

# Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench

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

## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://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](http://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](http://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](mailto: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](mailto: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](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor 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 Semiconductor 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 Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. ARC, the ARC logo, ARCangel, ARCform, ARChitect, ARCompact, ARCTangent, BlueForm, CASSEIA, High C/C++, High C++, iCon186, MetaDeveloper, MQX, Precise Solution, Precise/BlazeNet, Precise/EDS, Precise/MFS, Precise/MQX, Precise/MQX Test Suites, Precise/RTCS, RTCS, SeeCode, TotalCore, Turbo186, Turbo86, V8 μ RISC, V8 microRISC, and VAutomation are trademarks of ARC International. High C and MetaWare are registered under ARC International.

All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2010. All rights reserved.

Rev. 02  
12/2011

## Table of Contents

Getting Started with Freescale MQX™ RTOS and IAR Embedded Workbench.....	i
1 Read Me First .....	2
2 Building the MQX Libraries .....	3
2.1 Compile-time Configuration .....	3
2.2 Build Configurations .....	4
2.3 Batch Build in IAR Embedded Workbench IDE .....	4
3 MQX Task Aware Debugging .....	6
3.1 Debugging MQX Applications in IAR Embedded Workbench .....	6
3.2 TAD CSpy Debugger Plug-in.....	7
4 Using the MQX DebugIO Driver with EWARM IDE.....	11

## 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

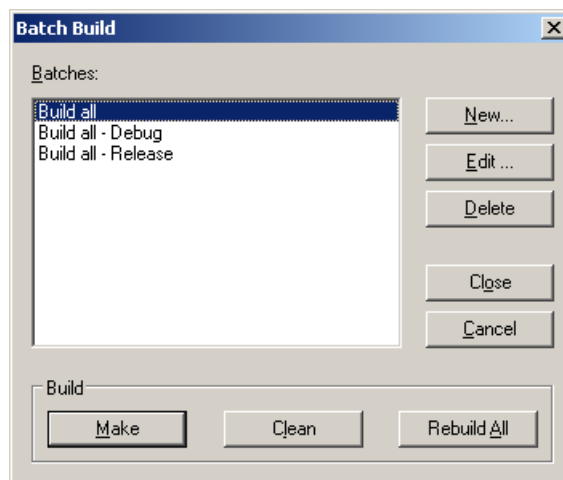
With IAR, the MQX build process can be simplified by using Batch Build feature. For each supported board, there is an IAR Workspace file which includes build projects for all related MQX libraries:

```
<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.

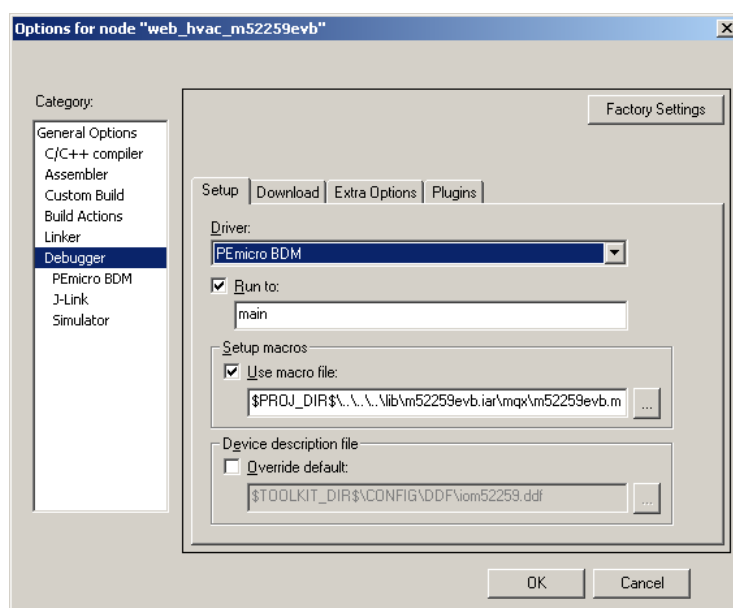



## 3 MQX Task Aware Debugging

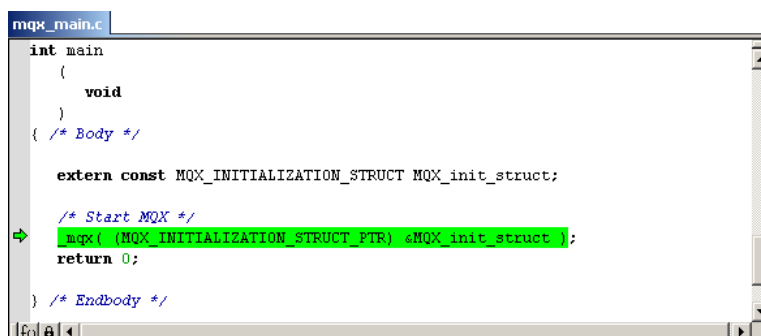
MQX Task Aware Debugging plug-in (TAD) is an optional extension to a debugger tool which enables easy debugging of multi-tasking applications. It helps to visualize internal MQX data structures, task-specific information, I/O device drivers and other MQX context data.

