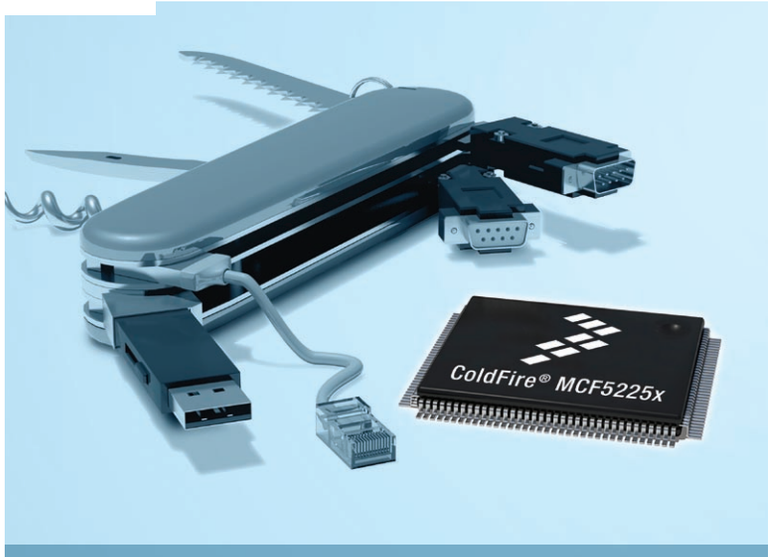


MCF5225X
Connectivity Labs 5 and 6

freescale semiconductor

About the Tower System

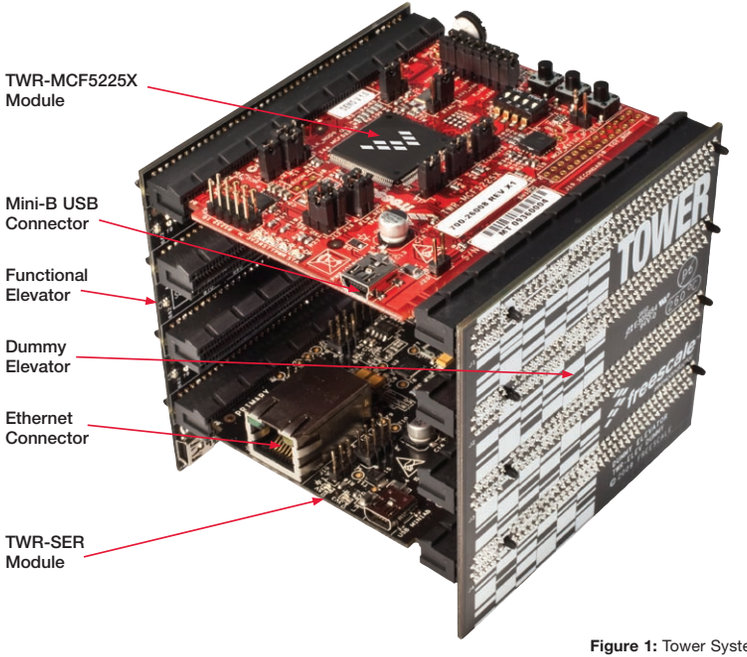
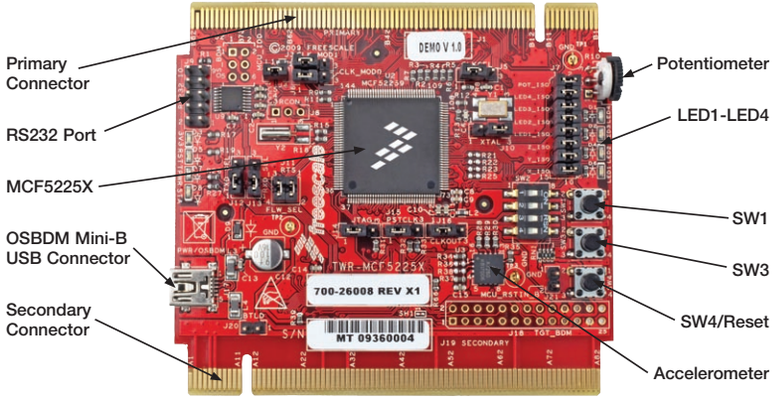


Figure 1: Tower System

Get to know the TWR-MCF5225X



TWR-MCF5225X-KIT
Freescale Tower System

The TWR-MCF5225X module is part of the Freescale Tower System, a modular development platform that enables rapid prototyping and tool re-use through reconfigurable hardware. Take your design to the next level and begin constructing your Tower System today.

MCF5225X—Lab Tutorials 5 and 6 *(sheet 3 of 3)*

LAB 5

Finding an Error Using Task Aware Debugging (TAD) in CodeWarrior™

This lab will show you the power of using task-aware debugging to troubleshoot your application. An error has purposely been introduced into this project, and this lab will show you how to find and solve that error.

