17.383 Microprocessors A
Lab 3
Fall 2010

Interfacing the PIC16F684 with the Seven-Segment LED

Objectives

1) To use MPLAB IDE software, PICC Compiler, and external hardware to demonstrate the following:

    a. Construct a breadboard containing a voltage regulator circuit to power the MCU
    b. Construct a circuit containing the PIC16F684 and a dual seven-segment LED display
    c. Develop Code to interface the PIC16F684 and the dual seven-segment LED display
    d. Counting and displaying switch actions

Materials

☐	1	Computer (PC) with an available USB Port
☐		MPLAB IDE and HI-Tech PICC software installed on PC
☐	1	PICkit ™ 1 with USB cable
☐	1	PIC16F684 – supplied with PICkit ™ 1 Starter Kit
☐	1	Oscilloscope
☐	1	Voltmeter
☐	1	Breadboard
☐	1	DC Power Supply or 9 volt battery
☐	1	9 volt battery clip (if using a 9 volt battery)
☐	1	LM7805 Voltage Regulator
☐	1	0.01 microfarad capacitor (103)
☐	1	0.33 microfarad capacitor (334)
☐	1	0.1 microfarad capacitor (104)
☐	1	LED
☐	1	Dual 7-segment LED (common anode)
☐	2	2N6519 PNP Transistors
☐	9	470 ohm resistors (yellow violet brown)
☐	2	SPST switch

WARNINGS AND PRECAUTIONS

1) Never remove the PIC16F684, or any other MCU, from an energized circuit

2) Do not construct circuits while energized

3) Follow electrical safety precautions

Source File Locations

1) HI-TECH Universal Toolsuite

   □ HI-TECH ANSI C Compiler

      C:\Program Files\HI-TECH Software\9.70\bin\picc.exe

2) Microchip MPASM Toolsuite

   □ MPASM Assembler (mpasmwin.exe)

      C:\Program Files\Microchip\MPASM Suite\mpasmwin.exe

   □ MPLINK Object Linker (mplink.exe)

      C:\Program Files\Microchip\MPASM Suite\mplink.exe

   □ MPLIB Librarian (mplib.exe)

      C:\Program Files\Microchip\MPASM Suite\mplib.exe

Background Information

Thus far this semester we have utilized the features of the PICkit 1 Starter Kit (LEDs, switch, and the power circuit). There are a limited number of circuits that can be constructed using the PICkit 1 Starter Kit; therefore we will start constructing circuits utilizing a breadboard. To protect the MCU from over voltage conditions, we will construct a voltage regulator circuit on the breadboard.

One method of displaying information is the use of a seven-segment LED display. Utilizing the PIC16F684 Input/Output ports, we are able to interface the MCU with the seven-segment display. Hardware interfacing techniques will be explored. Code will be developed which will control the devices. The basic ten digits can be displayed by turning on various segments of the seven-segment LED. Additional characters, those used for hexadecimals (A, B, C, D, E, and F), can be produced using different segments. In addition, there is a decimal point available.

Seven-segment LEDs come in common anode and common cathode configurations. We will be utilizing the common anode devices.

In addition to investigating the seven-segment LEDs, we will count the number of times a switch is activated and display the value on the seven-segment LEDs. A count reset switch will also be included in the circuit.

Additional hardware is required to connect additional dual seven-segment LEDs. We will investigate the additional hardware required.

Pre-Lab Preparation

1. Develop the required code as outlined in the procedural steps.

Procedure

Experiment 1. VOLTAGE REGULATOR CIRCUIT

a. Review lab Warnings and Precautions.

b. Construct the schematic shown in figure 1 on your breadboard. **DO NOT INCLUDE JP2.** Some guidelines for your construction follows:

1. Locate your voltage regulator circuit in a corner of your board as this circuit will be required throughout the semester.

c. Apply 9 volts to the VIN connection.

d. Using the voltmeter and an oscilloscope, verify that the +5V_REG port reads +5 volts and the output is clean and free of interference.

e. Construct the circuit shown in Figure 2. This will be a visual aid in determining if voltage is present (when the LED is turned on).

f. Apply 9 volts to VIN connection and verify that the LED lights.

g. Power down the circuit.

