Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	Freescale MQX 3.8.0 (or later)
DESCRIPTION:	Using IAR Embedded Workbench Tools with Freescale MQX™ RTOS
RELEASE DATE:	Dec 15th, 2011

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

Rev. 02
12/2011

Table of Contents

1 Read Me First

This document describes steps required to configure the IAR Embedded Workbench development tools and use it to build, run and debug applications of the Freescale MQX™ RTOS operating system. Refer to “Getting Started” and other user documentation included within the latest Freescale MQX™ RTOS installation for more details not specifically related to IAR Embedded Workbench tools.

Get the latest Freescale MQX™ RTOS at http://www.freescale.com/mqx.

2 Building the MQX Libraries

2.1 Compile-time Configuration

Major compile-time configuration options are centralized in a single user configuration file located in

```
<install_dir>/config/<board>/user_config.h
```

This user configuration file is included internally by private configuration files in MQX PSP and BSP.

To share configuration settings between different boards, the user_config.h file may include other header files with common settings. The header files may only be located in the same <board> directory or in the “common” directory:

```
<install_dir>/config/common
```

All MQX configuration files are also *indirectly* used by other core components like RTCS, MFS, etc. “Indirectly” means that the MQX PSP and BSP must be build first, which causes the configuration file being copied into the output (lib) directory. The other components then include the configuration file from the `/lib` output directory.

Caution: Until the PSP or BSP libraries are rebuilt, configuration changes made in the `user_config.h` file are not used by any other MQX component. On the other hand, after the PSP and BSP libraries are re-compiled with a new configuration, it is important to recompile the other libraries so the compiled code is consistent with the configuration file. See the next section for more details.

2.1.1 Build Process

After any change to the compile-time user configuration file or MQX kernel source files, the MQX libraries need to be re-built. The build process is similar with all core components:

- The output directory for any MQX library component is *<install_dir>*`/lib/`*<board>.<compiler>/<component>*
- For example the MQX PSP and BSP libraries for the TWR-K60N512 board are copied into the `/lib/twrk60n512.iar/psp` and `/lib/twrk60n512.iar/bsp` directories after successful build process.
- All public header files needed by an application to make use of the library are also copied from internal include folders to the same output directory.
- During PSP or BSP build process, also the `user_config.h` file and other header files from the `config/<board>` and `config/common` directories are copied into the `lib/<board>.iar` output directory.
- Other components like RTCS, MFS, Shell or USB use the copied configuration files only.
- Applications which make use of any MQX library do not need to make any reference to the internal source and include paths of the MQX components. Applications use solely the paths in the `/lib/`*<board>.<compiler>* as the search paths for header files or libraries.

To summarize the points above, there are simple **rules to obey** when re-building the MQX libraries.

- After any change to the `/config/common/user_config.h` file, all MQX libraries should be re-built.
- The PSP and BSP libraries must be build first, before the MFS, RTCS and other libraries.

Important: No changes should be made to header files in the output build directory (`/lib`). The files get overwritten any time the libraries are built.

2.2 Build Configurations

Each IAR project in Freescale MQX™ RTOS contains multiple compiler/linker configurations (so called build „targets“).

Two different types of build targets exist for different compiler optimization settings:

- **Debug** – the compiler optimizations are turned off or set to low. The compiled code is easy to debug but may be less effective and much larger than the Release build. All output libraries (or executables) have **_d** postfix in the file name (e.g. `rtcs_<board>_d.a`).
- **Release** – the compiler optimizations are set to maximum. The compiled code is very hard to debug and should be used for final applications only. There is no postfix in the output file name (e.g. `rtcs_<board>.a`).

Build target name of any MQX application project makes a reference either to **Debug** or **Release** builds of the core libraries. On top of that the target names also specify board memory configuration which gets built. For example:

Devices with internal Flash memory (e.g. TWR-K60N512):

- **Int. Flash Release** – this target is suitable for final application deployment. When programmed to Flash, the application starts immediately after reset. Variables are allocated in internal SRAM memory.
- **Int. Flash Debug** – same as above, only the Debug-compiled libraries are used. This target is suitable for debugging before deployment. On boards without external memory, this is the only target suitable for debugging larger applications.

