

Getting Started with Performance Tool for Freescale MQX™ RTOS

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	4.0.2
DESCRIPTION:	Getting Started with Performance Tool for 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, the Freescale logo, and CodeWarrior 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.
© 2008-2013 Freescale Semiconductor, Inc.



Table of Contents

Getting Started with Performance Tool for Freescale MQX™ RTOS.....	i
1 Read Me First	2
2 What is the MQX Performance Tool?	2
3 Steps to use the MQX Performance Tool	4
3.1 Configuring the MQX Kernel and Application.....	4
3.2 Creating the Kernel Log in Application.....	6
3.3 Working with Timeline View.....	7
3.4 Working with Stack Usage View	9
3.5 Working with CPU Usage View	10
3.6 Working with Performance Data Viewer	10
3.7 Continuing the Debugger Session	11
4 Summary – What You Should Know.....	12
4.1 Performance Data and MQX Kernel Log	12
4.2 Enabling Kernel Log in MQX Kernel	12
4.3 Logging the Information in Your Application	12
4.4 Using Performance Tool.....	12

1 Read Me First

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

The MQX Performance Tool is a set of small utilities and features (views) which can be used during a MQX debugger session in the CodeWarrior 10.x. The utilities are available in the *MQX Tools* menu in the CodeWarrior IDE.

2 What is the MQX Performance Tool?

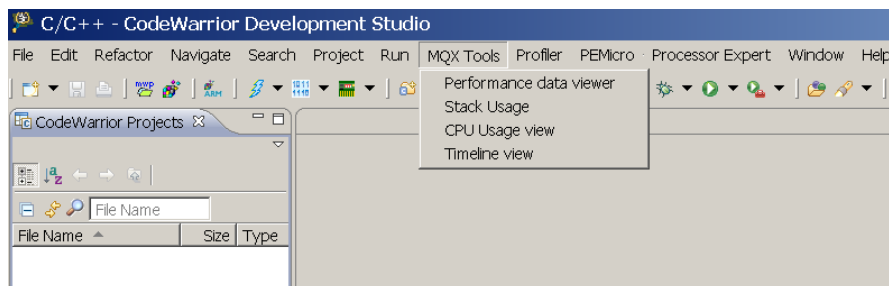
The MQX Performance Tool is a plug-in for CodeWarrior 10.x IDE which can be used to display various run-time information about MQX application. The information can also be retrieved from performance data file saved during any debugger session from Task-aware Debugger plug-in (TAD).

See more information about TAD in the *Getting Started with Freescale MQX™ RTOS* document distributed with the MQX installation.

The current version of the MQX Performance Tool, distributed with the MQX, includes the following features (or “views” as they are referred in this document):

- Timeline graph of task switches, interrupts, synchronization objects, etc.
- Task Stack view which displays size of stacks and used stack sizes of all tasks.
- CPU Usage view displaying a pie chart of CPU runtime allocation among different tasks.
- Performance Data viewer.

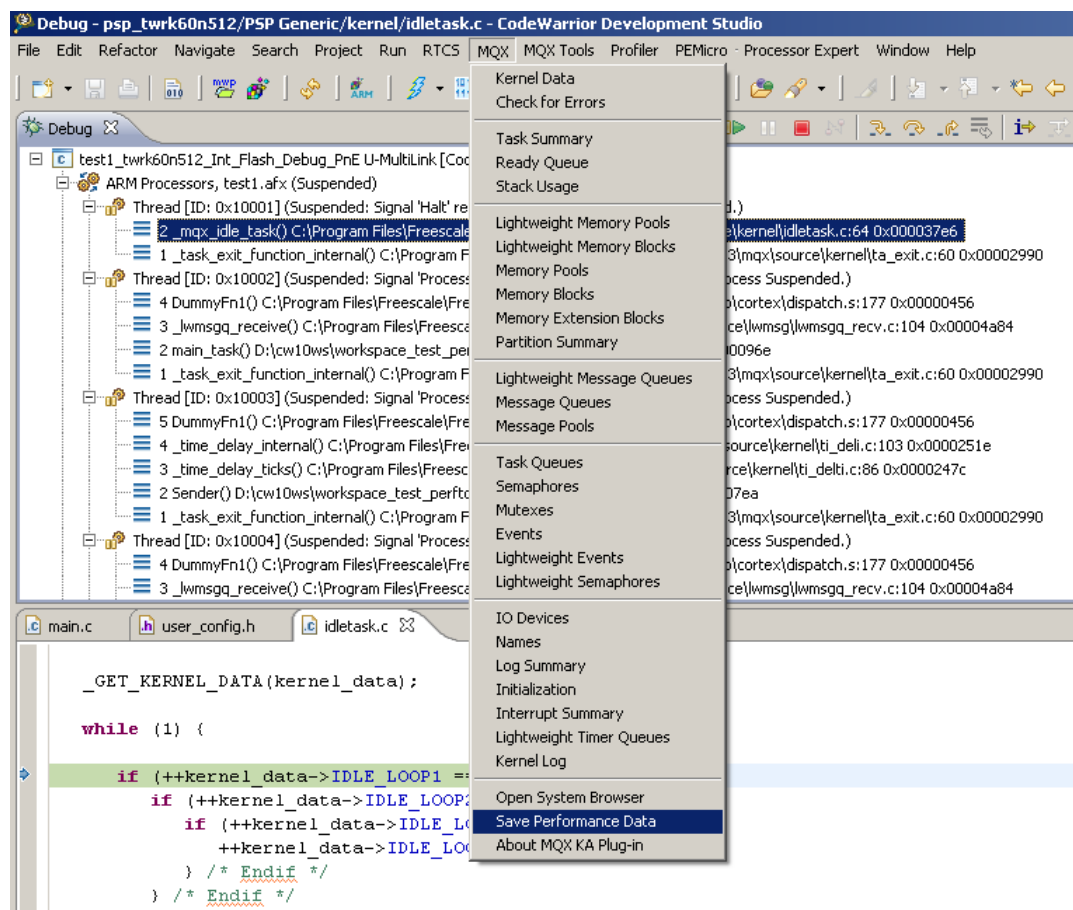
The MQX Performance Tool is available in the MQX Tools menu in the CodeWarrior 10.x IDE:



Each view can be set to Online or Offline mode:

- In the Online mode, the Performance Tool uses data obtained through TAD and debugger connection from the MQX kernel log. **To use the Performance Tool, the MQX application must have kernel logging enabled.** This document describes steps needed to set up the application for using the Performance Tool.