### 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.



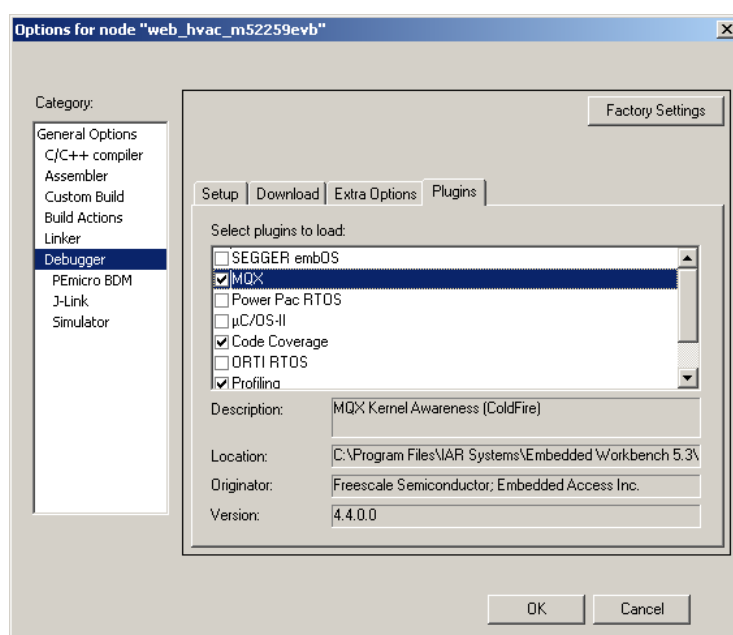


## 3.2 TAD CSpy Debugger Plug-in

### 3.2.1 Installing CodeWarrior TAD

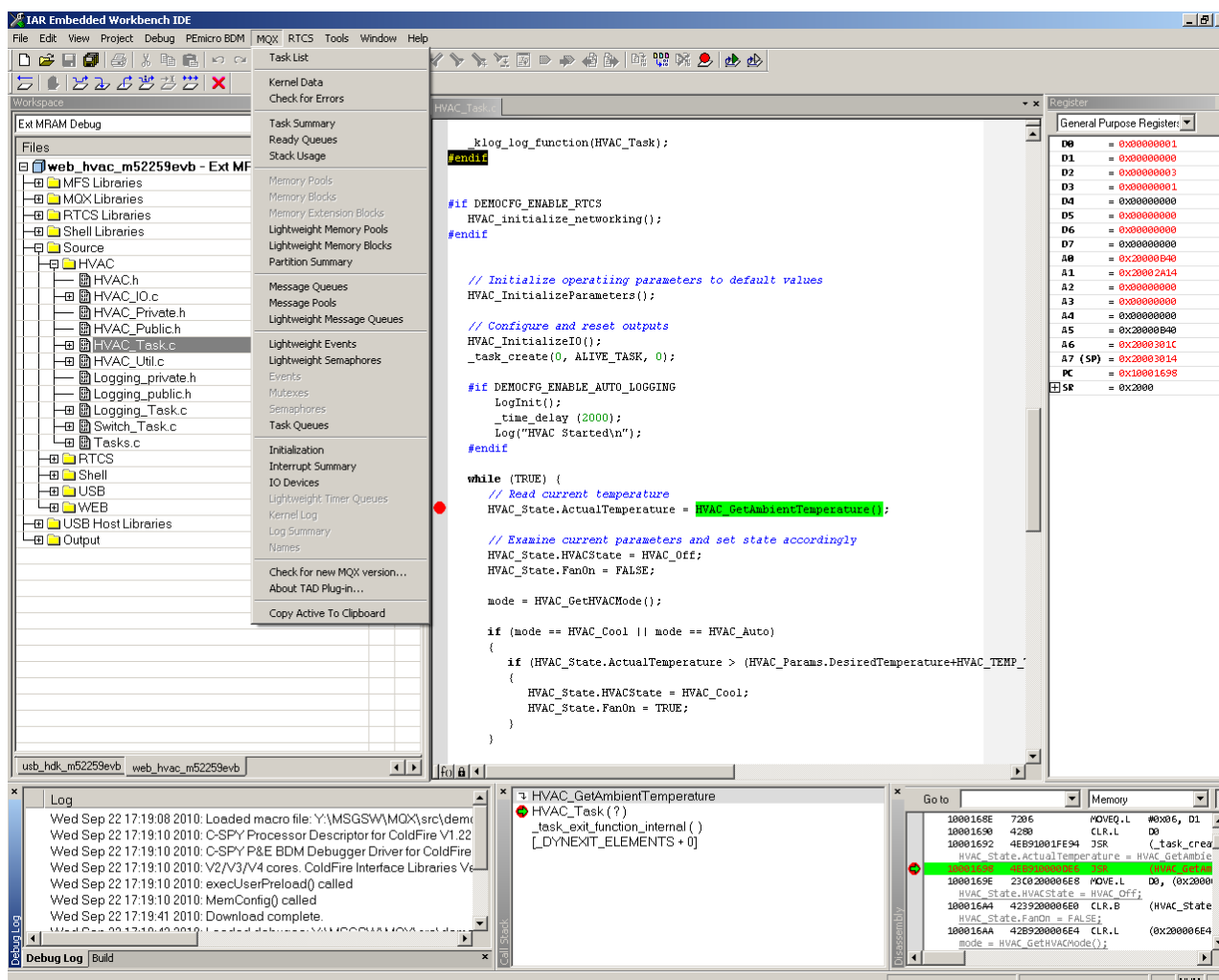
TAD plug-in DLL is pre-installed in IAR Embedded Workbench automatically. In case you need to update the plug-in to a new version included with the latest MQX installation, perform the following manual installation steps:

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.

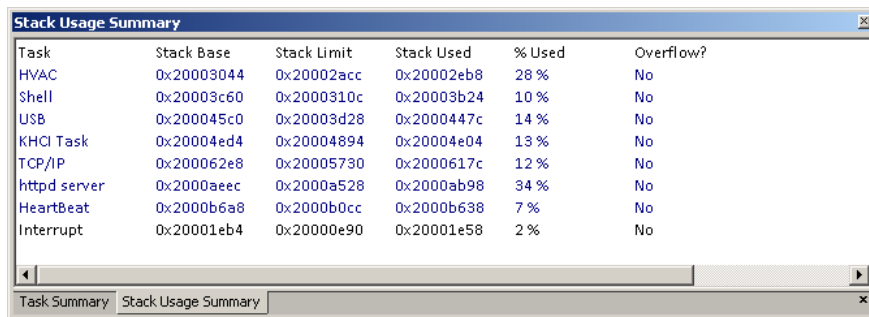


