University of Puerto Rico at Mayagüez
Department of Electrical and Computer Engineering
Microprocessor Interfacing Laboratory

Experiment 1: IDE, ASM/C Programming & IO

Objectives

- Become familiar with the process of assembling, debugging, and executing an assembly language program with your Microcontroller IDE
- Understand the basic anatomy of an assembly program
- Become familiar with your Microcontroller architecture
- Learn how to use the IO Ports with several examples
- Understand how to use an LCD with a microcontroller

Duration

- 2 Hours in laboratory and extra time for homework.

Materials

- Your Microcontroller IDE application
- Your Microcontroller development board
- 2x16 LCD W/HD44780 Controller, 4 Push Buttons, 3 Leds and a 2x12 Pin Header Development Board

Introduction

For all the upcoming experiments the microcontroller IDE will be use as the compiler, assembler, linker and code debugger. Some MCUs have more than one IDE choice. You will choose one depending on the main language to be used to program your microcontroller (assembly, C or other language). Although we'll probably only use assembly and C, some IDEs allow you to add compilers for different languages.

IDE stands for Integrated Development Environment also called integrated design environment or integrated debugging environment. Normally an IDE for programming microcontrollers consist of the following tools:

- A code editor
- A compiler or/and assembler
- A debugger
- A download tool to program the MCU flash memory
- Built-in automation options

When you choose an IDE it is important to know the maximum limit of code you can write with it. Most demo versions of IDEs limit the code size to only a few kilobytes of length. It is also important to know if your IDE supports the type JTAG available in your MCU. The JTAG interface is used for debugging your code in your embedded system.

In addition to gaining familiarity with an IDE, during this experiment you will also learn how to work with IO ports through several different examples. In particular, one of the examples will address developing proficiency using alphanumeric LCDs as I/O devices.

Procedure

MCU: _____ IDE: _____

IDE Tutorial

Locate the shortcut to your IDE

Since this is the first time using your IDE, it may ask for a location to store your projects files. Select one location.

Find the menu to create new projects and create a project as “ICOM5217EXP1”.

Specify your MCU model.

Now create a new source file. “ASM file”

Name your source file as “FlashLed.asm” under 'Source File:'

Write the Pseudocode below in Assembly code in the source window and then save the work done. For this code connect a LED with a 220 ohms resistor in an I/O port of your MCU.

```
Blinking LED, Pseudocode
;-------------------------------------------------------------------------
; Progam Start
; INIT RESET VECTOR
; INIT STACK POINT, WDT
;-------------------------------------------------------------------------
Port_bit = output   ;Set port as output
Port_pin = 0        ;initialize pin to “low”

While TRUE
  Port_pin = NOT(Port_pin) ;Toogle Pin
  Wait = 2000h
  While wait > 0            ;Delay Loop
    Wait = wait – 1
Endwhile
   Endwhile
  ;-------------------------------------------------------------------------
```

Find the menu for building your active project. Watch the console window any error message.

Correct any errors until you get 0 errors and 0 warnings.

Turn-in:

- Console message of 0 warnings and 0 errors.

The 2x16LCD4SW Board

This board includes a 2x16 LCD with an HD44780 controller, 4 push buttons, 3 LEDs and a 2x12-pin header, all mounted on a single PC board as shown in Figure 1. The 2x12 header provides access to all board components.

Figure 1: Picture of the 2x16LCD4SW Board.

The development board contains most of the required components for this laboratory, built in a single board. Minor parts such as resistors might need to be externally provided. Figure 2 shows an schematic of the board, while its pin layout is documented in Table 1.