Experiment 2. DUAL SEVEN-SEGMENT LED DISPLAY

a. Construct the schematic shown in Figure 3 on your breadboard. Some guidelines for your construction follows:

1. The location for your PIC16F684 should be such that you can construct additional circuits on the board.

2. Do not install the PIC16F684 on the board at this time.

3. Keep wiring as neat as possible ... this reduce EMI and will aid in troubleshooting, if required.

4. The seven-segment LEDs may be required for possible future use; therefore you may consider placing these components on the board so that they can remain in place.

b. Seven Segment Display Circuit Connections verification prior to installing the PIC16F684 ...

1. Study the circuit diagram shown in Figure 3. Answer the following questions (this will aid in the software development).

   a. Identify which MCU pin controls the selection of each 7-segment digit. Fill-in the following table:

MCU Pin Name	Select Digit
RA1	D1
	D2

   b. Identify which MCU pin controls each of the 7-segments of each digit. Fill-in the following table:

MCU Pin Name	Segment
RA5	a
	b
	c
	d
	e
	f
	g

c. As a good practice, you should place comment statements into your Code to describe the software/hardware configuration. From the above determination, your can develop your comment statements. Fill in the blanks below:

```
/*****************************************/
/* ------------------------------------------------- */
/* Software/Hardware Interface:                      */
/* ------------------------------------------------- */
/*                                                   */
/* Select Right Digit using          >> RA0      */
/* Select Left Digit using           >> ___      */
/*                                                   */
/* Segment a         >> RA5                          */
/* Segment b         >> ___                          */
/* Segment c         >> ___                          */
/* Segment d         >> ___                          */
/* Segment e         >> ___                          */
/* Segment f         >> ___                          */
/* Segment g         >> ___                          */
/* ------------------------------------------------- */
```

c. Make the following checks without having the PIC16F684 installed (this will verify if there are any circuit problems ***prior to*** *installing the MCU*.

1. Power on the circuit (***without*** *having the PIC16F684 installed*) and verify that the power on LED is energized.

2. Ground the MCU connection that will select Digit 1 (as determined in the table identified in 1a above).

    a. Ground each connection identified in the table identified in 1b above. Verify that the corresponding segment lights on the left seven-segment LED. If the segment does not light, troubleshoot your circuit.
    b. Repeat for the remaining LED segments for Digit 1.
    c. Remove all grounds used to test the circuit

3. Ground the MCU connection that will select Digit 2 (as determined in the table identified in 1a above).

d. Repeat 2.a through 2.c. and verify that the right seven-segment LED segments properly light.

d. Power down the circuit.

Experiment 3. CODE DEVELOPMENT AND HARDWARE OPERATION FOR THE SEVEN-SEGMENT DISPLAY

a. Code will be developed such that the circuit performs as follows:

1. The circuit is hardwired such that only one digit should be lit at a time.

    a. Explain why only one digit should be energized at a time.

    The required code must be written to compensate for the hardware construction (only one digit should be lit at a time).

2. When the circuit is first energized, the seven-segment LEDs will start counting at 0x00

3. The circuit will count up to 0xFF, ***increasing by 1 each time.*** The count MUST BE sequential. It is unacceptable to count 0 to F to 0 on digit 2, then increase digit 1 by 1. The count shall perform like a counter (0x00 to 0x0F then 0x10 etc.)

4. Once the count reaches 0xFF, the count will start over at 0x00

5. Code will contain sufficient delay between incrementing the count so that the count can be visually confirmed that the circuit/code is operating as designed

b. Code development … the following will walk you through the required Code development. If you choose to develop code in another manner, you must describe in detail your methodology using the below tables to record your hardware/software configuration:

1. Determine which LED individual segments (which of the 7 segments make up the desired number) need to be lit for each hexadecimal digit (0 thru F). D will be displayed in lower case (d). Remember

that we have common anode, therefore to light each segment we need to apply a ground (0) to that segment. The following table will capture the needed data:

Hex #	Seven Segment Energized 1 = off 0 = on							Binary Equivalent
	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	1	0b0000001
1								
2								
3								
4								
5								
6								
7								
8								
9								
A								
B								
C								
D								
E								
F								

2. From the information acquired in 3.b.1, create an array for each hexadecimal digit. Zero is filled out for you. Use the following form and name ...

```
const char LEDDigit[] = {
        0b0000001,              //  "0"
        0b          ,           //  "1"
        .
        .
        .
        0b          },          //  "F"
```

Record your code in the space provided in the Lab Report.