- In the Offline mode, the performance data file saved during a previous debugger session can be analyzed with Performance Tool views. In order to save the performance data file, the application should have the kernel logging enabled. Any time during a debugger session, use the *MQX / Save Performance Data* menu in the CodeWarrior IDE to save the data file for later analysis.



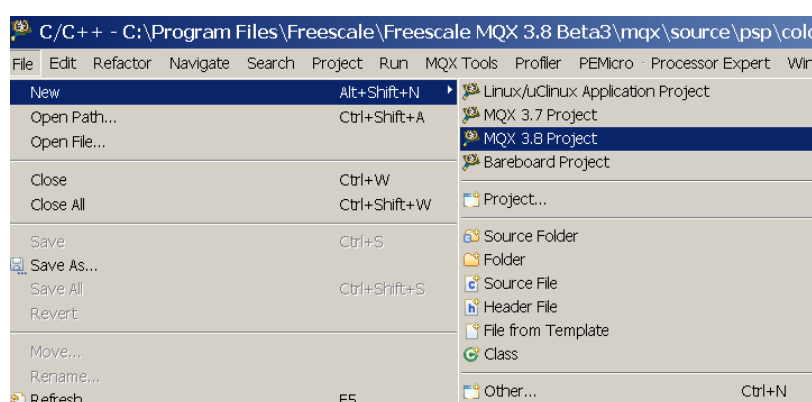
3 Steps to use the MQX Performance Tool

This section describes steps to configure, build, and run your application to make use of the Performance Tool. See the *Getting Started with Freescale MQX™ RTOS* document, for more details about building the MQX kernel and about debugging the MQX applications.

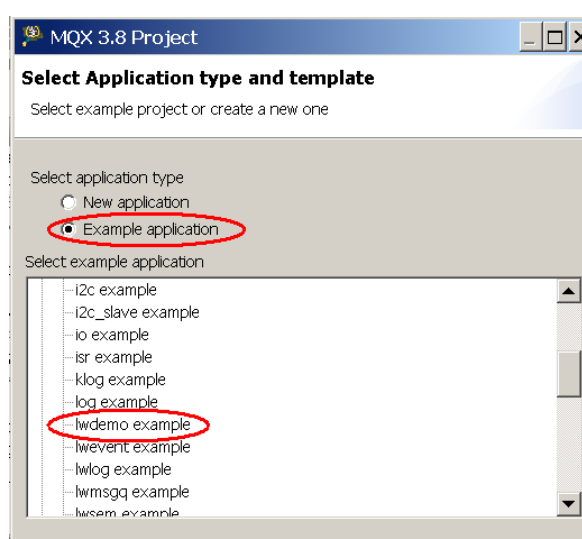
The “lwdemo” example application from MQX distribution is used to demonstrate the required steps. You can, of course, apply similar steps to any other example or your own MQX application developed with CodeWarrior 10.x tools.

3.1 Configuring the MQX Kernel and Application

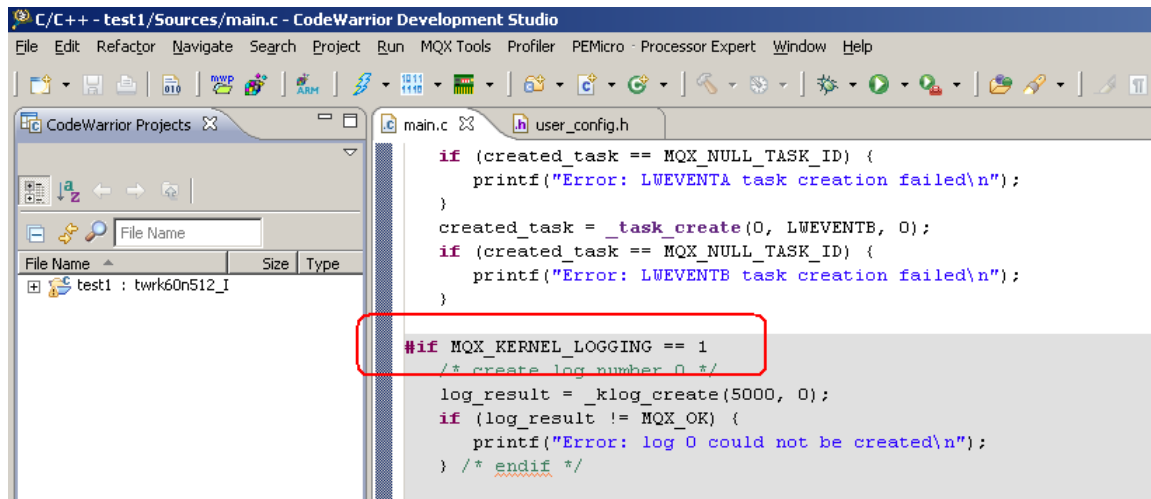
1. Make sure the MQX 3.8 (or later) is properly installed and run the CodeWarrior 10.x IDE.
2. Go to File menu and select *New / MQX 3.8 Project*:



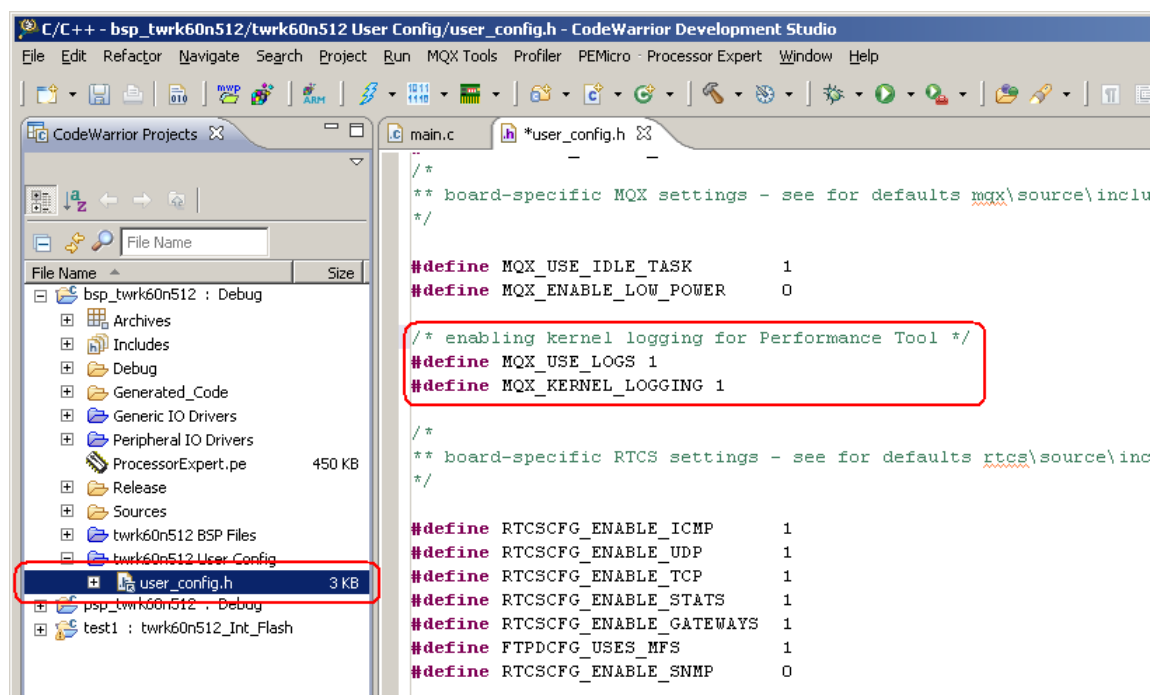
3. Follow the *MQX New Project* wizard steps. Select application name and target platform. Then choose the *Example application* type and select *Basic examples / lwdemo example* in the tree-like list.



4. Finish the wizard. The `lwdemo` example will get imported into your workspace under the name you have specified.
5. Take a look at the `main.c` file and see the section where kernel log is created. You can see the code is compiled conditionally only when logging is enabled in the MQX kernel.



6. By default, the kernel logging is disabled in MQX kernel. It needs to be recompiled with different compile-time configuration.
7. Go to the `<mqx_installation>\mqx\build\cw10` folder and import the PSP and BSP projects for your target platform into the CodeWarrior 10.x workspace. You drag and drop the project folders into the CodeWarrior IDE window to import them.
8. In the BSP project, open the `user_config.h` file. The full path to the file is as follows:
`<mqx_installation>\config\<board>\user_config.h`



9. Add the following lines into the *user_config.h* file:

```
#define MQX_KERNEL_LOGGING 1
#define MQX_USE_LOGS 1
```

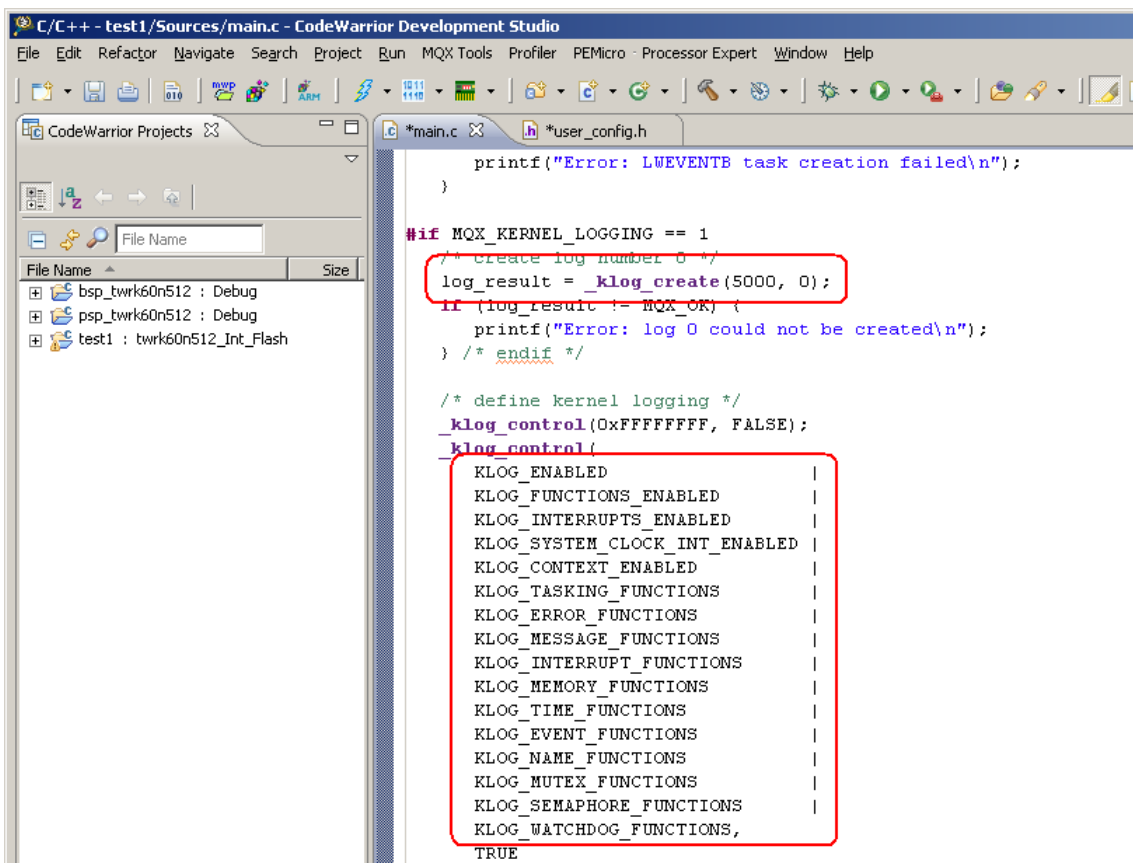
10. Re-build the PSP and BSP projects.