Boards and devices with internal Flash memory and additional external RAM for data (TWR-K70F120M):

- Int Flash <mem>Data Debug – The name of each target additionally defines a memory used as the default data storage. For example the application built with target named “Int Flash DDRData Debug“ will execute code out of internal Flash memory and will use the DDR memory for data storage.

Boards with external RAM memory:

- **Ext. Ram Debug** – solely for debugging purposes with code located in external RAM memory. Both code and variables are located in this external memory. Application executable is loaded to RAM automatically by the debugger.

Refer to BSP-specific information included in the latest MQX installation for description of build targets specific to particular boards.

2.3 Batch Build in IAR Embedded Workbench IDE

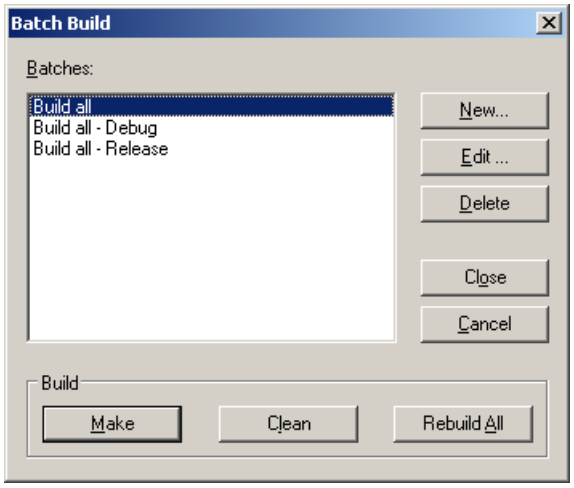
With IAR, the MQX build process can be simplified by using Batch Build feature. For each supported board, there is an IAR Workspace file which includes build projects for all related MQX libraries:

`<install_dir>/config/<board>/iar/build_libs.eww`

The Workspace file contains Batch Build configurations which can be used to build all MQX libraries at once.

- Go to menu “Project / Batch Build” or press the F8 key in the IAR IDE.

- Select Batch configuration to build (refer to next section for more details about build targets)
- Press the “Make” button to start the batch build process.

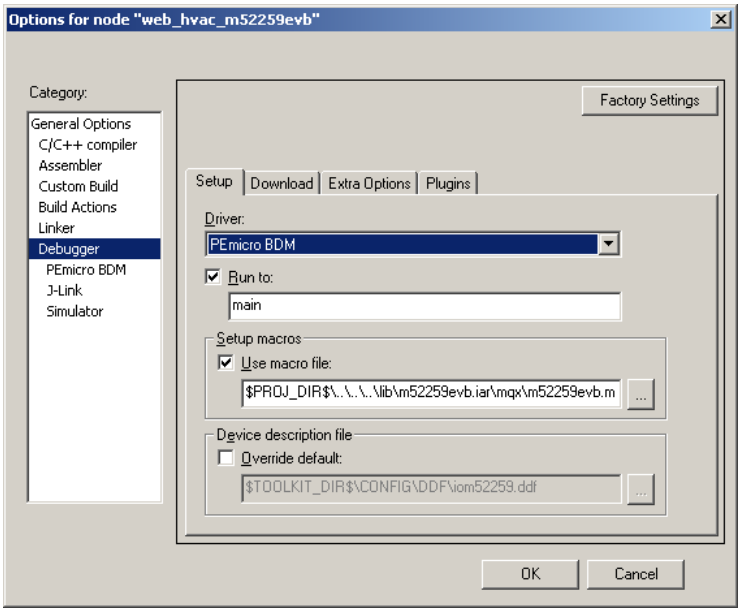


3 MQX Task Aware Debugging

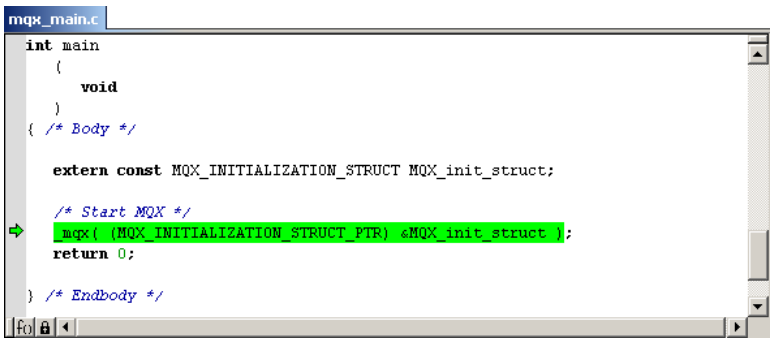
MQX Task Aware Debugging plug-in (TAD) is an optional extension to a debugger tool which enables easy debugging of multi-tasking applications. It helps to visualize internal MQX data structures, task-specific information, I/O device drivers and other MQX context data.