The most helpful and frequently used screens are shown in the pictures below:

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

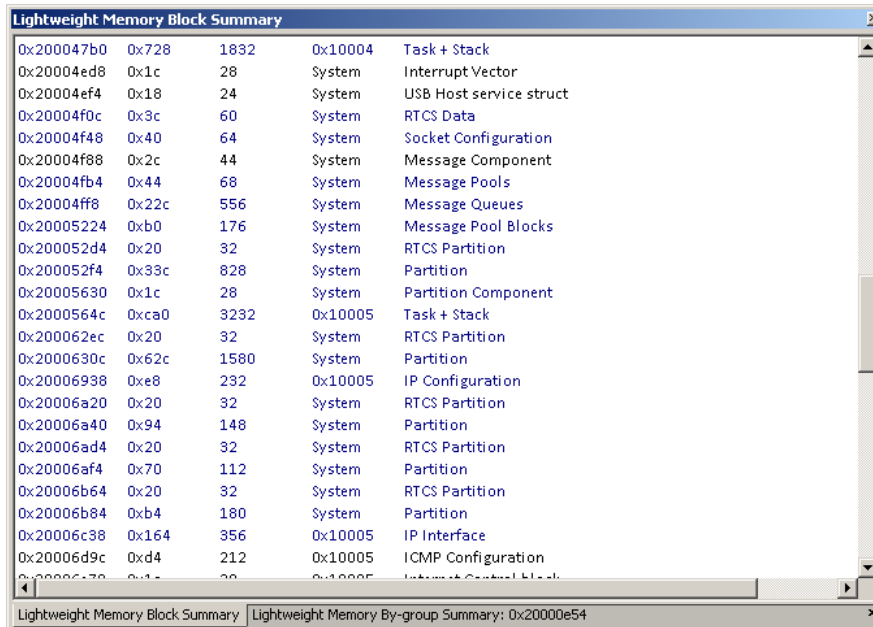
Task Summary						
Task Name	Task ID	TD	Priority	State	Task Error Code	
HVAC	0x10001	0x20002a14	9	Active	OK (0x0000)	
Shell	0x10002	0x20003054	12	Ready	OK (0x0000)	
USB	0x10003	0x20003c70	8	LW Event Blocked	OK (0x0000)	
KHCI Task	0x10004	0x200047bc	8	LW Event Blocked	OK (0x0000)	
TCP/IP	0x10005	0x20005658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

- **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.