3.2 Creating the Kernel Log in Application

Continue with the following steps:

11. In your imported *lwdemo* project, open the *main.c* file and increase the size of the kernel log being created. This is needed so that the Performance Tool has enough entries to display meaningful information. The default value is set quite low to also ensure that the application runs properly on platforms with small RAM resources.

Note that, in your application, you may also want to disable some of the KLOG events in the kernel log recording.




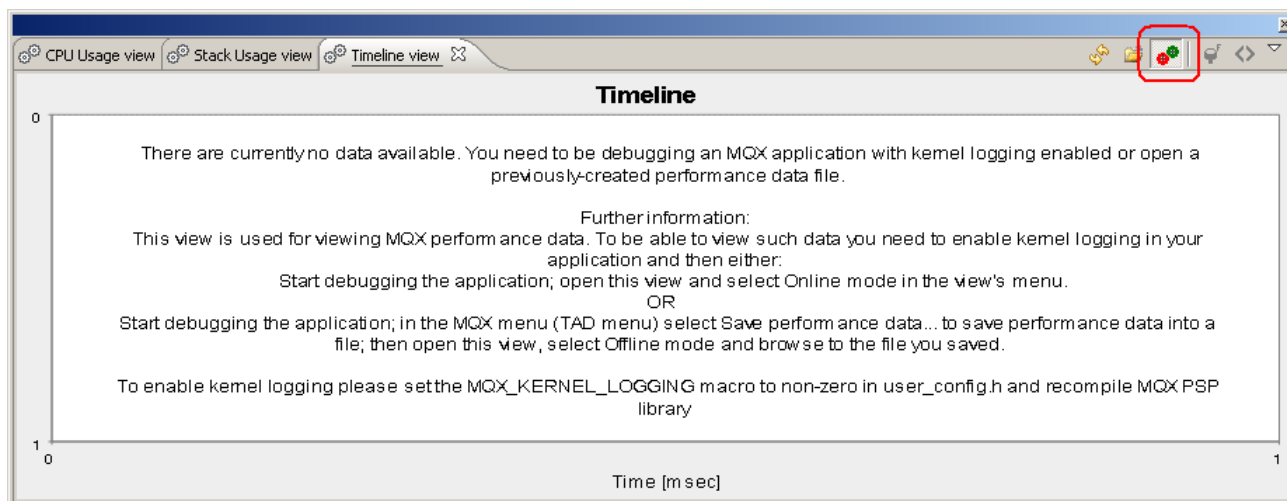
12. Build the application project.
13. Invoke the debugger session and run the application.
14. Suspend it after a few seconds.

3.3 Working with Timeline View

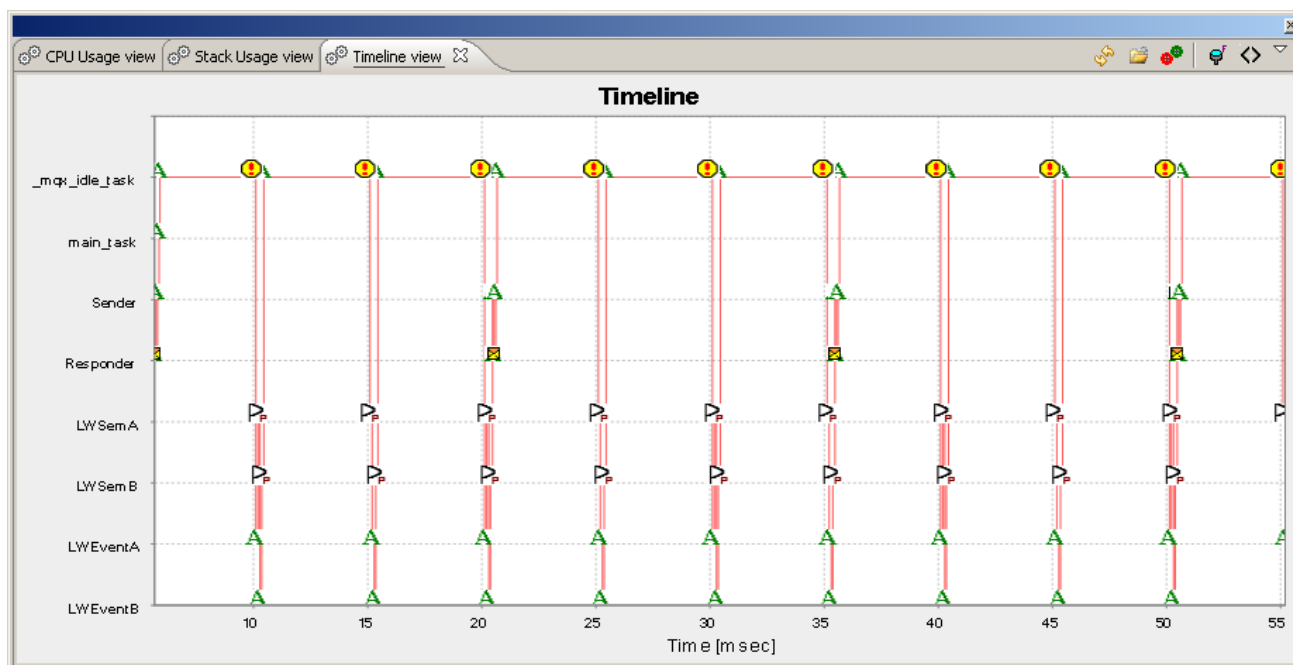
Continue with the following steps:

15. In the MQX Tools menu select *Timeline view*. The timeline opens in an offline mode by default with no data to be displayed.