3.1 Debugging MQX Applications in IAR Embedded Workbench

Loading and debugging MQX applications is an easy task with IAR Embedded Workbench tool and it is not different from debugging classic non-OS applications. Make sure the correct debugger interface is selected in the project options and correct processor initialization Macro file is used. MQX installation contains its own processor initialization macro files as a part of the BSP library for each supported processor.



When an MQX application is compiled and linked with all MQX libraries, press the “Download and Debug” button on the toolbar. The application gets executed and stops at the default C language entry point in the *main()* function. Be aware that at this breakpoint, the MQX Operating System is not yet running so use of TAD plugin features (as described in later sections) is limited.

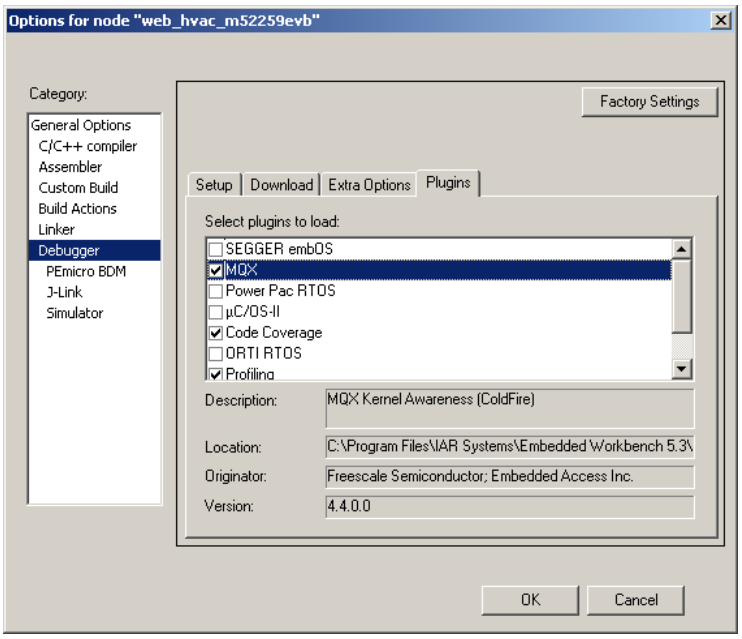


3.2 TAD CSpy Debugger Plug-in

3.2.1 Installing CodeWarrior TAD

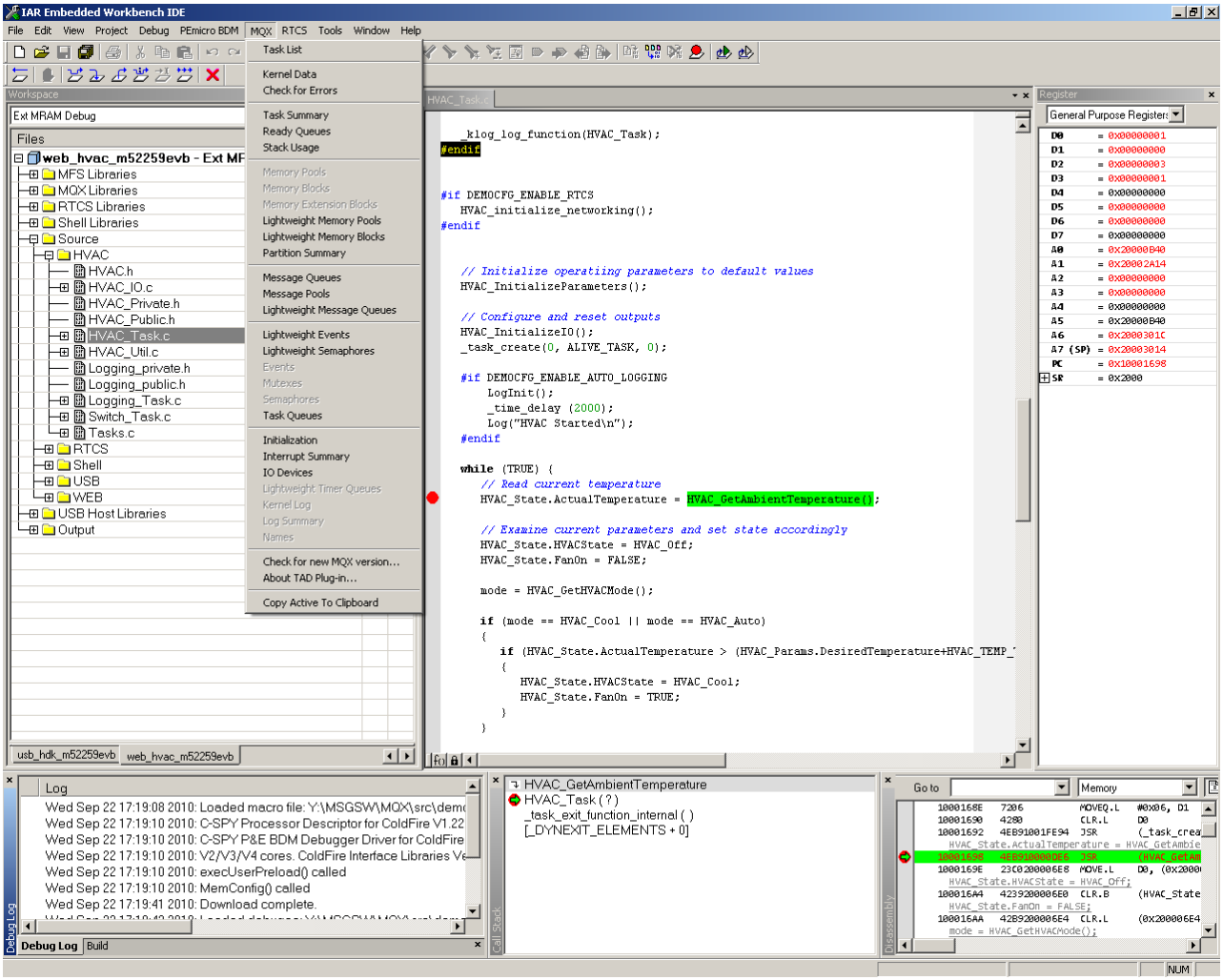
TAD plug-in DLL is pre-installed in IAR Embedded Workbench automatically. In case you need to update the plug-in to a new version included with the latest MQX installation, perform the following manual installation steps:

1. Close the IAR Embedded Workbench IDE
2. Locate the `tools\iar_extensions\<platform>` directory in Freescale MQX™ RTOS installation folder (by default `C:\Program Files\Freescale\Freescale MQX x.y`)
3. Copy the entire content of `tools\codewarrior_extensions\<platform>` directory to the IAR Embedded Workbench installation folder (e.g. `C:\Program Files\IAR Systems\Embedded Workbench 5.5\cf`)
4. After the steps above are done, verify the TAD plugin files exist at the new location:
   `<EW>\<platform>\plugins\rtos\MQX\MQXRtosPlugin.ewplugin`
   `<EW>\<platform>\plugins\rtos\MQX\MQXRtosPlugin<PLATFORM>.dll"`
5. Re-start IAR Embedded Workbench IDE.
6. In the Embedded Workbench environment, you should be now able to enable MQX TAD by selecting *"MQX"* in the *"Plugins"* tab of the "*Debugger"* panel of project settings. All example applications coming with Freescale MQX™ RTOS are already configured so.