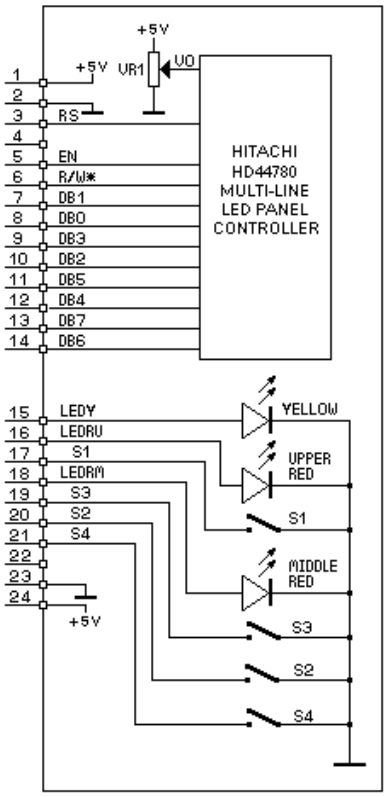


Figure 2: LCD/Keys board schematic.

Table 1: Pin organization in LCD/Keys board.

Pin	Label	Description	Ratings
1	Vdd	Supply voltage:	5V+-0.25 Max:-0.3-6.5
2	Vss	GND	
3	RS	Register Select:	0-5V Max:-0.3-Vdd+0.3
4	NC	No connection	
5	E	Read /write enable	0-5V Max: -0.3-Vdd+0.3
6	R/W	Read /write select	
7	DB1	Data bits	
8	DB0		
9	DB3		
10	DB2		
11	DB5		
12	DB4		
13	DB7		
14	DB6		
15	CLED	LED C (+)	Max:3V 25mA
16	ALED	LED A (+)	Max:3V 25mA
17	S1	Switch 1	Max:24Vdc 50mA
18	BLED	LED B (+)	Max:3V 25mA
19	S3	Switch 3	Max:24Vdc 50mA
20	S2	Switch 2	Max:24Vdc 50mA
21	S4	Switch 4	Max:24Vdc 50mA
22	NC	No connection	
23	Vss	GND	
24	Vdd	Supply voltage	

The Blinking LED

For running the code select 'Run > Debug Active Project'. The progress information is displayed while the code downloads. Once the download is completed the debug perspective opens automatically.

Select 'Run' from the 'Run' menu and verify the LED toggling on your MCU.

Use the debugger options and place a break point in the instruction where the ping is toggled and run the program and see how led state is changed.

Using the debugging tool run your code step by step and see the hardware reaction.

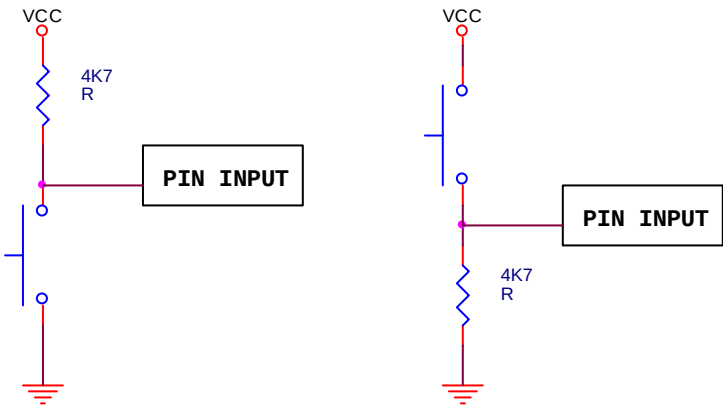
With the debugging options if your IDE allows to modify the port status where is connect the LED by using the debugger register modification options.

Exercise:

After you successfully make the led blink, change the “wait” variable to different values to see how the blinking frequency changes.

Polling a Switch

The main goal of the polling approach is to know the state of a switch. Connect the switch as indicated in Figure 3. We used a pull-up resistor when the switch in not pushed a high state is read; for it’s part the pull-down ensures that when the switch is not pushed a low state is read.



a) Pull-up *b)Pull-Down*

Figure 3: Pull-up vs. Pull-down schematic.

Select which port you use and set up the port as input to read the switch. This can be accomplished by setting up the direction register.

Next read the port data register to determine whether the push button was pressed. The polling approach is and I/O operation that will maintain the program in a looping state until the switch is pressed.

Change the switch connection as indicated in figure 1 and repeat step 3.