3. Determine which Port/bit is used for displaying each digit and segment along with the direction ... Input/Output (I/O).

Determination can be made by examination of the schematic. Record results in the table provided in the Lab Report Template.

	Port associated with each LED Segment							Digit Selection	
	a	b	c	d	e	f	g	D1	D2
Port/Bit	RA5								
Direction	O								

4. Rearrange the data recorded above in the following table. This will aid in the determination of TRISA and TRISC statements ….

PORT	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	TRISA	TRISC
A	0						0b	
C								0b

Resulting in the following Code …

TRISA = 0b_ _ _ _ _ _; //_____ are Outputs
TRISB = 0b_ _ _ _ _ _; //_____ are Outputs

5. Only one digit can be lit at one time, therefore you must light one then light the other and back to the first. This alternating sequence will be repeated over and over to give the appearance that both the displays are active at the same time. To avoid flickering, you must run the entire sequence at least 50 times per second. This means that for two seven digit displays, each display can be active for a maximum of 10 ms.

((1 second/50 times per second)/2 digits) = 10 ms

6. Due to the complexity of the required operation, Figure 6 is provided which contains most of the Code required for this portion of the lab. Your Lab Report Template requires you to study each portion of the given code and provide an explanation of what it is actually accomplishing.

c. Write your code to perform the required operation using Figures 5, 6, and the Code created earlier in the lab (Figure 6 contains identifiers for placement of Code created earlier in the lab).

d. Save the program as Lab_3A.c.

e. Build you code. Once successful, burn the code to the PIC16F684 using the PICkit 1 Starter kit.

f. POWER DOWN the development board and remove the PIC16F684.

g. IMPORTANT --- Verify power is secured from the breadboard.

h. Install the PIC16F684 into the circuit designed in steps one and two above.

i. Power up the circuit and verify proper operation. Troubleshoot as required.

j. If required, when the software and hardware are operating properly, notify the instructor so it can be observed. Annotate observance in your Lab Notebook.

k. When the hardware is functioning as desired, printout a copy of Lab_3A.c. Submit the printout or paste a copy in your Lab Report.

Experiment 4. COUNTING AND DISPLAYING SWITCH ACTIONS

a. Modify the design in Experiment 3 as follows:

1. Add two switches to the hardware design. Connect the switches to RA3 and RA4. Figure 4 contains an example of a proper switch wiring schematic.

2. The count will start at 0x00 and cannot advance any higher than 0xFF.

3. Each time the switch connected to RA3 is actuated; the count will increase by one.

4. The switch connected to RA4 will be used to reset the count to zero. When this switch is actuated, the count will reset back to 0x00.

b. You code will contain the course header information. Refer to Figure 5.

c. Save the program as Lab_3B.c.

d. Build you code. Once successful, burn the code to the PIC16F684 using the PICkit 1 Starter kit.

e. POWER DOWN the development board and remove the PIC16F684.

f. IMPORTANT --- Verify power is secured from the breadboard.

g. Install the PIC16F684 into the circuit designed in steps one and two above.

h. Power up the circuit and verify proper operation. Troubleshoot as required.

i. If required, when the software and hardware are operating properly, notify the instructor so it can be observed. Annotate observance in your Lab Notebook.

j. When the hardware is functioning as desired, printout a copy of Lab_3B.c. Submit the printout or paste a copy in your Lab Report.

Summary:

We have covered the following topics during this lab:

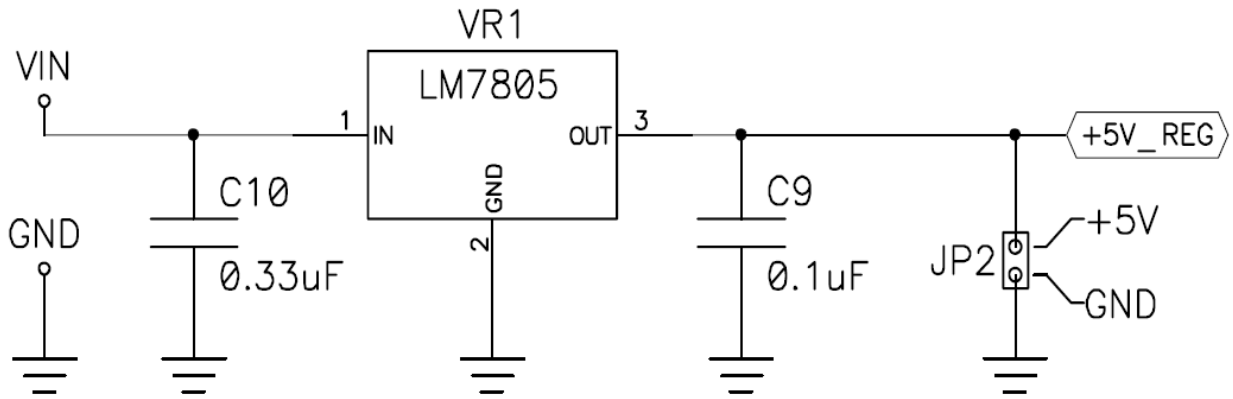
1. Voltage regulator circuit for use with the PIC16F684 when not installed in the PICkit 1 Starter kit
2. Seven-segment LED displays
3. Switch interfacing with the PIC16F684 MCU
4. Code development for the interfacing of the PIC16F684 with digital components

Many applications will require the control of the digital microcontroller ports. By understanding how to control these ports, any digital device can be connected and controlled by the microcontroller.

Questions:

1. Refer to the Lab Report Template.

Figure 1 – Voltage Regulator Circuit



7805 Pin Connections - Top View

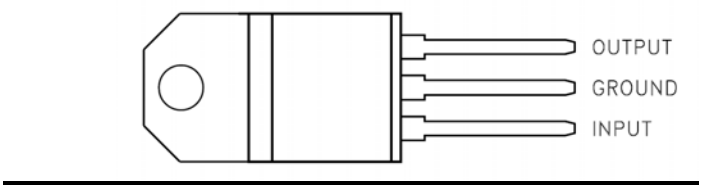
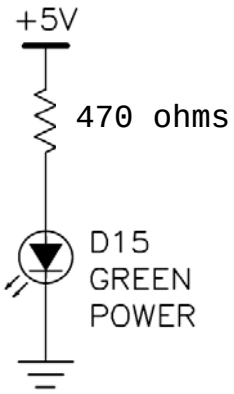


Figure 2 - PICkit 1 Starter Kit LED Layout



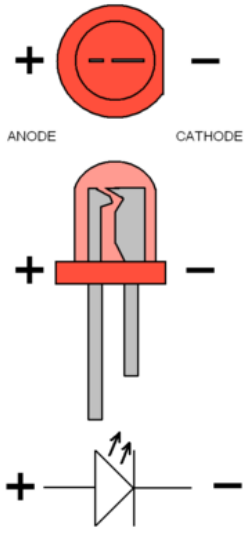
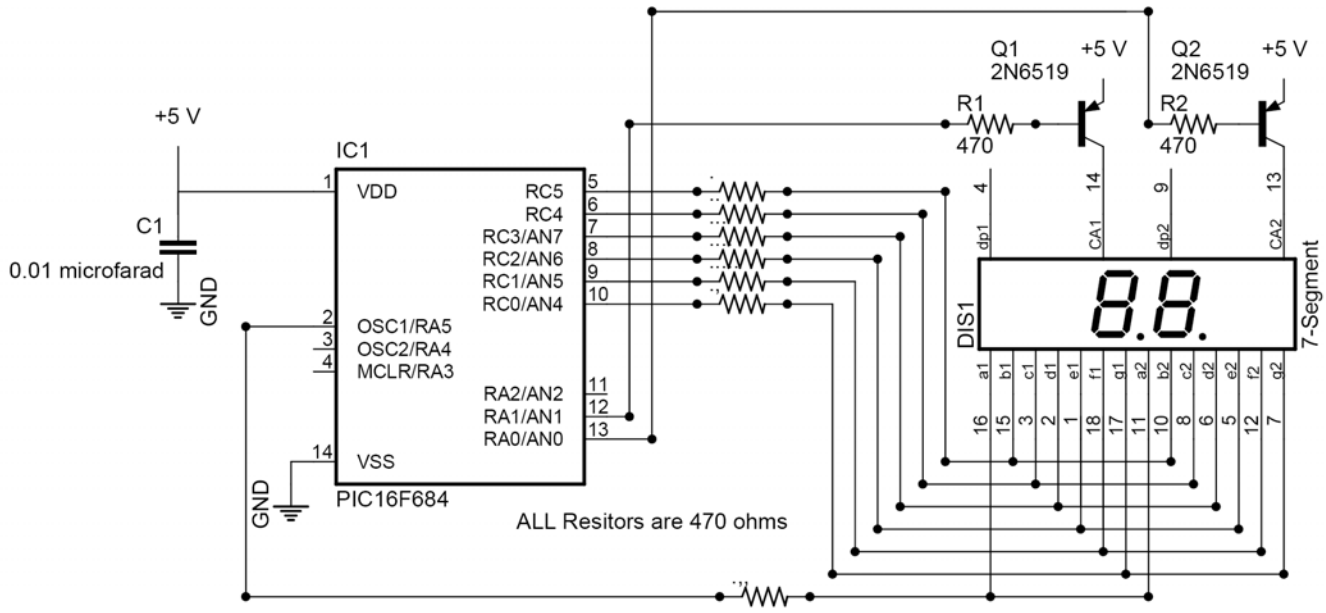
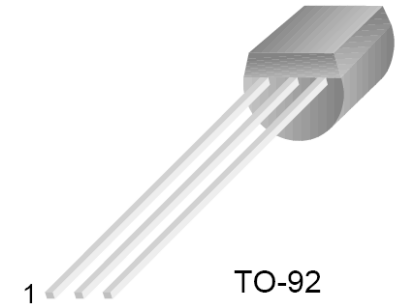