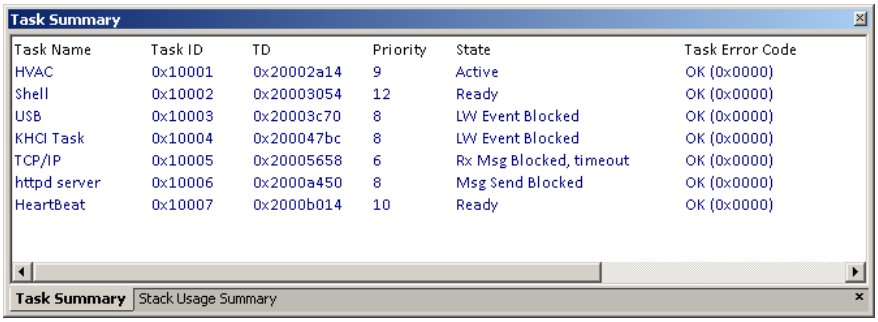
3.2.2 Using MQX TAD Screens

Using the MQX or RTCS menu in IAR IDE main window, several TAD “screens” may be opened during the debugging session.



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

- *Task Summary* – overview about all tasks created in the MQX application.

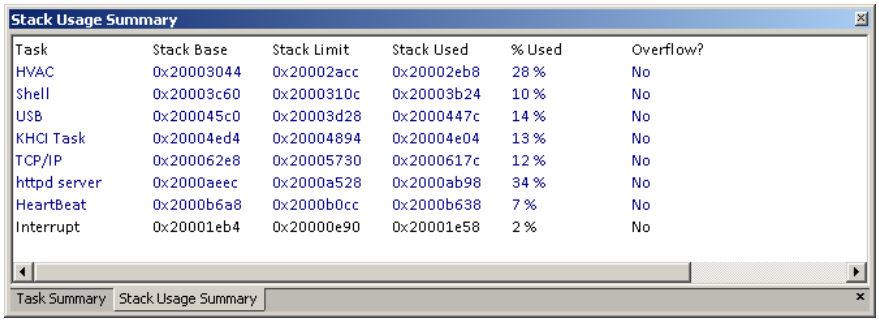


Task Summary

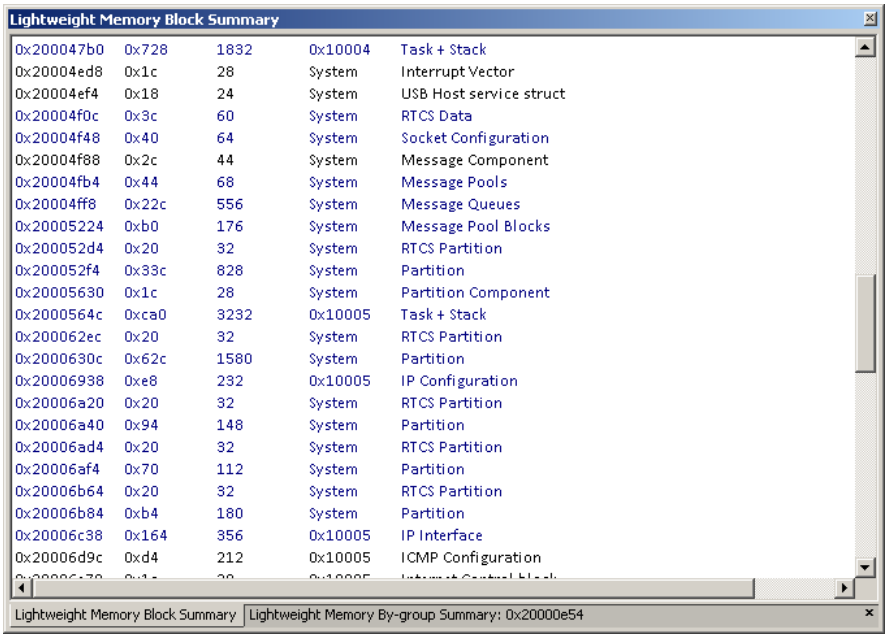
Task Name	Task ID	TD	Priority	State	Task Error Code
HVAC	0x10001	0x20002a14	9	Active	OK (0x0000)
Shell	0x10002	0x20003054	12	Ready	OK (0x0000)
USB	0x10003	0x20003c70	8	LW Event Blocked	OK (0x0000)
KHCI Task	0x10004	0x200047bc	8	LW Event Blocked	OK (0x0000)
TCP/IP	0x10005	0x20005658	6	Rx Msg Blocked, timeout	OK (0x0000)
httpd server	0x10006	0x2000a450	8	Msg Send Blocked	OK (0x0000)
HeartBeat	0x10007	0x2000b014	10	Ready	OK (0x0000)

