MC9S08LL64
Sheet 1 and 2

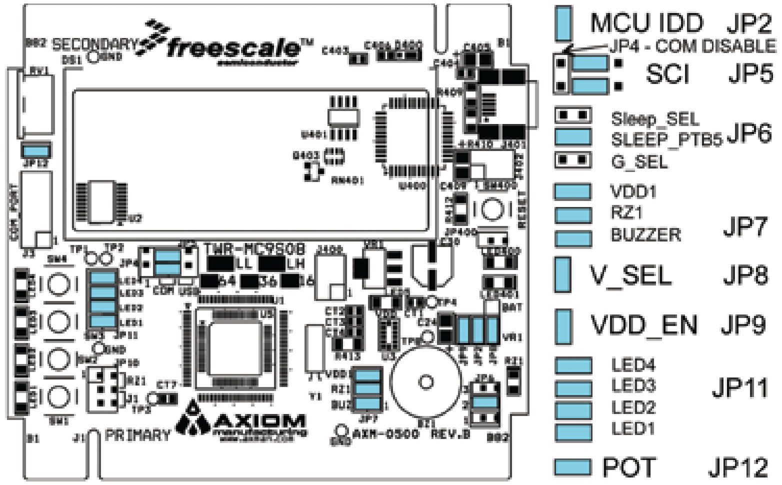
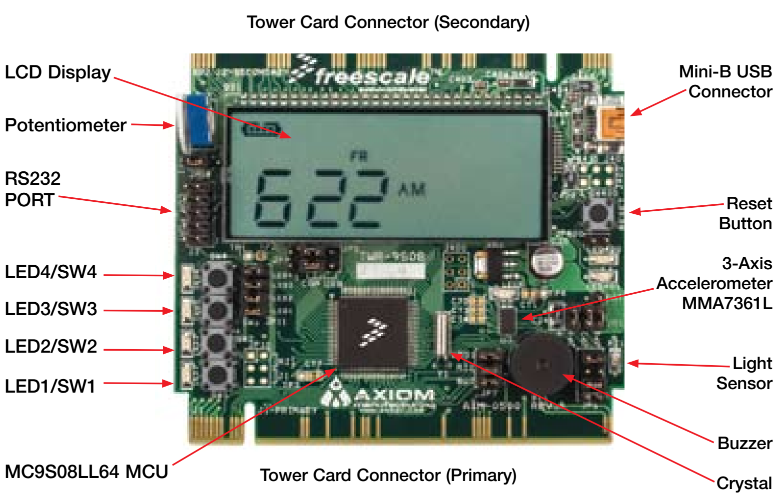


Figure 2
The jumper settings on the MCU Tower module are set to enable operation of the Quick Start application programmed into the flash memory of the MC9S08LL64 MCU.

Get to know the TWR-S08LL64



TWR-S08LL64-KIT
Freescale Tower System

The TWR-9S08LL64 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.

Learn More: For more information about Freescale products, please visit **www.freescale.com/tower** and **www.freescale.com/LCD.**

Doc Number: TWRS08LL64LAB1 / REV 0
Agile Number: 926-78375 / REV A

TWR-S08LL64—Lab Tutorials 1 and 2 *(sheet 1 of 2)*

Introduction

The MC9S08LL64 device is Freescale's low-power microcontroller with an integrated liquid crystal display (LCD) driver. TWR-S08LL64 contains an on-module display that allows developers to explore software development using the integrated LCD driver. This Lab Tutorial guide is designed to get you ready to develop your next LCD application using the MC9S08LL64 within minutes.

Note:
This Lab Tutorial can be followed once the steps in the Quick Start Guide, installing all of the software and documentation, have been finalized.

STEP 1

Lab Using Quick Start Code, Open CodeWarrior and the Project

This lab will highlight the capabilities of the MC9S08LL64 microcontroller with the application provided by the Quick Start Lab. Pushing switch SW2 will step you through these these four states.

State 1. Low power operation with time display
State 2. Potentiometer vs. light sensor comparison
State 3. Accelerometer demo, X, Y and Z output to SCI
State 4. ADC demo: when When in the ADC demo, pushing SW4 will increment the channel converted and displayed

1. Install software and tools as directed in the Quick Start Guide.
2. Open CodeWarrior for microcontrollers. From Windows start menu, you can locate it using **"Programs > Freescale CodeWarrior > CW for Microcontroller 6.2 > CodeWarriorIDE"**
3. Choose the **"Start Using CodeWarrior"** button.
4. Using CodeWarrior, click **File > Open** and open the **PE_LL64 Quick_Start.mcp** file from the directory on your c:\ where you unzipped the compressed projects. This is the Quick Start project which uses CodeWarrior's Processor Expert for device initialization.