Demonstrates

- Sending messages between tasks (logging task)
- CodeWarrior task-aware debugging windows

Step by Step Instructions

1. Stop the application if it is currently running (**Debug>Kill**)
2. Open the Lab Project by selecting the **File > Open** menu item: **C:\Program Files\FreescaleMQX3.4\demo\hvac_error\codewarrior\hvac_error_twrmcf52259.mcp**
3. Enable the auto logging feature. This is done by opening the **hvac.h** file
4. Then change the auto logging define:

#define DEMOCFG_ENABLE_AUTO_LOGGING 0

to this:

#define DEMOCFG_ENABLE_AUTO_LOGGING 1

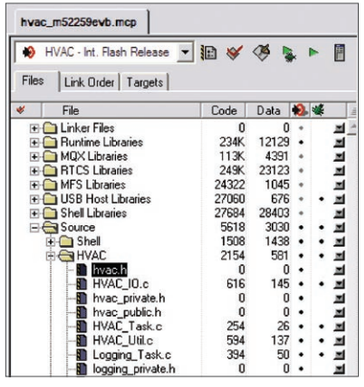


Figure 1: MQX source tree

5. Notice that the impacted files have a red check mark beside them to indicate that they need to be re-compiled, or are "touched." Compile, download, and run the application as was done in steps 8 to 15 of Lab 1.
6. Go to the shell console in hyperterminal. Note that a string of logging information will be printed out every 15 seconds. It will also be printed out when there is an update to any of the parameters such as the desired temperature. You can test this by pressing SW1.
7. Press SW1 until the desired temperature gets to 24°C and then use SW3 to bring it back down to 20°C. You should notice that the logging will eventually stop and that no more updates are printed.

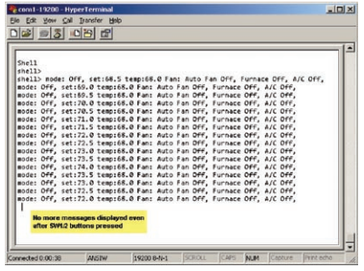


Figure 2: Up to 24, on the way down it stops logging

8. Your job now is to use the task aware debugging (TAD) feature to check for errors to determine why this error is happening. To see the TAD data, pause the application by clicking on the **Break** icon (which is a red square) or by selecting **Break** in the **Debug** menu.

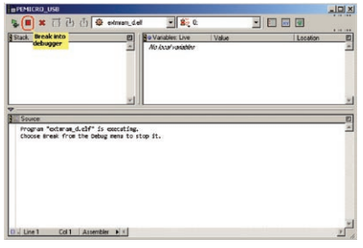


Figure 3: Break application execution

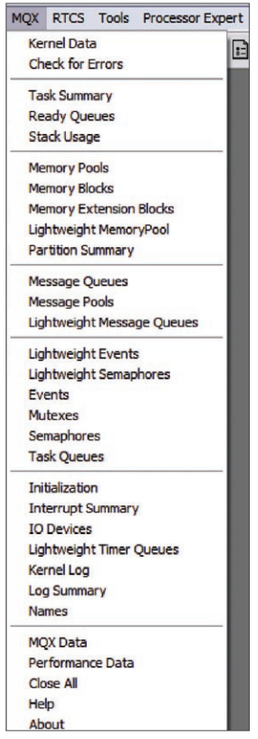


Figure 4: MQX Task-Aware Debugging menu

9. Then click on the **MQX** pull down menu. Read down the list of available information windows and select the one(s) that you might think would be a good indication as to what happened.
10. One of the TAD windows which will quickly help you to get an application status overview is Task Summary or Check for Errors.

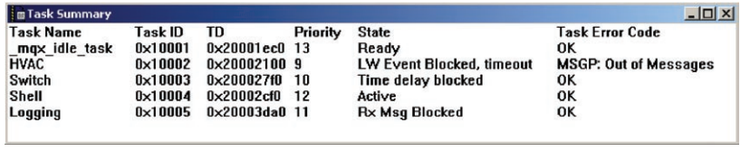


Figure 5: Task Summary TAD window

Solution

The HVAC task is using MQX messages to send data to the Log task (see HVAC_LogCurrentState() and the Log() functions) in **HVAC_Util.c** and **Logging_Task.c**. The HVAC task, as the message sender, assumes each message is "consumed" by the Log task and removed from the message pool after the text is printed to the console.

From the HVAC task error code found in the Task Summary TAD window seen in Figure 5, it is apparent that a message could not be sent because the message pool is full. You can verify this assumption by showing the Message Pools TAD window (Figure 6) and double-clicking on the one and only message pool entry to bring up the window shown in Figure 7.