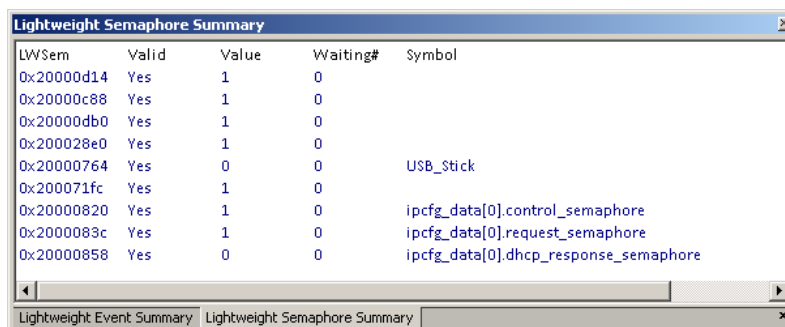
Task	Stack Base	Stack Limit	Stack Used	% Used	Overflow?
HVAC	0x20003044	0x20002acc	0x20002eb8	28 %	No
Shell	0x20003c60	0x2000310c	0x20003b24	10 %	No
USB	0x200045c0	0x20003d28	0x2000447c	14 %	No
KHCI Task	0x20004ed4	0x20004894	0x20004e04	13 %	No
TCP/IP	0x200062e8	0x20005730	0x2000617c	12 %	No
httpd server	0x2000aeec	0x2000a528	0x2000ab98	34 %	No
HeartBeat	0x2000b6a8	0x2000b0cc	0x2000b638	7 %	No
Interrupt	0x20001eb4	0x20000e90	0x20001e58	2 %	No

- **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.



