CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Spring 2019-2 Microcontroller Lab Felix Pinai

LAB 4: A/D converter, Temperature Sensor & Light Sensor

Build the circuit shown on the attached schematics. You will need to get the following datasheets:

For the device at the location marked 'U2':

http://www.ti.com/lit/ds/symlink/lm34.pdf

Check on both devices that I have provided to your team in class and look for the part that has the indicator 'LM34' printed on it.

The other part is the LM4040. The part may have the indicator '4040' on it or it may not, depending on the supplier of that part. Since I am going to provide you in the class with these two parts and as long as you have identified the LM34 part, then the second part is the LM4040. It should be located at 'U3' and its spec can be found at:

http://www.ti.com/lit/ds/symlink/lm4040-n.pdf

Make sure that you pay attention on the datasheets about how to determine the pinouts of those devices especially that the pictures shown on the documents do indicate that the views are for BOTTOM VIEW. Failure to properly connect the pins could result into damaging the parts especially the LM34! The view on the schematics shows the TOP view of the parts.

The LM4040 is to provide a fixed voltage of 4.096V while the LM34 is to measure the temperature by outputting the results into equivalent voltages. From the datasheets, pay attention to the relationship between the temperature in degree Fahrenheit and the voltage output.

The goal of this lab is to write a program that will perform the following steps:

1) Read the voltage from the LM34 using the PIC18F4321's ADC Channel 0

2) Convert that voltage into the equivalent temperature (in degree F)

3) Convert the temperature into BCD

4) Output the range of result onto to two 7-segment displays.

5) Output the same result to the TeraTerm terminal so that we can see the data on the computer screen

6) In addition, the circuit does include two **common cathode** RGB LEDs called D1 and D2. Get the program to do the following:

a) D1 will show different colors based on the following conditions:

Temperature Range	D1's color
Below 65F	Off
65F – 72F	Red
73F – 76F	Green
77F – 83F	Blue
Above 83F	White

b) D2 will show the range of the temperature as follows:

Temperature Range	D2's color
Below 10	Off
10-19	Red
20-29	Green
30-39	Yellow
40-49	Blue
50-59	Purple
60-69	Cyan
Above 70	White

7) Measure the voltage at the pin AN1 coming from the photoresistor PR1. Change the color of the **Common-Anode** D3 based on the following voltage reading:

AN1 voltage	D3's color
< 2V	RED
>=2V && < 3V	GREEN
>= 3V	YELLOW

Guidance/Help:

1) ADCON0 and ADCON1

To convert the voltage into digital readings, you will need to learn how to use the Analog/Digital Converter:

First, you need to initialize the A/D Converter. Chapter 19 of the PIC18F4321 Datasheets (http://ww1.microchip.com/downloads/en/DeviceDoc/39689b.pdf) talks about this piece of hardware. You need to pay attention to the two registers ADCON0 and ADCON1.

The first register (ADCON0) is to select which pin to use in the A/D conversion (See page 229 of the datasheets). The output from the temperature sensor LM34 is connected to AN0 or RA0. You need to set the proper value for ADCON0 in order to use AN0. In addition, you need to set bit 0 to '1' and bit 1 to '0'.

The second register ADCON1 (see page 230) is to specify what pins to be used as analog (for A/D purpose) versus digital. Read the definitions of the ADCON1. The lower 4 bits specify which pins are to be used as analog or digital. In our application here, AN0 through AN3 are to

be analog. Choose the minimum configuration to force those pins to be analog while the remaining pins must be set as digital.

Next, bits 4 and 5 specify the source of the two reference voltage pins. Use the settings that pick VREF+ instead of VDD for bit 4 and VSS for bit 5.

