusion, so you may want to avoid it.

Note: In case you use the same interface (e.g. OSBDM) in Runtime TAD as is required for debugging, the CodeWarrior debugger fails to open the interface and displays an error message similar to the one shown in the picture below:

Getting Started with Runtime TAD Tool
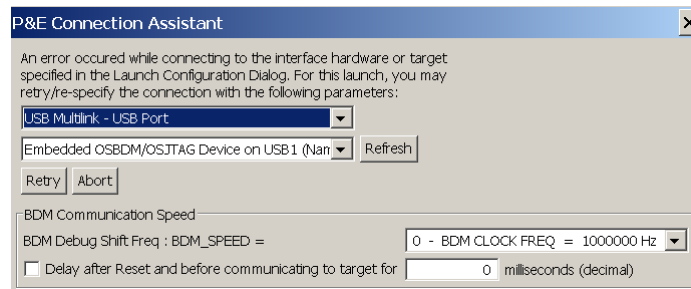
Pressing the *Abort* button cancels the debugger session and returns to the Runtime TAD mode.

5 Troubleshooting

Issue: Runtime TAD connection cannot be established

Possible Cause:

- Target board is not connected or powered up

- Communication plug-in is not configured properly

- Communication interface is opened by another program (e.g. the CodeWarrior debugger)

Recommendation:

- In case an OSBDM or other "debugger cable" plug-in is selected for Runtime TAD connection, check the cable between target board and PC. Unplug the USB cable and plug it again. Try another port on the host PC.

- Check the communication settings in the Attach to target dialog. Use the Test connection button in the configuration dialog for selected communication plug-in (if available) to verify that the connection works.

- Make sure you are not connected to the target via the same interface in another program (e.g. debugging the application in CodeWarrior or using a standalone FreeMASTER tool).

- Try to use the standalone FreeMASTER tool to connect to target outside of the CodeWarrior IDE.

Issue: *Task summary* view and MQX Menu empty or disabled

Possible Cause:

- Wrong ELF file selected in the *Attach to target* dialog.

- MQX application is not properly loaded in the target. The application needs to be running on target.

Recommendation:

- Check the ELF file selected in the *Attach to target* dialog. The ELF must match the executable running on target.

- Use the *Check for errors* command in MQX menu to display information about TAD plug-in status. See if the *_mqx_kernel_data* variable and other important variables have been located in the ELF file and values loaded from target.

6 Summary – What You Should Know

6.1 What is Runtime TAD?

The MQX Runtime TAD is a tool which allows information screening about MQX OS from a running application or from a memory dump file. Unlike the "standard" MQX TAD which requires the application to be connected to a debugger and suspended, the Runtime TAD can display information from an application while it is running. This functionality is similar to what FreeMASTER tool already provides for any application, but the Runtime TAD is also aware of the MQX data structures. It displays the OS objects, such as tasks, memory blocks, semaphores, etc. The Runtime TAD also allows displaying MQX information from a memory dump file, which can be saved from standard TAD while debugging. This enables the analysis with Runtime TAD to be done at later time or on another computer.

Important Note: Both and advantage and a limitation of the Runtime TAD use is that it runs asynchronously and independently on the target application. It is a limitation in a way that the consistency of data memory cannot be granted as it may be changed at the moment of being read. It is recommended to refresh Runtime TAD screens when the application runs in an "idle" mode without intensive dynamic memory or MQX objects manipulations. Try to avoid cases when the Runtime TAD screens are refreshed during memory allocations, task creation/destruction, semaphore or other synchronization object creation/destruction performed by the target application.

6.2 Where can I find Runtime TAD?

The Runtime TAD tool is installed along with MQX 3.8.1 or later and it is only available for the CodeWarrior IDE version 10.2 or later.

The Runtime TAD uses the same views as the "standard" MQX TAD plug-in. You can access the Runtime TAD from the *MQX Tools* menu in the CodeWarrior IDE. The views are available in the in the *MQX* and *RTCS* menus.

6.3 How is the Runtime TAD connected to the application?

There are three options for connecting to the target application, all described in detail in section 3.1.2 above:

1. **Serial line** – the native communication over RS232 line.
2. **FreeMASTER communication plug-ins** – CAN, BDM, JTAG or a custom communication lines supported by the FreeMASTER tool.
3. **File mode** – offline memory dump file analysis.

6.4 Do I need to modify my application to be able to analyze it with Runtime TAD?

This depends on the communication mode which is used for connecting to the application. In general, the same modifications are required as if you would decide to analyze the application with FreeMASTER tool:

- For Serial communication mode or Packet-driven BDM modes there must be a task running in the target application which communicates with Runtime TAD. The driver code is available in the *FreeMASTER Serial Driver* package available on the Freescale web site.

- For most direct BDM and JTAG interfaces there is no need for any change in the application code.

- For analyzing the application offline from a memory dump file there is no need for changes in the application. The memory dump can be saved for any application which is debugged with the standard MQX TAD plug-in enabled.

Getting Started with Runtime TAD Tool