Address	Size	Type	Description
0x200047b0	0x728	1832	0x10004 Task + Stack
0x20004ed8	0x1c	28	System Interrupt Vector
0x20004ef4	0x18	24	System USB Host service struct
0x20004f0c	0x3c	60	System RTCS Data
0x20004f48	0x40	64	System Socket Configuration
0x20004f88	0x2c	44	System Message Component
0x20004fb4	0x44	68	System Message Pools
0x20004ff8	0x22c	556	System Message Queues
0x20005224	0xb0	176	System Message Pool Blocks
0x200052d4	0x20	32	System RTCS Partition
0x200052f4	0x33c	828	System Partition
0x20005630	0x1c	28	System Partition Component
0x2000564c	0xca0	3232	0x10005 Task + Stack
0x200062ec	0x20	32	System RTCS Partition
0x2000630c	0x62c	1580	System Partition
0x20006938	0xe8	232	0x10005 IP Configuration
0x20006a20	0x20	32	System RTCS Partition
0x20006a40	0x94	148	System Partition
0x20006ad4	0x20	32	System RTCS Partition
0x20006af4	0x70	112	System Partition
0x20006b64	0x20	32	System RTCS Partition
0x20006b84	0xb4	180	System Partition
0x20006c38	0x164	356	0x10005 IP Interface
0x20006d9c	0xd4	212	0x10005 ICMP Configuration

- **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.





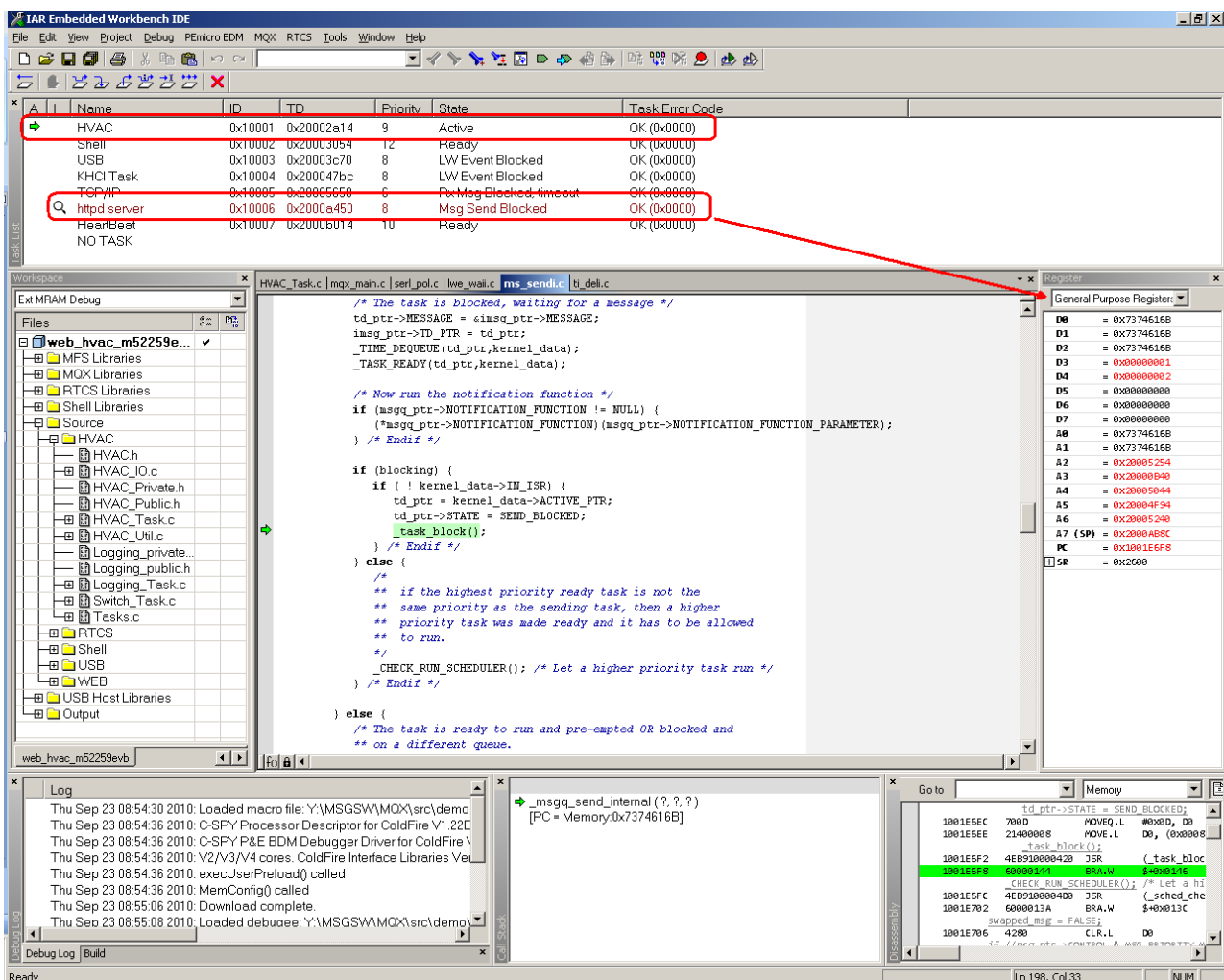
LWSem	Valid	Value	Waiting#	Symbol
0x20000d14	Yes	1	0	
0x20000c88	Yes	1	0	
0x20000db0	Yes	1	0	
0x200028e0	Yes	1	0	
0x20000764	Yes	0	0	USB_Stick
0x200071fc	Yes	1	0	