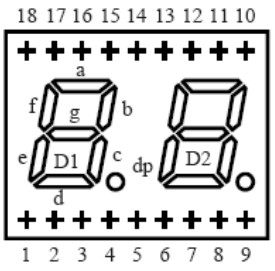
Figure 3 – Dual Seven-Segment LED Display Circuit



Terminal Connection



1. Emitter 2. Base 3. Collector



Pin No.	Assignment	Assignment
1	Cathode e1	Anode e1
2	Cathode d1	Anode d1
3	Cathode c1	Anode c1
4	Cathode dp1	Anode dp1
5	Cathode e2	Anode e2
6	Cathode d2	Anode d2
7	Cathode g2	Anode g2
8	Cathode c2	Anode c2
9	Cathode dp2	Anode dp2
10	Cathode b2	Anode b2
11	Cathode a2	Anode a2
12	Cathode f2	Anode f2
13	Common Anode D2	Common Cathode D2
14	Common Anode D1	Common Cathode D1
15	Cathode b1	Anode b1
16	Cathode a1	Anode a1
17	Cathode g1	Anode g1
18	Cathode f1	Anode f1

Figure 4 - PICkit 1 Starter Kit - Switch (SW1) Schematic

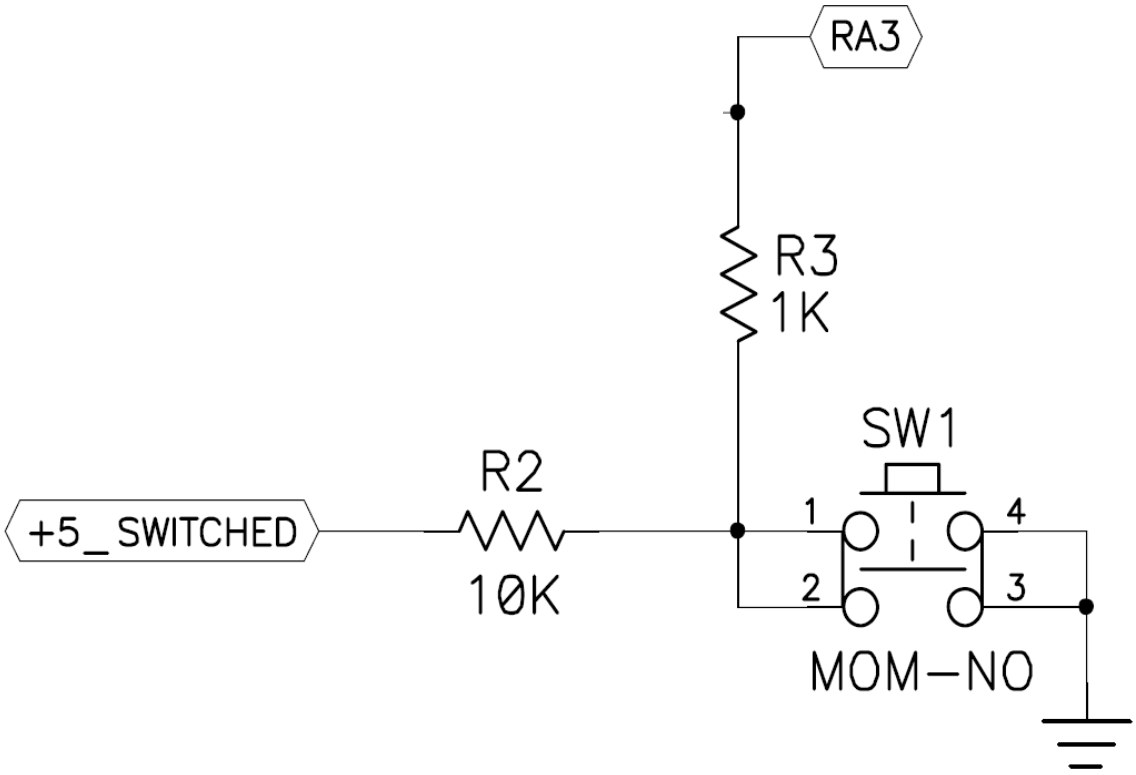


Figure 5 - Required Header Code

```
/*******************************************************
*
* Program Title:     xxxx
*
* Program File Name: Lab_XA.c
*
* Microprocessors A 17.383
*
* xxxxxxxx - Put in Semester (i.e. Fall 2010) here
*
* xxxxxxxx - Put in your name here
*
* xx/xx/xx - Put date here
*
*******************************************************/
```

Figure 6 - Code for Experiment 3

```
#include <pic.h>
/*

PLACE Software/Hardware Interface statement HERE


*/

__CONFIG(INTIO & WDTDIS & PWRTEN & MCLRDIS & UNPROTECT \
  & UNPROTECT & BORDIS & IESODIS & FCMDIS);


int i, j;
int DisplayValue, DisplayLED;

PLACE LEDDigit ARRAY HERE


main()
{

    PORTA = 0;
    PORTC = 0;
    CMCON0 = 7;                 //  Turn off Comparators
    ANSEL = 0;                  //  Turn off ADC
    PLACE TRISA STATEMENT HERE
    PLACE TRISC STATEMENT HERE

