# Lab 1: Stopwatch

## Demo Due: February 19th, 2016

## Learning Objectives:

- Solder and wire wrap to integrate additional components with your PIC microcontroller
- Use timers to make precise timing measurements
- Interface with external components (switches, LEDs, LCD) with the PIC microcontroller
- Interface with an 8x2 character LCD display
- Use interrupts for precise timing
- Use bit masking operations to access 4-bit simultaneously from a single port
- Integrate the hardware and software components to build a stopwatch capable of timing minutes, seconds, and 1/100th of seconds with basic start, stop, and reset controls.

## Datasheets and References (also on D2L)

LCD Display Datasheet

Microchip PIC32 Starter Kit III User's Guide

Microchip PIC32MX470F512L Datasheet

Microchip PIC32 Expansion I/O Board Datasheet

## Hardware Provided for Lab:

2 - LEDs
2 - 1k Ohm resistors
1 - 8x2 LCD Screen (1.50” x 1”)
2 - Momentary tactile switch
Male Headers
*Optional:* 4x4 Vector Prototype Board
*Optional:* 2 - 1K Resistors
*Optional:* 2 - 10K Resistors
*Optional:* 2 - .1uF Capacitors

## Software Provided for Lab (Found on D2L):

- Part 1: lab1p1.c
- Part 2 and Part 3: lab1p2.c, lcd.h, lcd.c

# Lab Procedure and Demonstration:

## Part 1: Dueling LEDs

### Description:

In this part, you will interface the PIC32MX with two LEDs and control their state using an external switch.

### Requirements:

***For every distinct press of an external switch, the LED currently being illuminated should be alternated. For example, if the Run LED is currently on, the Run LED should turn off and the Stop LED should be turned on.***

### Report Responsibilities:

| Hardware Design | Software Design | Quality Assurance | Systems Integrator |
|---|---|---|---|
| The circuit diagram is complete. Appropriate colors are chosen for the wire-wrapping portion | Buttons are debounced. LEDs are clearly distinguished in the code. The RUN LED is initially on. | A circuit diagram for the circuit to test the LEDs is provided.<br><br>Picture, circuit diagram, or proof of at least two other tests are provided. | LEDs, resistors, and headers are properly soldered onto the vector board. Wire wrapping is done. A picture is provided. |

### Hardware Design:

1. Connect the 2 LEDs to TRD1 and TRD2 of the expansion board. The anodes *(long wire)* of the LEDs should be connected to 5V through a 1k Ohm resistor. The cathodes *(short wire)* should be connected to the TRD1 and TRD2 inputs.
2. Connect the momentary tactile switch to TRD3 of the expansion board. The momentary switch has 2 poles but 4 pins. Two pins on each side of the switch are connected together. Connect one side of the switch to ground and the other side to TRD3.

### Software Design:

1. Using the provided C code template (**lab1p1.c**) for Part 1 of this lab, write C programming code implementing the given requirements:
    a. Use define statements which represent the STOP and RUN LEDs outputs.
    b. Initially, the RUN LED should be illuminated.
    c. Both the button press and button release should be debounced in software using precise 5 ms delay. This delay should be controlled using Timer1 of the microcontroller.

### Quality Assurance:

1. Test the provided LEDs and make sure they work. Using the breadboard, create a circuit that will test the LEDs independently from the microcontroller.
2. Measure the resistance on any resistors used in the circuit to verify they are of the correct resistance.
3. Using the continuity measurement of your multi-meter, determine which pins correspond to the two poles of the switch. Test to see if pressing the switch shorts the appropriate sides.
4. Perform any other planned tests.

### Systems Integrator:

1. Once the components have been tested, the hardware design is complete, and the software is working properly, solder the resistors and LEDs onto a vector board. Do not solder the switch to the vector board at this time. Take a picture or two of your work. You will need it for the report.

2. Solder headers to the vector board and wire wrap both power and ground pins to the header.

# Part 2: LCD Interface and Software Driver

## Description:

In this part, you will interface the PIC32MX with an LCD screen using a 4-bit interface. You may not move onto this part until Part 1 has been demonstrated to the Lab TA.

## Requirements:

***The LCD screen should be able to display any character string specified in the code. All helper functions described in the following section should behave properly.***

## Report Responsibilities:

| Hardware Design | Software Design | Quality Assurance | Systems Integrator |
|---|---|---|---|
| Pin numbers on the LCD and microcontroller are included. Appropriate colors are used for each wire. | All functions work properly | Code for testing the delayUs() is provided and clearly listed.<br>Code for testing writeLCD() is shown and the oscilloscope screenshot is provided.<br>A screenshot for each of the four data pins for the writeLCD is provided | Same as for Hardware, Part 2. Also, have header pins properly soldered to the LCD Display. Show a picture of your work. |

## Hardware Design and Systems Integrator:

1. You will need to solder header pins onto the PIC32MX expansion board. You will not have enough headers to use have a pin on all 120 connections. Additionally, you will need to conserve some of your header pins for other labs. Therefore, attach headers onto the connections according to this table:

| Test Point Header on Expansion Board | Pins to Attach Header |
|---|---|
| **J10** | 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 47, 49, 51, 53, 55, 57, 59. **(Odds 1-37, Odds 47-59)** |
| **J11** | 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48. **(Evens 4-48)** |

2. Solder a header onto the LCD device so that the microcontroller can interface with it easily. Take pictures of your work. You will need this for the report.
3. Test your LCD by connecting the first 3 pins correctly and adjusting the potentiometer used on the board until you see black boxes on the first line of the LCD.