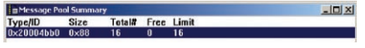


Figure 6: Message Pool Summary TAD window

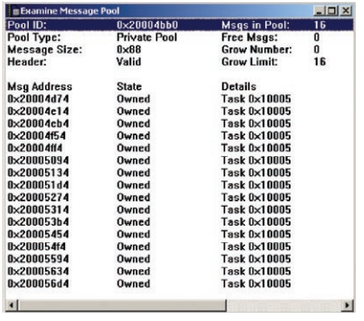


Figure 7: Message pool examined (after double-clicking the pool entry)

The message pool is exhausted. The problem is on the receiving side, as it is always the message receiver's responsibility to reuse the message object or free it when no longer needed.

Looking at the Logging_task() function, located in **Logging_Task.c**, you can see the message is received by

msg_ptr = _msgq_receive(log_qid, 0);

and after that the data of the message (the log text) is printed

printf(msg_ptr->MESSAGE);

What is missing is deletion of the message after the log text is printed:

_msg_free(msg_ptr);

Add this line: recompile and run the application. The message memory will now be released after the message is printed out.

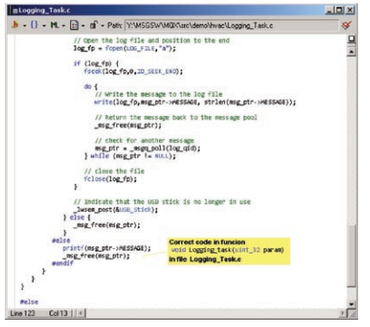


Figure 8: Correct code

LAB 6

Ethernet to Serial Bridge, Freescale MQX RTCS

This lab demonstrates how to create a bridge between a TCP/IP (telnet) connection and a serial line.

Demonstrates

- MQX RTCS TCP/IP network stack
- Custom telnet server implementation
- Re-directing STDIN and STDOUT output within an MQX task

Step by Step Instructions

1. Make the following connections from the Tower System to the computer.
   a. USB debugger connection (J17 on the TWR-MCF5225X module) to a USB port on PC
   b. Serial port on the **TWR-SER** module to a serial port on PC (serial cable not included)
   c. An Ethernet cable between the **TWR-SER** module and an Ethernet port on your computer
2. The first time you connect the USB debugger cable to your PC, Windows will install a driver for the debugger. Follow the prompts to automatically detect and install the driver.
3. Open the lab project by selecting the **File > Open** menu item: **C:\Program Files\FreescaleMQX3.4\demo\telnet_to_serial\codewarrior\telnet2ser_twrmcf52259.mcp**
4. The default IP address of the board is 169.254.3.3. Typically, when you connect your computer directly to the board, the computer will default to an auto IP address on the same subnet as the board (169.254.x.x), therefore requiring no setup. Note: The PC may take a few minutes to default to the auto IP address and make the connection.

continued on reverse side...

freescale semiconductor

continued from reverse side...

However, if you have trouble connecting, you may configure the IP address of the computer manually. Select Start > Settings > Network Connections > Local Area Connection. Note your original TCP/IP settings, and then set your IP address to 169.254.3.4 and your subnet mask to 255.255.0.0.

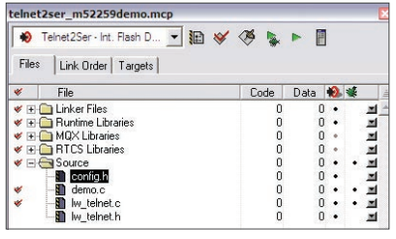


Figure 1: MQX source tree

5. Open the **config.h** file in the CodeWarrior window as shown in Figure 1. Double click the file item located in the "Source" group in the CodeWarrior project tree.
6. Locate the line of code starting with **#define ENET_IPADDR** and specify your target IP address by using the IPADDR macro. Set the target address to 169.254.3.3, and the line will be:

#define ENET_IPADDR IPADDR(169,254,3,3)

7. Do the same with the IP address mask value ENET_IPMASK:

#define ENET_IPMASK IPADDR(255,255,0,0)

8. Compile, download, and run the application and open a hyperterminal window as was done in steps 8 to 15 of Lab 1.
9. Open a command prompt on the PC (Start > All Programs > Accessories > Command Prompt). At the prompt invoke a telnet session to the board by typing telnet 169.254.3.3 You will be connected to the MQX shell via telnet.
10. Now the serial console and the telnet sessions should be "bridged." Type some characters into the telnet session, and you should see the characters appearing on the console terminal window. See Figure 2 for how it will appear.

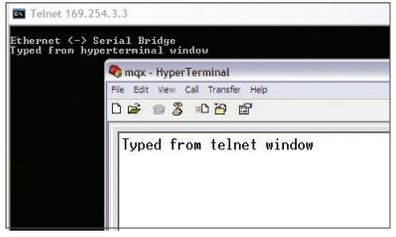


Figure 2: Ethernet to Serial Bridge

11. Then try typing into the console terminal window, and you should see the characters appearing in the telnet session.

Learn More: For more information about MQX and Freescale solutions, please visit **www.freescale.com/mqx** and **www.freescale.com/tower**.

Doc Number: TWRMCF52259LAB3 / REV 0
Agile Number: 926-78390 / REV A

freescale semiconductor