16. Click the *Toggle offline and online mode* button to switch to online mode. 

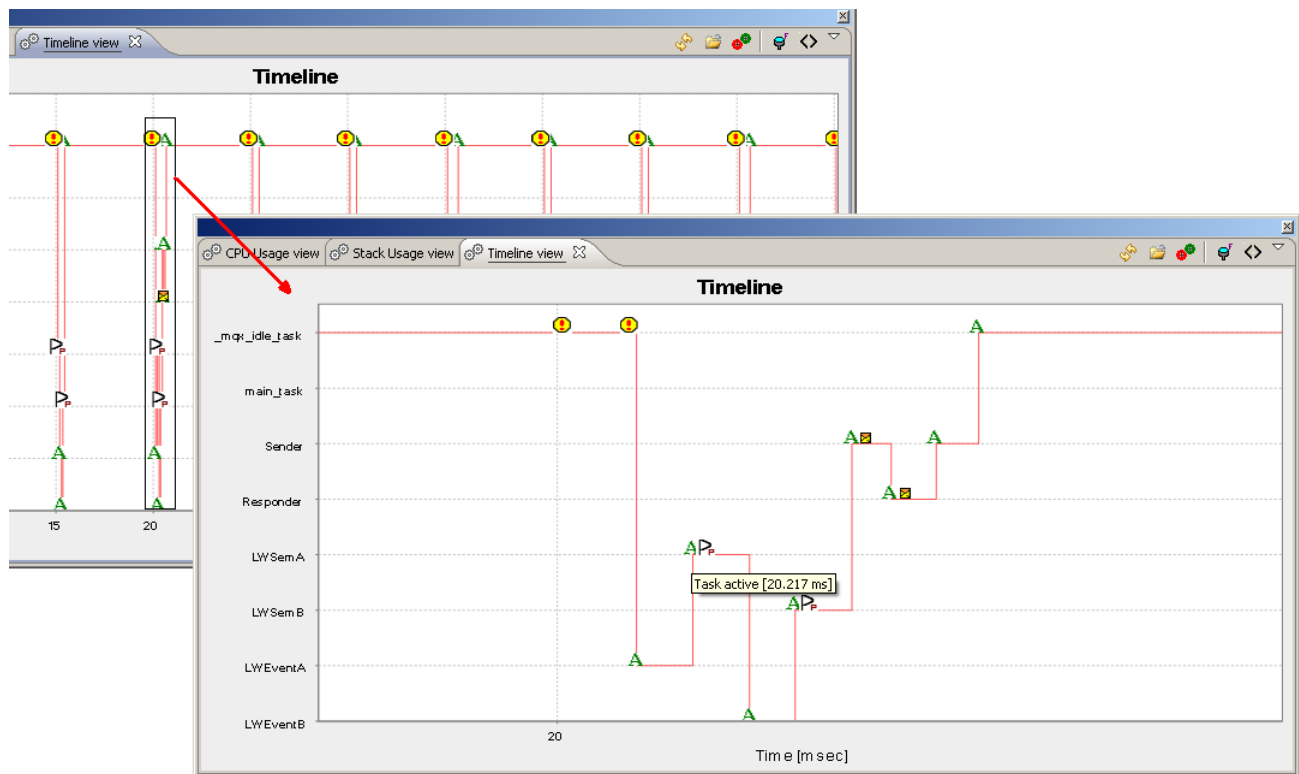



17. The view should display the Timeline graph reconstructed from the Kernel Log records.

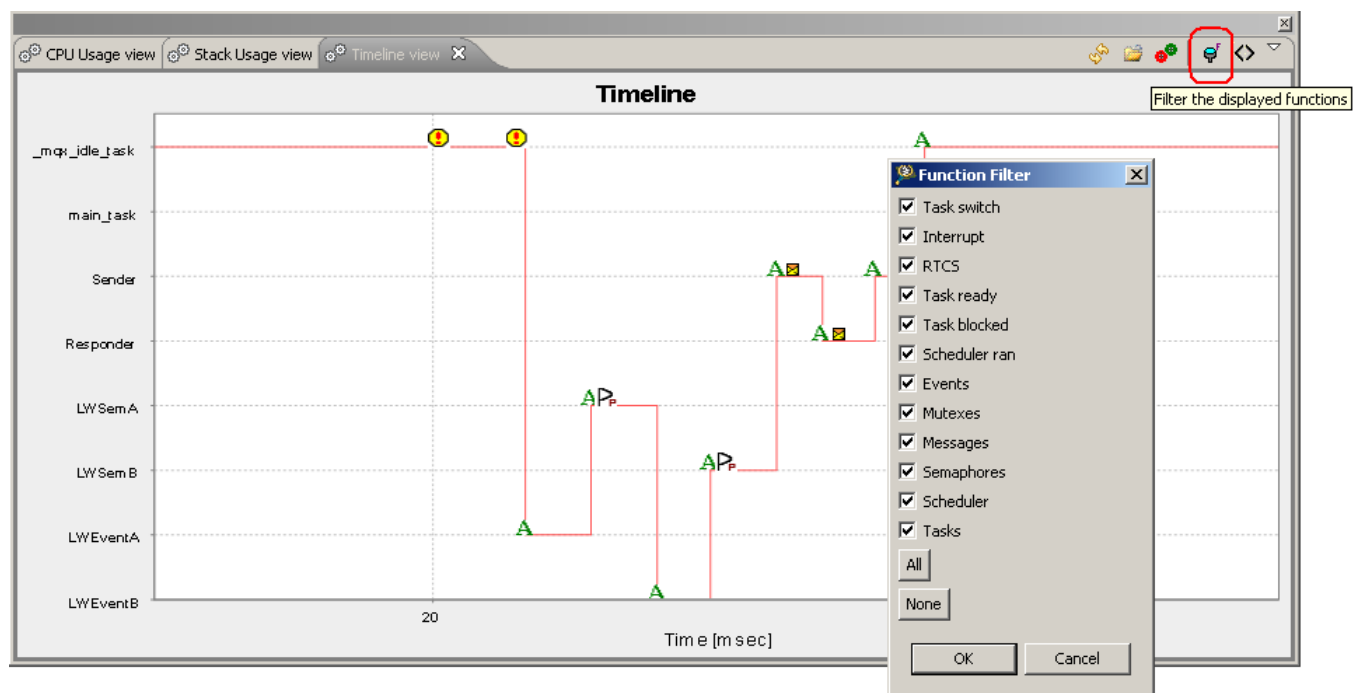


The graph shows task switches in your application. Icons represent various events logged by the MQX kernel such as interrupts, synchronization calls, message sending, etc.