    DisplayValue = 0;           //  Start Displaying at 0x00
    DisplayLED = 0;             //  Display the 1s first

    while(1 == 1)               //  Loop Forever
    {
        if (0 == DisplayLED)    // True, then display right digit
        {
            RA5 = LEDDigit[DisplayValue & 0x0F] >> 6; // Clears display bits 4 - 7 of
                                                      // DisplayValue, then selects bit 7 of LEDDigit

            PORTC = LEDDigit[DisplayValue & 0x0F] & 0x03F; // clears display bits 4 - 7 of
                                                     //DisplayValue, then selects bits 0 - 6 of LEDDigit
        }
        else
        {
            RA5 = LEDDigit[(DisplayValue >> 4) & 0x0F] >> 6;
            PORTC = LEDDigit[(DisplayValue >> 4) & 0x0F] & 0x03F;
        }  //
        TRISA = TRISA ^ 0b011111;                   //  Swap Left/Right
        PORTA = PORTA & 0b111100;                   //  Make Sure Bits are Low
        DisplayLED = DisplayLED ^ 1;                //  Other Digit Next

        NOP();                                  //  Used for 10 ms Timing
        for (i = 0; i < 660; i++);              //  10 ms Delay Loop
        NOP();                                  //  Used for 10 ms Timing

        j = j + 1;                              //  Increment the Counter?
        if (25 == j)                            //  1/4 Second Passed?
        {
            DisplayValue++;                     //  Increment the Counter
            j = 0;                              //  Reset for another 1/4 Second
        }  //
    }  //
}  //
```