STEP 2

Explore Project Window of Quick Start Code

The figures below illustrate the **"Processor Expert"** and **"Files"** tabs.

Processor Expert **Files**

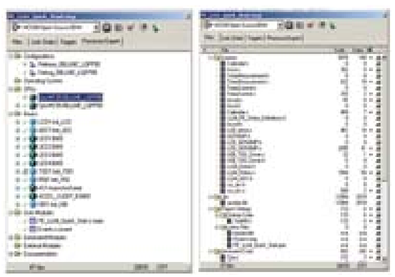


Figure 3

There are four tabs in the project window;
| File | Link Order | Target | Processor Expert |
Each tab displays the contents of the project with respect to the context of the tab selected. The user code is displayed in the Files tab.

STEP 3

Set Up TWR-S08LL64 Module

1. Connect the 10-pin connector to the 10-pin header on the module labeled COM PORT and J3 (see Figure 4). During states 2, 3 and 4 data is sent out the SCI port on the MCU to the terminal console on your computer.

2. Connect a RS232 cable from the DB9 pigtail to your computer serial port.

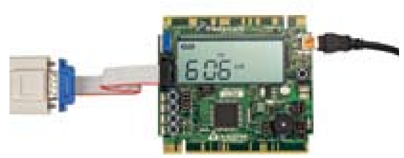


Figure 4

STEP 4

Start P&E Toolkit Application and Enter the Programmer/ Debugger

1. Open P&E Multilink Toolkit Launch Pad and Terminal Window. From Windows start menu, select **"Programs > P&E Embedded Multilink Toolkit > Toolkit Launchpad"**
2. Start a terminal console and configure your computer's com port for 19200 baud, 8 bit, 1 stop, no parity.
3. Using CodeWarrior, compile and program the MC9S08LL64 microcontroller with the application by clicking on **"Debug"** button or hitting F5 on your keyboard, launching the programmer and the open source BDM debugger.
4. If the message **"Load Executable File"** appears, click **"Yes."**

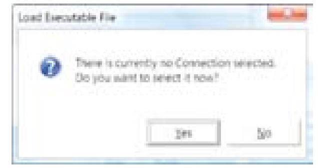


Figure 5

5. If the messages **"Set Connection"** and **"Set Derivative"** appear, set HCS08 – FSL Open Source BDM and derivative to **"MC9S08LL64"** and click **"OK."**

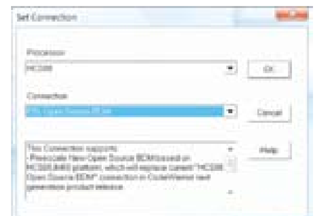


Figure 6

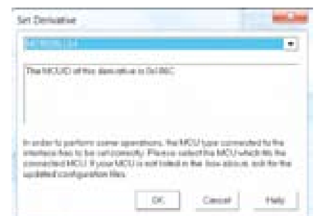


Figure 7

6. If the MCU is in low-power stop mode you may see the message **"There is currently no communication…"**Click **"OK."** Re-establish communication by hitting **SW2** and under the Component > Set Connection, set HCS08 and FSL open source BDM. This re-establishes BDM communications.

Figure 8

7. When the message, **"Loading a new application will stop execution of the current one"** appears, click **"OK."**

Figure 9

8. When the message, **"The debugger is going to mass erase the non-volatile memory of the current device, then reprogram the application"** appears, click **"OK."**

Figure 10

STEP 5

Debugger Window

A new debugger environment will open. From the main menu, choose **"Run > Start/Continue"** or press the button. The program will be executed in real time.

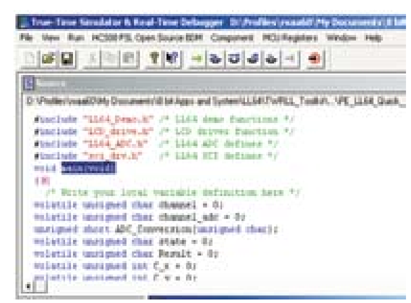


Figure 11

Lab 1 State 1

Clock Display State

Upon power-on or reset, the LCD will flash all segments on, then off, and will display "9LL64" and "CL." Next, it enters a low-power time/day display mode. The display is initialized with a clock value and the day of the week. The MCU will keep track of time with the TOD module running off the 32.768 watch crystal.