Follow the descriptions on the comment lines below to determine what value you should use for each register

```
void Init_ADC(void)
{
        ADCON0=0x??;        // select channel AN0, and turn on the ADC subsystem
        ADCON1=0x??;        // select pins AN0 through AN3 as analog signal, VDD-VSS as
                            // reference voltage
        ADCON2=0xA9;        // right justify the result. Set the bit conversion time (TAD) and
                            // acquisition time
}
```

2) Measuring the Analog Voltage

You will need to use the following routine to get the result (this routine was in your tutorial):

```
unsigned int get_full_ADC(void)
{
int result
        ADCON0bits.GO=1;                        // Start Conversion
        while(ADCON0bits.DONE==1);              // wait for conversion to be completed
         result = (ADRESH * 0x100) + ADRESL;    // combine result of upper byte and
                                                // lower byte into result
        return result;                          // return the result.
}
```

3) Temperature Conversion

Read the spec of the LM34 to determine how to convert the voltage reading into equivalent temperature.

4) Display Temperature into 7-segment displays

Once you get the digital readings of the temperature, you will need to break that value into two numbers, one for the upper digit and the other one for the lower digit.

After the two BCD digits are obtained, you need to convert each digit into a hex value that decodes the value of that BCD digit into 7-segment display. Create a bitmap for each digit as follows: (Remember that the 7-segment is a common anode and therefore a logic 0 with turn on the segment while a logic 1 will turn it off).

BCD Digit	PORT 7 6 5 4 3 2 1 0 g f e d c b a	HEX number
0	x 1 0 0 0 0 0 0	0x40
1	x 1 1 1 1 0 0 1	0x79
2		
3		
4		
5		
6		
7		
8		
9		

Refer to the following link to see all the combinations of the BCD digits (scroll down to 'Displaying letters'):

https://en.wikipedia.org/wiki/Seven-segment_display

After you have generated all the patterns, place all the values into an array. Use the digit of the number to be displayed as the offset to the array in order to obtain the bit pattern for that digit. Output the bit pattern to the Port associated with the 7-segment display. You should use the same array to get the pattern for both upper digit and lower digit of the 2-digit number.

5) Display on the TeraTerm

To debug and troubleshoot your software, you may have to use the serial port and the TeraTerm software to print message on the screen. In order to get the serial port to work, you will need to have the following sections added into your code.

First, you have to make sure that the following lines appear at the beginning of the program:

```
#include <p18f4321.h>
#include <stdio.h>
#include <math.h>
#include <usart.h>

#pragma config OSC = INTIO2
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOR =OFF
```

Second, you do need to add this routine at the start of the program (preferably before the main() routine):

```
void init_UART()
{
        OpenUSART (USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_HIGH, 25);
        OSCCON = 0x60;
}
```

Next, at the start of the main() routine, you have to add to call the init_UART() routine above.

To print something out on the serial port, you have to use the 'printf' statement. You can 'google' for the proper use of this printf statement. For example, to print the content of a variable 'x' with a heading in from of the data, you can have something like:

```
printf (" x =  %d \r\n",x);
```

Notice the use of %d and the control function '\r' and \n'. You need to revisit those control functions in the 'printf' statement to know how to use them.

Here are the lines that you should have all together:

```
#include <p18f4321.h>
#include <stdio.h>
#include <math.h>
#include <usart.h>

#pragma config OSC = INTIO2
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOR =OFF

void putch (char c)
{
   while (!TRMT);
   TXREG = c;
}

void init_UART()
{
        OpenUSART (USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_HIGH, 25);
        OSCCON = 0x60;
}
```

```
void main()
{
        init_IO();
        init_UART();

// Place the rest of your code here
}
```

Note: Remember to set the 'Link in Peripheral Library' as you have done in LAB1 Part 2.

Here are the steps from that lab:

- Select the project and right click on it. Scroll all the way down to Properties and click on it.

- A new screen will appear. Select 'XC8 linker' and then go the right side and scroll until you see the selection. Check on that option. Hit OK afterwards.

6) Set different colors for the RGB LED

One convenient way to set an individual output is to use the '#define' statement. For example, we can have the following:

```
#define RED         PORTBbits.RB0
#define GREEN       PORTBbits.RB1
#define BLUE        PORTBbits.RB2
```

Once you have defined those variables, you can force a color upon the RGB LED by applying the proper logic to each variable to get the desired color. For example, to get:

```
void SET_RED()
{
RED = 1;
GREEN = 0;
BLUE = 0;
}

void SET_YELLOW()
{
RED = 1;
GREEN = 1;
BLUE = 0;
}
```

Do the same for the other colors.