## Software Design:

1. Using the provided C code (**lab1p2.c, lcd.h, lcd.c**) complete the software driver code required to interface with the LCD. This C code includes a simple demonstration program that will utilize the LCD software driver to print sample messages on the LCD.
2. Develop a ***DelayUs( )*** function that will utilize ***Timer 2*** to create a user specified delay in micro seconds provided as function parameter.

3. Complete the ***LCDWrite( )*** function to write the most significant 4-bits followed by the least significant 4-bits to the LCD_D output. You will need to utilize bit masking and shifting operations to perform these writes.
4. Complete the ***LCDInitialize( )*** function to configure the operational mode of the LCD. Details of the initialization process and control instruction needed to configure the display are explained within the LCD Display Datasheet.
5. Complete the ***LCDClear( )*** function to clear the LCD display using the appropriate control instruction.
6. Complete the ***LCDMoveCursor( )*** function that will move the cursor to the user specified (x,y) coordinate. This operation must be performed using a single control instruction. As the move cursor control instruction depends on the format of the display, careful attention must be paid to utilize the correct control instruction for the 8x2 display format. This information can be found within the LCD Display Datasheet. Note that every address is in hexadecimal, not decimal so address 40 is actually 0x40.
7. Complete the ***LCDPrintChar( )*** function for displaying a single ASCII character on the LCD Display.

## Quality Assurance:

1. Test the connectivity of the header that is soldered onto the LCD screen. Make sure that every pin is connected to the board properly and that no shorts are created from the soldered joints.
2. Construct a software test that tests for ***delayUs()*** independently of the LCD being connected.
3. Using the oscilloscope, verify that the ***LCDWrite()*** function is working properly. To do this, you can place a call to ***LCDWrite()*** in a simple while loop. Since this behavior is periodic, the oscilloscope will be able to pick this behavior up continuously and you can display it on the screen.
4. Capture the oscilloscope screen for each of the data pins of the LCD while the ***LCDWrite()*** is called in the while loop.

# Part 3: The Stopwatch

## Description:

In this part, you will interface with the LCD interface and the previous set up with two LEDs to create a stopwatch.

## Requirements:

1) ***The Stopwatch will have 2 user inputs from the momentary tactile switches for Start/Stop and Reset. NOTE: The "Start/Stop" button has already been implemented in Part 1 of this lab.***
2) ***The Start/Stop input will be utilized to start and stop the execution of the Stopwatch. When stopped, the current time will be maintained, such that when the Stopwatch is started again, the timing will continue from the current time.***
3) ***When the Stopwatch is started, the top line of the LCD Display should display "Running:", and the "Run" LED should be on. When the Stopwatch is stopped, the top line of the LCD Display should display "Stopped:", and the "Stop" LED should be on.***
4) ***Use the switch on the development board, SW, to reset the current time back to 0 when the stopwatch is stopped. If the stopwatch is started (i.e. running), the reset input should be ignored.***
5) ***A change notification interrupt must be utilized to detect user inputs for the Start/Stop and Reset inputs.***
6) ***The Stopwatch must have an accuracy of 1/100th of a second (i.e. 10 milliseconds). The bottom line of the LCD Display will be utilized to display the current time using the format MM:SS:FF where MM is the current minute, SS is the current seconds, and FF is the current 1/10th ad 1/100th of a second.***
7) ***An interrupt using Timer 1 must be utilized to control the timing of the Stopwatch.***

## Report Responsibilities:

| Hardware Design | Software Design | Quality Assurance | Systems Integrator |
|---|---|---|---|
| Same as part 1 | Interrupts are used, state machine implementation, requirements are fulfilled. | Software is provided for testing the getTimeString() function. | Everything is wire-wrapped and soldered. |

## Hardware Design

1. Combine the hardware of Part 1 and Part 2 appropriately.
2. Create a final circuit diagram that combines the design in Part 1 and Part 2 appropriately for the stopwatch functionality.

## Software Design:

3. Construct a finite-state machine to define the behavior of your stopwatch. Implement the stopwatch as a finite state machine using a switch statement.
4. Construct a function called ***getTimeString()*** which returns an appropriately formatted string to be written on the LCD Screen based off the elapsed time.

## Quality Assurance:

1. Write a software routine which verifies the functionality of the ***getTimeString()*** function written by the software designer. You do not need to verify this with the LCD. Rather, verify that the string it returns is in an appropriate format and value.

## System Integrator:

1. Once all of the requirements have been met, wire-wrap any connections that remain unwrapped. There should be no connections that are not wrapped, soldered, or connected by a cable.