Figure 12

Highlighted features include the TOD and LCD module operations in low-power stop mode. While in stop mode, the LCD blinking function is operational.

Measure current: To measure the average current of the MCU, remove the jumper "MCU IDD" J2 and place a current meter between the two pins of J2. Reset the TWR_S08LL64 module with the Reset button. You should measure less than 3 micro-amps

Wake up time: The MCU is currently waking up every second and updating the seconds counter in the software. To reduce power consumption, the MCU wakeup time could be changed to 60 seconds. This reduces the times the MCU has to wake up, enter run mode and update the time on the LCD display.

In the file LL64_Demo.c the code to enable 60 second wake up is commented out. In the function **"void StopClock(void)"**, comment out the existing line that begins with **vfnTOD_Init** (and remove the comment **"//"** from the next line. When programmed with this new code, the MCU will wake up every 60 seconds. This changes from a second to the match interrupt. Measure the current again and compare.

TWR-S08LL64—Lab Tutorials 1 and 2 *(sheet 2 of 2)*

Lab 1 State 2

Potentiometer vs. Light Sensor State

This state uses two ADC channels to read the position of the potentiometer and the light sensed by RZ1, the on-module light sensor. Press switch SW2 and the program enters Potentiometer vs. Light Sensor state.

The display shows **"LI"** and the potentiometer value on the large characters on the bottom and the light sensor value on the smaller top right two characters. The third character displays the <, > or = char comparing the POT to the light sensor. The two values are also sent through the SCI port at 19.2 K baud.

1. Launch the **"Serial Grapher"** utility from the **PEMICRO UTILITY LAUNCH PAD** program.
2. Select your computers com port and set the baud rate to 19200, 8 bits, 1 stop bit, no parity. (Typically COM1)
3. Click **"Open Serial Port and and Start Demo."** The following should appear in the terminal window:
   "POT = FD Light Sensor Z1 = C4 "
   "POT = FD Light Sensor Z1 = C4 "
   "POT = FD Light Sensor Z1 = C4 "

Lab 1 State 3

Accelerometer Graphing State

Press switch **SW2** and the program enters the Accelerometer Graphing state. This mode uses the 3-axis accelerometer, the ADC and the SCI to measure and output the 3-axis data. Using the same setup as State 2, you will be able to graph the movement of the accelerometer as you move the module. Lab 2 provides a more in-depth accelerometer demo. To get to the display below you must zoom in and then reengage by clicking the Play button.

Figure 13

Lab 1 State 4

ADC Demo State

Press switch **SW2** again and enter the ADC demo state. The LCD displays **"ADC"** and the 12-bit value in the first three characters as well as the channel number in the two upper right characters. Pushing SW4 will increment the channel converted and displayed.

This mode measures and displays the selected ADC channel while sending the data to the SCI output. Using the same terminal setup as State 2 you will be able to see the following data being displayed in the terminal window.

15	00	0058	7F	15
15	00	0059	80	15
15	00	0055	81	15
or
15	13	05F7	4F	15
15	13	05F7	50	15
15	13	05F6	51	15

Where:
Column 1 is the ADCCFG1 register contents
Column 2 is the channel number as per Figure 15
Column 3 is the right justified 12 bit conversion result
Column 4 is a variable in the code that is being written to the VREF Trim register
Column 5 is the ADCCFG2 register contents

Figure 14 shows a table of the ADC channel assignments.
Press switch SW4 to change the channel and watch the LCD screen and SCI output.
Note that the VREFO channel (0x13) varies since the code is looping, modifying the trim of this voltage reference. This demonstrates the ability to adjust or trim the VREF output voltage.