0x20000820	Yes	1	0	ipcfg_data[0].control_semaphore
0x2000083c	Yes	1	0	ipcfg_data[0].request_semaphore
0x20000858	Yes	0	0	ipcfg_data[0].dhcp_response_semaphore

### 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.



The screenshot displays the IAR Embedded Workbench IDE interface. The **Task List** window at the top shows a list of tasks with columns for Name, ID, TD, Priority, State, and Task Error Code. The **HVAC** task (ID 0x10001) is highlighted with a green arrow, indicating it was active at the moment of the breakpoint. The **httpd server** task (ID 0x10006) is highlighted with a red circle and a red arrow pointing to the **Register** window, indicating it is the current context being examined.

Name	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)
TSP/JP	0x10005	0x20005650	6	PxMsg Blocked timeout	OK (0x0000)
httpd server	0x10006	0x2000a450	8	Msg Send Blocked	OK (0x0000)
HeartBeat	0x10007	0x2000b014	10	Ready	OK (0x0000)
NO TASK					

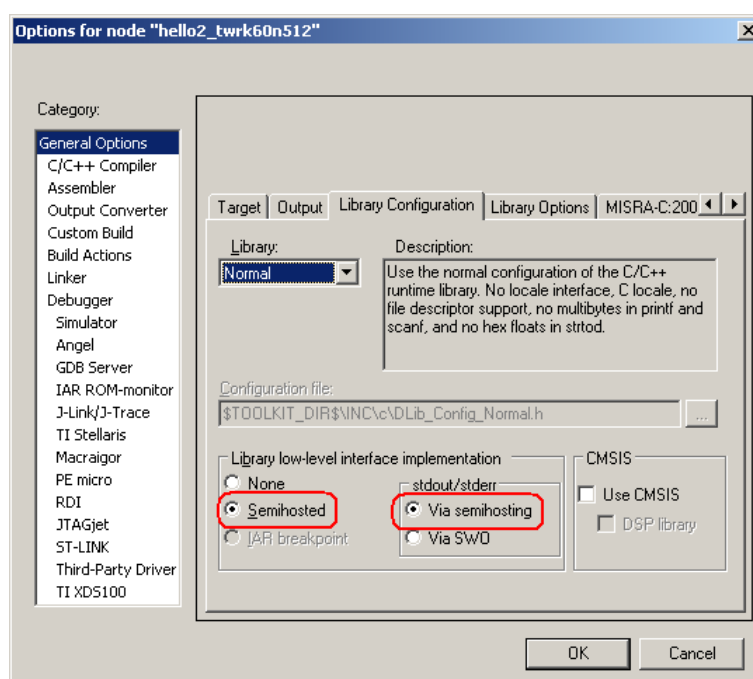
The **Source Code** window shows the `ms_send.c` file. The **Register** window displays the General Purpose Registers (D0-D15, SP, PC, SR) with their current values. The **Log** window shows the debug log, and the **Go to** window shows the memory address `0x7374616B`.

## 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.

