CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Spring 2025 Microcontroller Lab Felix Pinai
Session 1

LAB 5: A/D converter, Temperature Sensor & Light Sensor

Build the circuit shown on the attached schematics. You will need to get the following datasheet for the device at the location marked 'U2':

https://www.ti.com/lit/ds/symlink/lmt84.pdf?ts=1695487852813&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US%2526searchTerm%253DLMT84%2526nr%253D135

Make sure that you pay attention on the datasheets pdf page 4 with the package TO-92 about how to determine the pins of this device especially that the pictures shown on the document do indicate that the views are for TOP VIEW. Failure to properly connect the pins could result into damaging the parts especially the LMT84! The view on the schematics shows the same TOP view of the parts.

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

1) Read the voltage from the LMT84 using the PIC18F4620's ADC Channel 0 connected to AN0 (also called RA0).

2) Calculate the temperature in degree C.

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

4) Display the temperature in degree F onto to two 7-segment displays. See the schematics to get the connections of each of the 7-segment display.

5) Output the same result to the TeraTerm terminal so that we can see the data on a computer screen. Display on screen the voltage readout of the LMT84 and the temperatures in degree F.

6) Use of a photo sensor to measure the amount of light and display it the voltage readout of that sensor onto the terminal.

7) In addition, the circuit does include three **common cathode** RGB LEDs called D1, D2 and D3 used to display different statuses. Look at the schematics to determine the connections to those three RGB LEDs.

Get the program to do the following:

a) The RGB LED D1 will show the range of the temperature as follows:

Temperature Range	D1's color
Below 10 F	Off
10-19 F	Red
20-29 F	Green
30-39 F	Yellow
40-49 F	Blue
50-59 F	Purple
60-69 F	Cyan
>=70 F	White

Write the 'c' code in the most efficient way for this part b)! Even though you can use 'if statement' or 'case statement' to check the conditions, find a more direct implementation to control the LED. Look closely at the relationship between the ranges of the temperature and the associated color to derive a simple formula to generate the required table of color. The code implementaion will not be more than 5 lines of codes.

b) The RGB LED D2 will show different colors based on the following conditions:

Temperature Range	D2's color
Below 32F	Off
32F – 42F	Red
42F – 52F	Green
52F – 62F	Yellow
62F – 72F	Blue
72F – 77F	Purple
77F – 80F	Cyan
Above 80F	White

c) Measure the voltage at the pin AN1 coming from the photo resistor PR1. Change the color of the RGB LED **Common-Cathode D3** based on the following voltage reading:

AN2 voltage	D3's color
< 2.6V	WHITE
>= 2.6V && <2.9V	BLUE
>=2.9V && < 3.2V	GREEN
>=3.2V && < 3.6V	RED
>= 3.6V	OFF

Note: to measure the voltage for the pin AN4, don't forget to select that pin in register ADCON0

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 PIC18F4620 Datasheets (http://ww1.microchip.com/downloads/en/devicedoc/39626e.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 pdf page 225 of the datasheets). The output from the temperature sensor LMT84 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 force bit 0 to '1' and bit 1 to '0'. Here is the general routine to program ADCON0 based on the Analog pin used:

```
void Set_ADCON0 (char AN_pin)
{
        ADCON0 = AN_pin * 4 + 1;
}
```

The second register ADCON1 (see pdf page 226) 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
}
```

Once the above routine is defined, call that routine at the beginning of the main program to initialize the ADC pins.

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.
}
```

This function will provide the number of steps in the A/D conversion. To get the actual voltage, use the number of steps and multiply by the voltage per step (This is 4mV/step).

3) Temperature Conversion

From page 11 of the datasheet, here the equation of the voltage versus temperature.

V = (-5.50 mV / oC) T + 1035 mV

Rearrange the equation to get the temperature versus voltage that was calculated from step 2) above.

Once the temperature is calculated, then convert it into degree F.

4) Display Temperature into 7-segment displays

The provided dual-digit 7-segment has the following datasheet:

https://www.datasheet.live/index.php?title=Special:PdfViewer&url=https%3A%2F%2Fpdf.datasheet.live%2F31e73430%2Fledtech.com%2FLA5622-11.pdf

Once you get the digital readings of the temperature in degree C, you will need to convert it into degrees F.

Next, you will have to break that temperature into two numbers, one for the upper digit and the other one for the lower digit. For example, a value of 81F will have the upper value of 8 and the lower value of 1. To get the upper digit, jump simply take the temperature and divide it by 10. The lower digit is just modulo 10 of that temperature.

After the two BCD digits are obtained, you need to convert each digit from a decimal value into a value that decodes 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 logic 0 with turn on the segment while a logic 1 will turn it off). Here are two examples of the decoding for the numbers 0 and 1. Do the remaining numbers afterwards.

BCD Digit	PORT 7 6 5 4 3 2 1 0 a b c d e f g	HEX number
0	x 0 0 0 0 0 0 1	0x01
1	x 1 0 0 1 1 1 1	0x4F
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

Once all the values are obtained, gather them and place them into an array to be used to output to the corresponding port supporting the 7-segment display. For example:

char LED_7seg[10] = { 0x01, 0x4F, … };

Next, 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. Once that value is obtained, output it to the Port associated with the 7-segment display. Check the schematics to determine what port to use to output for the upper digit and for the lower digit.

5) Display on the TeraTerm

To debug and troubleshoot your software, you will 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 <p18f4620.h>
#include <stdio.h>
#include <math.h>
#include <usart.h>

#pragma config OSC = INTIO67
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOREN =OFF
```

Second, you do need to add the following two routines 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;
}

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

Next, at the start of the main() routine, you have a call to the init_UART() routine above. No need to have the call for the 'putch' routine because it will be used anytime you call the function 'printf'

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 <p18f4620.h>
#include <stdio.h>
#include <math.h>
#include <usart.h>
#pragma config OSC = INTIO67
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOREN =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_UART();
        Init_ADC();
        Init_TRIS();


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

Note:

1) Init_ADC() was the routine mentioned above.
2) Init_TRIS() – You will need to write this routine to setup all the TRIS registers in order to setup the pins that are inputs or outputs.
3) 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, look at the connections of the RGB D3 on the schematics. We can type the following:

```
#define D3_RED          PORTEbits.RE0
#define D3_GREEN        PORTEbits.RE1
#define D3_BLUE         PORTEbits.RE2
```

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 set the RED color for a common-cathode RGB LED, you can write this routine:

```
void SET_D3_RED()
{
        D3_RED = 1;
        D3_GREEN = 0;
        D3_BLUE = 0;
}
```

The same for setting the yellow color:

```
void SET_D3_YELLOW()
{
        D3_RED = 1;
        D3_GREEN = 1;
        D3_BLUE = 0;
}
```

Do the same for the other colors. Also, repeat the same procedure for the other RGB LEDs.

Once you have finished defining the routines to set the color, them implement the software to check the conditions and based on the decision, call the appropriate set color routine to activate the desired color.