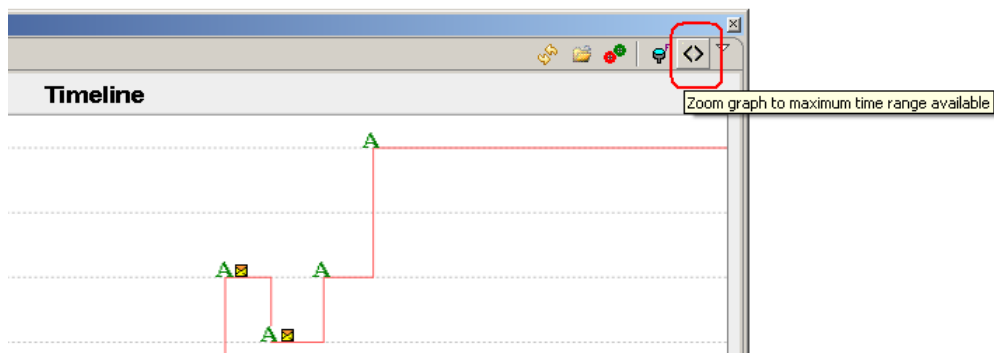
18. Use your mouse and drag an area to zoom it in the graph. Hover the mouse over an icon to display information about the event.



19. You can also filter some events out of the graph by the *Function filter* button in the view's toolbar :

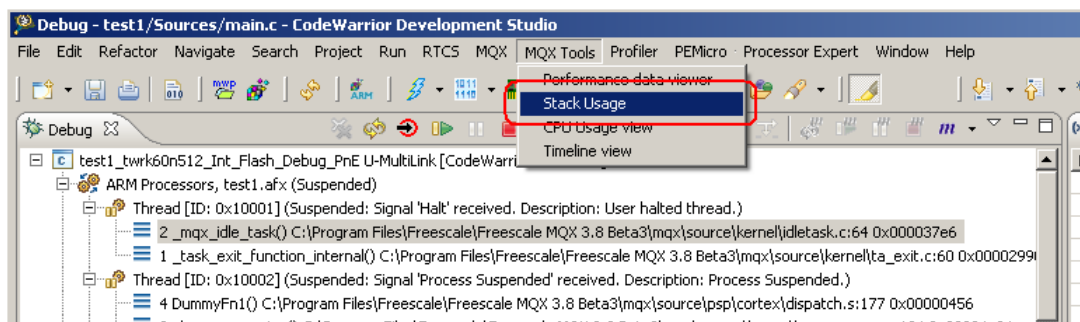


20. To display the whole chart again, click the *Zoom out* button in the view's toolbar <>.

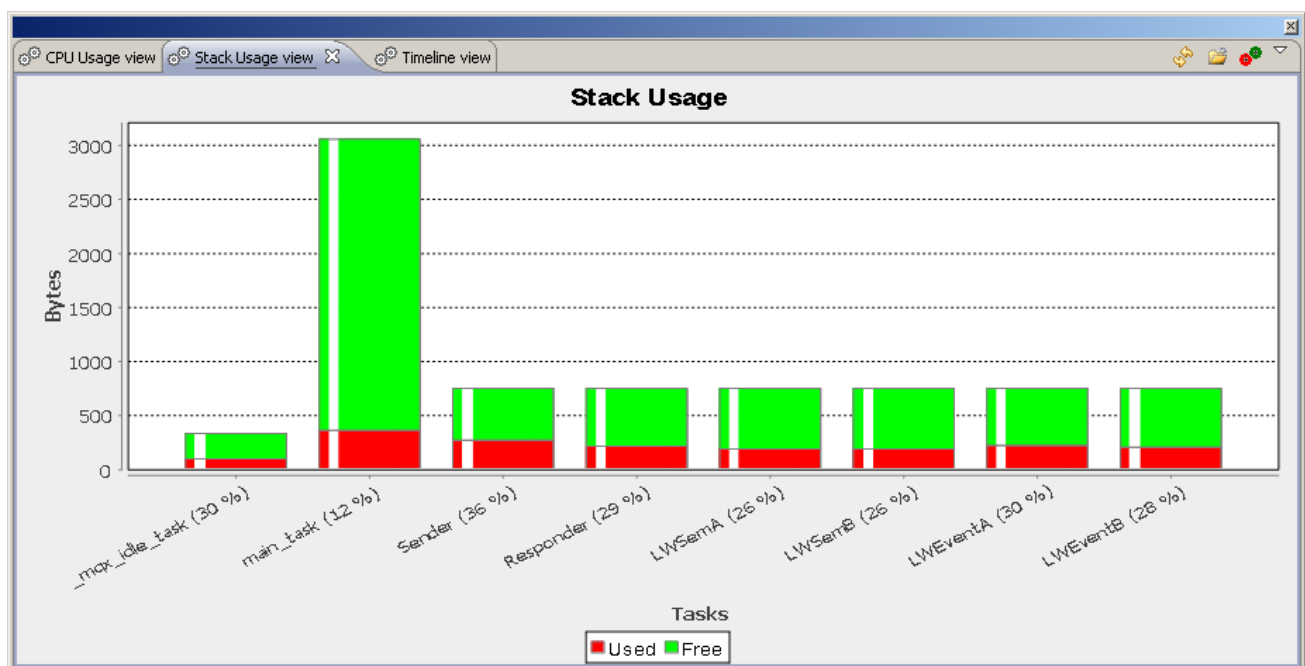


3.4 Working with Stack Usage View

Similarly as the Timeline View, other MQX Performance Tool views can be displayed. Go to the *MQX Tools* menu and select *Stack Usage*.



