

LAB #2: CodeWarrior

Lab writeup is due to your TA at the beginning of your next scheduled lab. Don't put this off to the last minute! There is pre-lab work to complete before the start of the next lab. **NO LATE LAB REPORTS WILL BE ACCEPTED.**

1 Objectives

- Learn how to use the CodeWarrior simulator for the MC9S12C32 derivative.
- Gain experience using the CodeWarrior debugger.
- Complete LCD output code that will be useful in future labs.

2 Tasks

1. Simulate the Lab2counter.asm program using the CodeWarrior simulator.
2. Examine Lab1Example.c program using the CodeWarrior debugger.
3. Add additional functionality to the Lab2LCD.c

3 Tools

1. Your prototype board and CSM12C32 module.
2. CodeWarrior for HCS12 Special Edition. Loaded on lab PCs and available for free download.
3. Your Project Board and USB cable.
4. Lab2counter.asm, Lab1Example.c and Lab2LCD.c code (download source from course website)

Students should also bring a diskette or USB memory drive for saving work completed during the lab. Internet is available on lab machines for e-mail.

4 Procedures

1. Simulating with CodeWarrior
 - (a) Visit the class webpage and download [Lab2counter.asm](#)
 - (b) Create a new assembly CodeWarrior project and replace the contents of main.asm with the contents of Lab2Counter.asm. Make sure when you create the project you have "Full Chip Simulation" selected. The method to create an assembly project is the same for C, except on Page 3 of the setup wizard, select assembly and deselect the other options. On page 4, make sure to select Absolute Assembly.

- (c) Before you perform a make, select “Full Chip Simulation” from the combo box under the project window.
- (d) Make the file and start the debugger (Project→Debug)
- (e) The debugger window will come up, do not press the green start button.
- (f) We now need to create some inputs and outputs for our simulation. To do this, click Component→Open. Select Visualizationtool from the list and click OK.
- (g) In the VisualizationTool window, right-click on the gray background and select “Add New Instrument,” and add an LED
- (h) Connect the simulated LED to your program by double-clicking on it to bring up its “properties” menu where you should select “kind of port” to be memory, “port to display” to be the address of PTAD, and “bitnumber to display” to be 0
- (i) To find the address of PTAD, open the file `mc9s12c32.inc`, typically located here:

```
c:/Program Files/Freescale/CodeWarrior for HCS12 V4.7/lib/hc12c/include/
```

This file contains all of the memory address mappings of the I/O devices. Do a search for PTAD. Another option for finding the correct memory addresses of the ports is to look under section 1.2.2 in the [MC9S12C Family Reference Manual](#).
- (j) Create two more LEDs and connect these to bits 1 and 2 of PTAD
- (k) Now, press the green run button. You should see the CPU cycles in the Register window incrementing.
- (l) Stop the simulator and start single-stepping it; the three LEDs should be operating as a 3-bit counter
- (m) Check-off this working simulation with your TA.

2. Debugging with Codewarrior

- (a) If you have not already done so, close the True-Time simulator & Real-Time debugger window. Do not close the CodeWarrior project.
- (b) Within the CodeWarrior project, select “P&E Multilink CyclonePro” from the combo box.
- (c) Make the program. Connect your CSM12C32 module to the project board, and connect the Project Board to the PC using the USB cable. Now run the debugger to download your program to the module.
- (d) Click the green run button and press either SW1, or PB2 and make sure the LEDs on the board are acting in the same fashion as in the simulation.
- (e) Click the red halt button to stop the program. Click the second button to the right of the green run button (Step Over(F10)) and observe how you can step through the program.
- (f) Press the PB2 button while stepping through the program and observe the behavior of the if statements as you step through.
- (g) What you may notice is that the logic levels of the push buttons of the project board and simulator are the opposite. The push buttons on the Project Board are low when pressed, and high otherwise. The push buttons in the simulator are high when pressed, and low otherwise. This difference needs to be taken into consideration when performing simulations of your code.

3. Add Functionality to Lab2LCD

- (a) Create a new project and make sure to have at least “P&E Multilink CyclonePro” selected.
- (b) Download Lab2LCD.c from the course website and replace your main.c with it.
- (c) Make the program and download it to your module over USB.
- (d) Run the program and you should see a message on the LCD screen. It should say “ECE5780” followed by “Lab2” If it does not, double check the jumpers on your project board and make sure MOSI, MISO, and the SCK jumpers are in place.
- (e) Once you’ve verified that the program is working properly, add additional functionality to the program by adding an LCDNum function (`void LCDNum(int val)`) that outputs the correct numerical character to the screen for a number between 0 and 9.
- (f) Next, add two functions called LCDDecimal (`void LCDDecimal(unsigned char val)`) that takes any byte and outputs its decimal value onto the LCD screen and LCDInt (`void LCDInt(unsigned int val)`) that takes a 16 bit integer and outputs the value of the integer.
- (g) Finally, add a function called LCDHex (`void LCDHex(unsigned char val)`) that takes any byte and outputs its hexadecimal value onto the LCD screen.
- (h) Use these functions to output the hex values 0xA9 and 0x3F to the screen followed by the integer 25134. Check-off this functionality with your TA.

5 Prelab

1. You should complete step 1 above before coming to your lab section and be prepared to show this simulation to your TA.
2. You should write the code for step 3 before coming to lab and be prepared to show this to your TA at the beginning of your lab section. This code does not need to be 100 percent debugged, but getting it close to working will allow your lab section to go more smoothly.

6 Writeup

1. Include a printout of your final commented code.
2. Describe any problems you encountered.
3. Compile the following assembly code into object code:

```

        ldx    #1234
        ldy    $234,X
        ldaa  1,X+
        ldab  2,Y+
        cba
        beq  done
        bgt  loopb
loopa:  deca
        cba
        beq  done
        bra  loopa
loopb:  decb

```

```
    cba
    beq done
    bra loopb
done: staa 1,-X
      stab 2,-Y
```