Task Summary | Stack Usage Summary

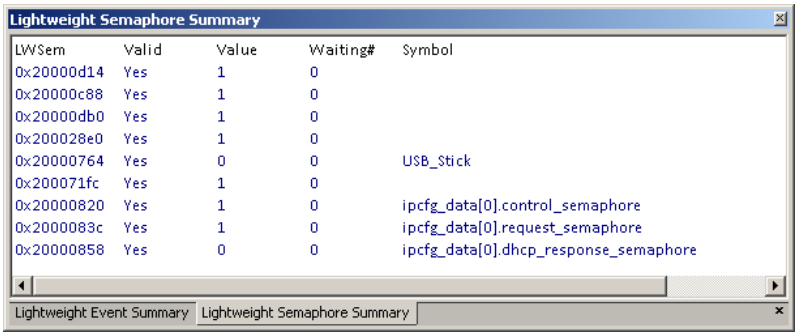
- *Stack Usage Summary* – displays information about interrupt and task stacks. Typically, stack overflow is a root cause of vast majority of problems in MQX user applications.



- *Memory Block Summary* (or *Lightweight Memory Block Summary*) – displays address, size and type information about each memory block allocated in the default memory pool by the MQX system or applications. Additional memory pools (if used) may be displayed using the "*Memory Pools*" screen.



- *Semaphores, Events* (or *Lightweight Semaphores, Lightweight Events*) – displays address and status of synchronization objects created by the MQX system or application. When a synchronization object is allocated as a global or static variable in the system, 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.