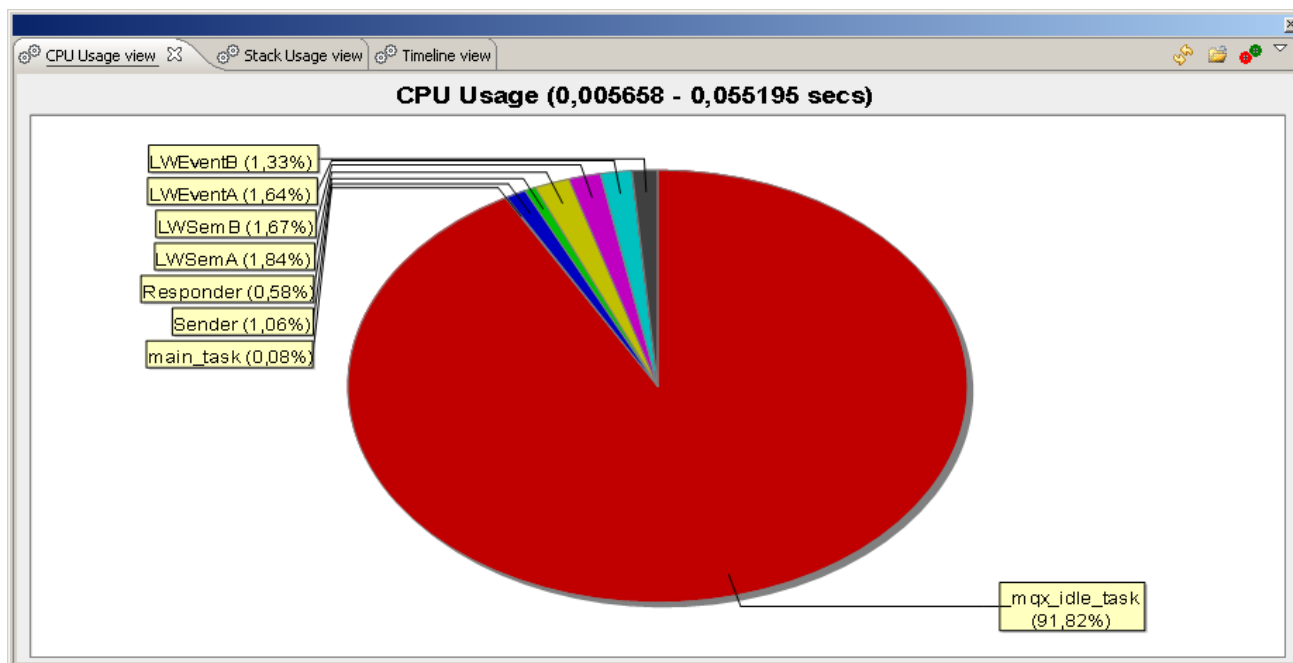
The bar graph displays how much stack is used by each task. Don't forget to click the *Online mode* button to switch to online mode.



3.5 Working with CPU Usage View

With debugger session still open, go the *MQX Tools* menu and select *CPU usage view*. The pie chart shows how different tasks utilize the CPU run time.

Click the *Online mode* button to switch the view to online mode .




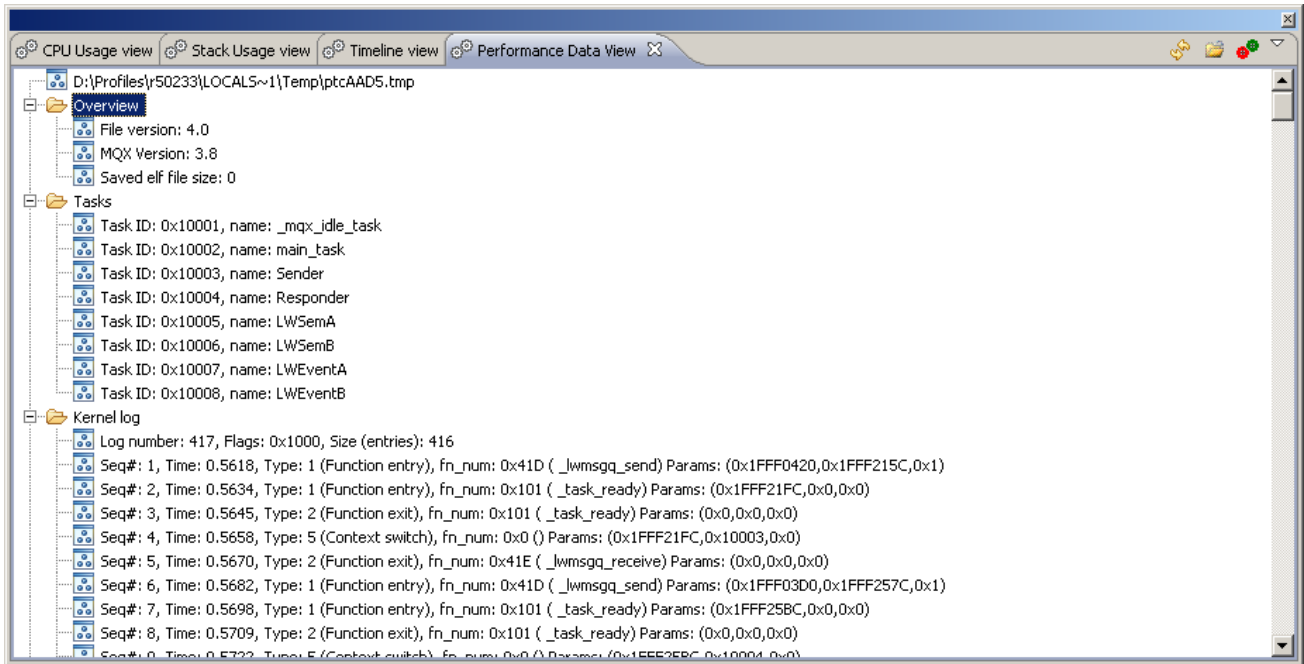
Be aware that this chart is not “cumulative” and is calculated based on the kernel log information only. The chart reflects only a short interval of the whole application runtime.