ADCH	Channel	Input	Pin Control
0	AD0	ADP0	ADPC0
1	AD1	Reserved	ADPC1
10	AD2	Reserved	ADPC2
11	AD3	Reserved	ADPC3
100	AD4	PTA0/ADP4	ADPC4
101	AD5	PTA1/ADP5	ADPC5
110	AD6	PTA2/ADP6	ADPC6
111	AD7	PTA3/ADP7	ADPC7
1000	AD8	PTA4/ADP8	N/A
1001	AD9	PTA5/ADP9	N/A
1010	AD10	PTA6/ADP10	N/A
1011	AD11	PTA7/ADP11	N/A
1100	AD12	ADP12	ADPC12
10000	AD16	Reserved	N/A
10001	AD17	Reserved	N/A
10010	AD18	Reserved	N/A
10011	AD19	VREFO	N/A
10100	AD20	Reserved	N/A
10101	AD21	Reserved	N/A
10110	AD22	Reserved	N/A
10111	AD23	VLCD	N/A
11000	AD24	VLL1	N/A
11001	AD25	Reserved	N/A
11010	AD26	Temperature Sensor	N/A
11011	AD27	Internal Bandgap	N/A
11100	AD28	Reserved	N/A
11101	VREFH	VREFH	N/A
11110	VREFL	VREFL	N/A
11111	Module Disabled	None	N/A

Figure 14

Below is a table of the ADC channel assignments as they are connected on the Tower module. Reserved channels are skipped in the IRQ interrupt routine to set the next channel.

Channel (Hex)	Function 1	Function 2	Function 3
0	No Connection	JP10	J1-A30
4	Potentiometer	Zero-G Accel	
5	X – Axis		
6	Y – Axis		
7	Z - Axis		
8	SW3	J2-B27	
9	No Connection	J2-B28	
0A	RZ1 Light sensor	SW1	J1-A27
0b	SW2	J1-A28	
0C	JP10	J1-A29	
13	VREFO		
17	VLCD		
18	VLL1		
1A	Temp Sensor		
1B	Bandgap		
1D	VREFH		
1E	VREFL		

Figure 15

Lab 2

Accelerometer Demo

This lab will highlight the performance capability of the MC9S08LL64 microcontroller and show how this microcontroller can easily interface with a sensor. It will also detail how to use one of several software utilities included with your TWR-S08LL64 module.

The accelerometer application reads the X, Y and Z axes of the 3-axis accelerometer on the TWR-S08LL64 module using the microcontroller's A/D converter. It outputs the raw values of the accelerometer data on the microcontroller's serial communication interface.

Pressing the SW1 switch outputs a rolling average of the raw accelerometer data. Pressing the SW2 switch outputs a filtered version. Pressing the SW3 switch reverts back to the raw data output.

Lab 2

Open and Program MCU with Accelerometer Code

1. Remove the RZ1 jumper from JP6 so that the light sensor does not interfere with the operation of SW1. The rest of the jumpers should be in their default position.
2. Using CodeWarrior, click **File >Open** and open the **TWR9S08LL64_Accelerometer mcp** file from the directory on your c:\ where you unzipped the compressed projects.
3. Compile the code and program the MCU by clicking on **"Debug"** button, launching the debugger.
4. When the message, **"Loading a new application will stop execution of the current one"** appears, click **"OK."**
5. When the message, **"The debugger is going to mass erase the non-volatile memory of the current device, then reprogram the application"** appears, click **"OK."**
6. A debugger environment will open. From the main menu, choose **"Run > Start/ Continue."** The program will be executed in real-time.
7. Launch accelerometer utility from the **PEMICRO TOOLKIT LAUNCH PAD** and select **"Accelerometer."**
8. Set your computer's Com port, set the baud rate to 19200 and click **"Open Serial Port and Start Demo."**

Lab 2

Run Demo and Observe Graph

Note the X, Y and Z and C bar graphs and the scope window on the accelerometer graph.
If the values are too small to view, highlight a box around the graph data and hit the play button in the graph window. The raw XYZ data is being displayed and the response is very quick.

Press SW2 switch and "averaging" will be enabled. Notice the **"C"** bar chart or cycle count increase and the smoothing effect on the XYZ data as you move the module.

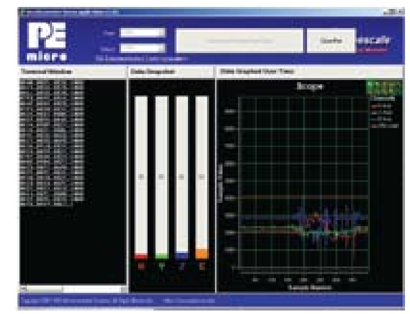


Figure 16

Press SW1 switch and the FIR filter algorithm begins executing. The cycle count will increase again and the response of the graph will change. Observe the change in the graph response.

Pressing SW3 switch will return to streaming the raw data from the accelerometer.

freescale semiconductor