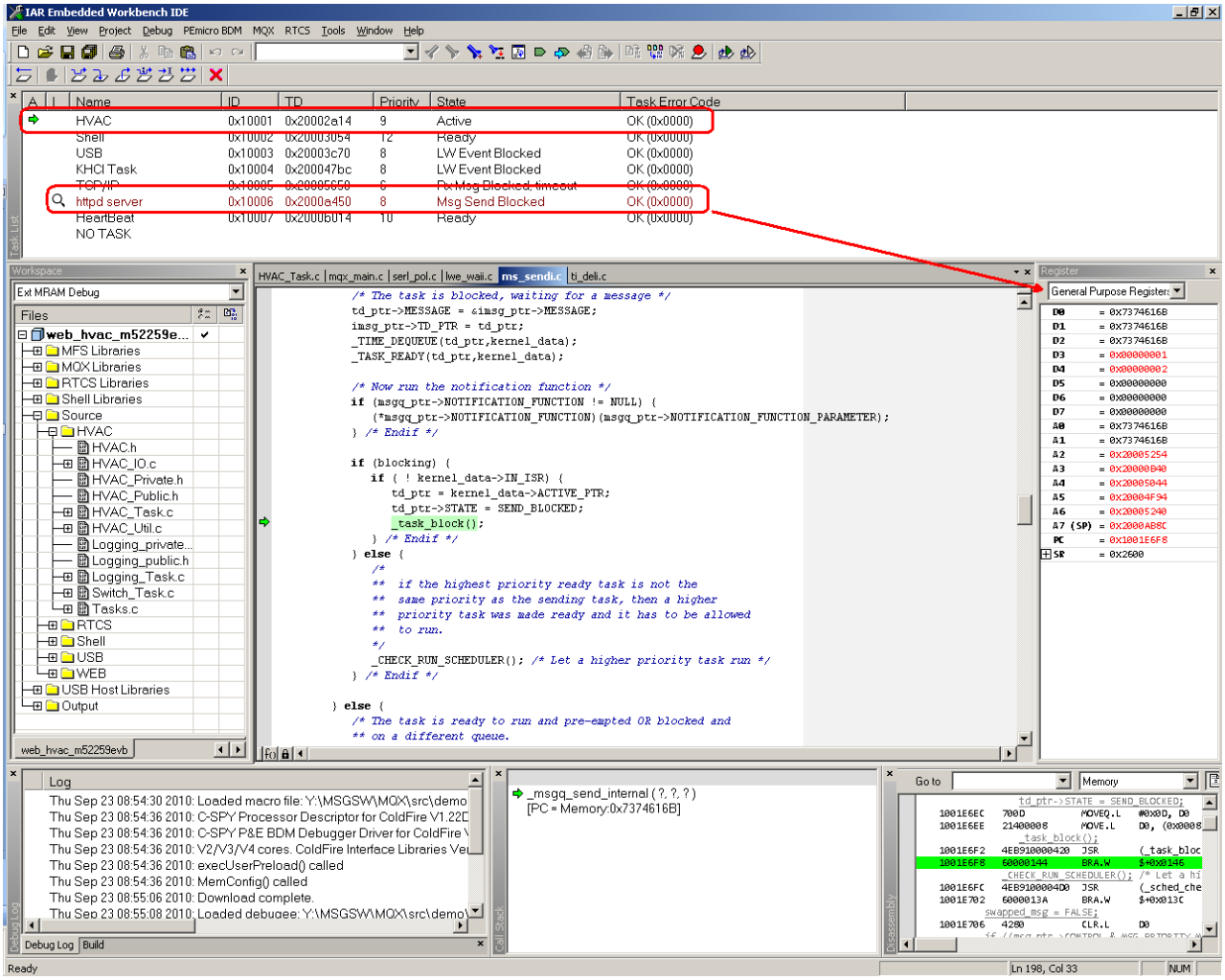


3.2.3 Task-aware Debugging

The TAD plug-in also provides native debugger support for multi-tasking MQX environment. Individual tasks can be examined any time the application stops on breakpoint or when it is stopped manually by pressing the “Break” red-hand toolbar button.

In the MQX menu in the IAR IDE main window, select the “Task List” item at top of the menu. The Task List view will open at the top of the window and will give you a list of all running tasks.

- The Green Arrow ➪ icon indicates which task was active at the moment of break.
- The Lens 🔍 icon indicates which task context is currently examined in the debugger in terms of execution point, register values, etc. Double click task items in the “Task List” view to move the lens and examine other tasks.

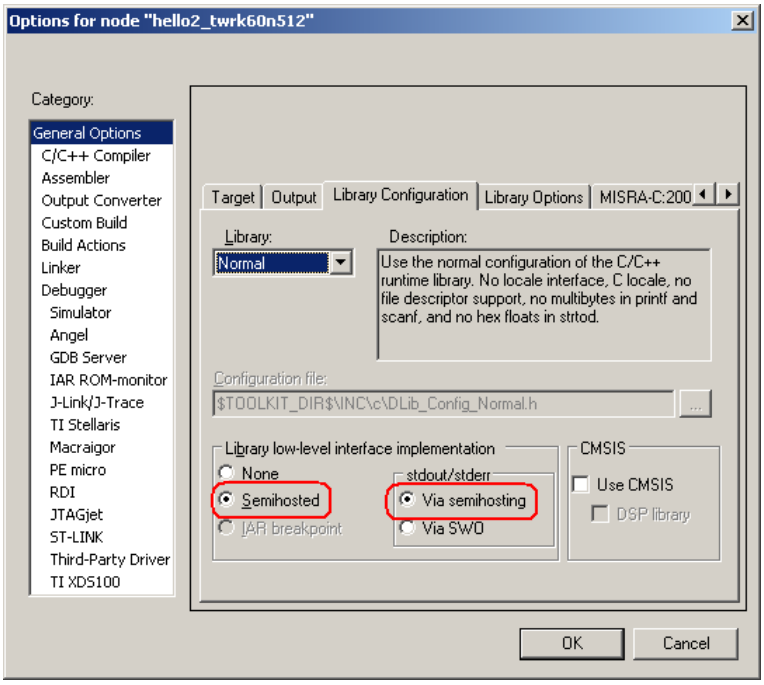


4 Using the MQX DebugIO Driver with EWARM IDE

The MQX provides the DebugIO driver allowing the processor to communicate with PC host computer via a debugger probe. The DebugIO channel can also be used as a default console for standard input and output operations. See more details about this driver in the “Getting Started with Freescale MQX™ RTOS” document.

The MQX RTOS currently supports ARM CortexM Semihost and ITM technologies. The IAR EWARM supports the Semihost communication channel for both input and output direction.

Change the “low-level interface implementation” settings in the project options, the *General Options* group, *Library Configuration* tab to enable debug console in the IDE:



The console can be opened during debugger session using the *View / Terminal I/O* menu in the EWARM IDE.