3.6 Working with Performance Data Viewer

The last view available in the MQX Performance Tool is the Performance Data Viewer.

In an online mode, this file is managed internally by TAD plug-in. You will see a path to dynamically created temporary file in the view heading.

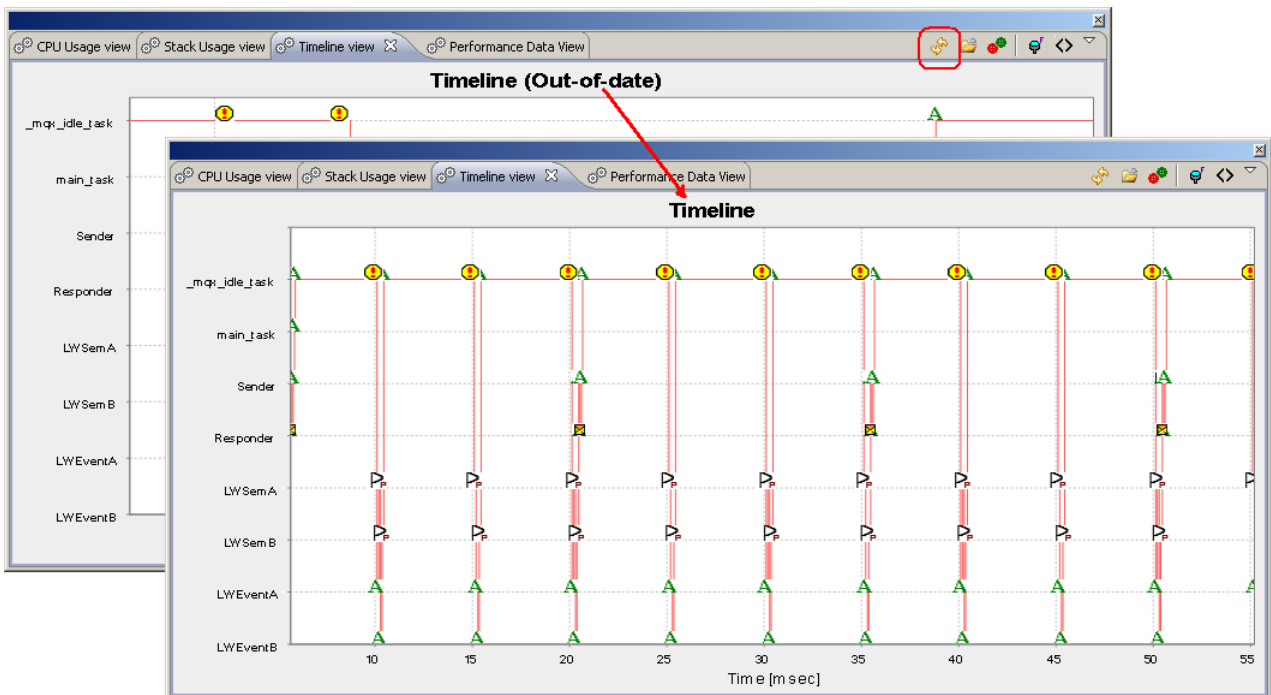
To view data from a file which you have previously saved from the TAD *MQX* menu, click the Toggle offline and online mode button  in the view's toolbar.



3.7 Continuing the Debugger Session

When you are done with analyzing the performance data at a current breakpoint, you can resume the application from a suspended mode in the debugger. Be aware that the Performance Tool views are not automatically refreshed when the target is suspended again. The view's title is updated with "Out-of-date" information.

In the online mode, click the *Refresh* button in view's toolbar  to refresh the view content.



4 Summary – What You Should Know

4.1 Performance Data and MQX Kernel Log

The MQX Performance Tool uses data obtained from MQX Kernel Log to display information about the application such as at what time a task switch occurred, a kernel function was called, etc.

The logging features in the MQX RTOS include Logs and Lightweight logs (LWlogs). The difference between the two is that the records in Logs contain user-defined variable-length data while the LW log records are all fixed in length and structure.

There can be up to 16 logs (index 0 to 15) and up to 15 lightweight logs (index 1 to 15) created in the application. The lightweight log #0 is reserved for the Kernel Log.

4.2 Enabling Kernel Log in MQX Kernel

As described in this document, for an MQX application to use Kernel log, it must be linked with MQX libraries built with `MQX_KERNEL_LOGGING` and `MQX_USE_LOGS` macros set to one. This is not the default setting so, to be able to use the Performance Tool features, you need to rebuild the PSP and BPS libraries.

4.3 Logging the Information in Your Application

The size of the Kernel Log determines how many events can be logged, for example, how many task switches you will be able to see in the Timeline view. The log size is specified by an application code in a call to `_klog_create`.

Besides increasing the log size, it may also be useful to disable logging of unwanted types of events in the call to `_klog_control`. When not configured properly, the log may fill up quickly with unwanted data and you may not be able to see all the tasks switches in the time range you need. In extreme cases, there can be no task switch record at all in the log and the Timeline view will be displayed void.

4.4 Using Performance Tool

The MQX Performance Tool views are accessed via the *MQX tools* menu in CodeWarrior 10.x IDE. The views display the information in one of two modes: Online or Offline.

In the Online mode, the data is obtained from currently debugged application through a temporary file dynamically created by the TAD debugger plug-in. The views are not refreshed automatically any time the application gets suspended. When the view title says the data is out-of-date, press the *Refresh* button in the view's toolbar.

In the Offline mode the data is read from a file as specified by you. This file can be created using the MQX TAD plug-in during a debugger session in menu *MQX / Save Performance Data*.