Exercise:

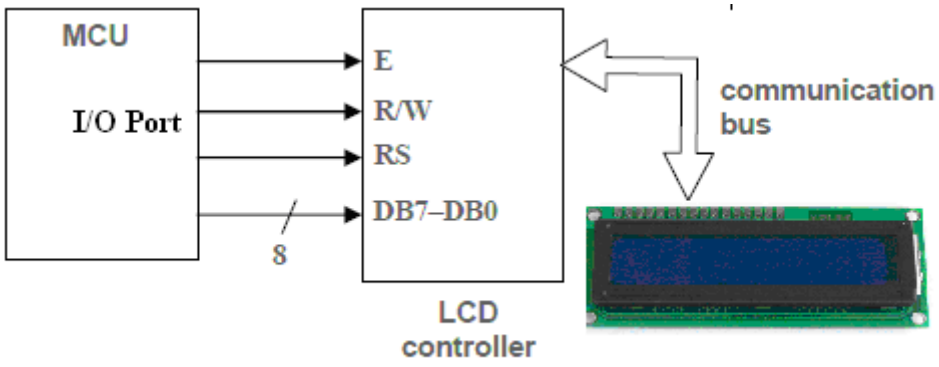
Modify your code so you can toggle the led every time the switch is pressed.

Turn-in:

- Code and demo.

LCD Configuration

Open the LCD datasheet and try to understand the signal sequence and timing metrics for your LCD and verify the LCD requirements and MCU Specs. Most LCDs are connected with the MCU as shown in Figure 4.



E= Enable, R/W = Read Write, RS= Command or data

Figure 4: LCD Connection.

First initialize the LCD using the provided sequence, this is done only once in the process.

Use the timing protocol to convey with the LCD for having more modular software, it is better to create subroutines for each one of the LCD commands. One for initialization, one for sending messages, etc. in this point you should be see blinking a cursor on LCD.

Write efficient string manipulation routines for displaying messages and data on the LCD.

Test all your LCD subroutines; be sure to have at least functions for the following operations: Clear LCD, Set cursor, Write a character, and write a command.

Turn-in:

- Code and demo.

Exercise:

Make a short program to display “Hello World” in your LCD.

Homework

Scroll List

For next week bring a scrolling list on the LCD, connect the LCD and two push buttons as shown in Figure 5.

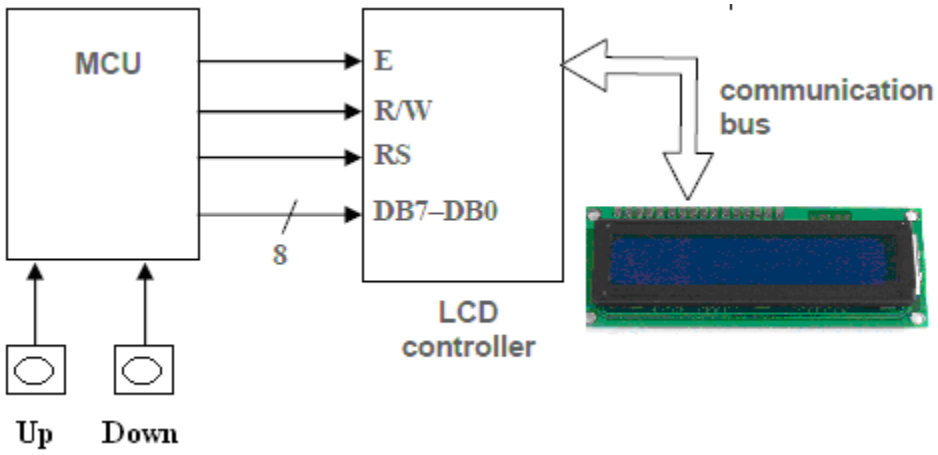


Figure 5: Scroll list connection diagram.

The scroll list must have alt least 16 entries (you can choose the messages).

With the UP and DOWN keys you should be able to scroll messages of the.

Turn-in:

- Software plan and explanation (pseudo code or flowchart)
- Connection schematic with components calculations
- Listing of code
- Have your circuit assembled and produce a demo to the TA or professor.