CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Fall 2025 Session 2 **Felix Pinai**
Microcontroller Lab

LAB2
Basic Input/Output using Microcontroller parallel ports

To perform the lab below, you need to download the spec of the processor PIC18F4620 that we are using in the lab. Go to the following link:

http://ww1.microchip.com/downloads/en/devicedoc/39626e.pdf

Remember that this processor is an upgrade to the processor PIC18f4321. It has the same peripheral hardware but its internal ROM/RAMs are larger.

Download the datasheet and save somewhere (like Desktop) that you can easily have access to it.

MATERIALS:

Provided by instructor:

- 2 RGB LEDs

Student must get the following:

- (3) 1K or higher resistors
- (3) regular LEDs (any color)
- 1 bank of 4 minimum DIP Switches

PART 1)

Write a program that will input from a bank of 3 switches (DIP switches) and output each corresponding input to an equivalent LED. There would be a total of 3 LEDs. The input switches are connected from PORT A bits 0-2 while the output LEDs are tied to PORT B bits 0-2.

Hint:

a) Use the downloaded datasheet of PIC18F4620; go to chapter 9 'I/O Ports'. Study the definitions of the **TRISx** registers where 'x' represents the PORT name. Set

the right value of each bit in the TRIS register to set up whether the corresponding pin is an input or output.

A '1' will make the pin an input while a '0' makes it an output. Reading something from a switch to a pin of a port will make that pin an input. Controlling a LED from a pin of a port will make that pin an output. Based on those definitions, you should be able to determine what port should be set up as an input port or an output port. From there, you should be able to figure the value to be placed into the TRISx registers. In this case, you should deal with the registers TRISA and TRISB.

b) If you use PORTA or PORTB for the inputs and/or outputs, you need to study the definitions of the register **ADCON1** (see chapter 19 '10-bit Analog-to-Digital Converter A/D Module' in the datasheet) and its PCFG3:PCFG0 bits. You need to make sure that the pins from ports A and B are set up in digital mode and not analog meaning all the pins must be digital. We cover the purpose of this register on a later lab. For this lab, **we will load this register with the value of 0x0F**.

c) Write an endless loop to read the input switches and output the values to the LEDs. Based on the schematics shown on 'LAB2_Schematics.pdf', the three inputs should be port A bits 0 through 2 while the three LEDs should be connected to port B bits 0 through 2.

Example:

```
void main()
{
char in_sw;                         // Use variable 'in_sw' as char
        TRISA = 0x??;               // fill out the ?? with the proper values
        TRISB = 0x??;               // fill out the ?? with the proper values
        ADCON1 = 0x0F;              // Set the ADCON1 to this value to force
                                    // the pins to be in digital mode

        while (1)
        {
                in_sw  = PORTA;     // read data from PORTA and save it
                                    // into 'in'
                in_sw = in_sw & 0x07;// Mask out the unused upper four bits
                                    // while preserving the lower 3-bits
                PORTB = in_sw;      // Output the data to PORTB
        }
}
```

PART 2)

Connect the provided Common-Cathode RGB LED at D1 to the pins RC0 through RC2. Pin RC0 will be used to turn on the RED color of LED D1, RC1 is for the GREEN and RC2 is for the BLUE

Write an endless loop where the lowest three pins of the DIP Switch SW1 (pins connected to port A bits 0 through 2) are read and then these three bits are outputs to the port C whereas the following color will be generated on the LED D1:

Inputs (SW1)			Outputs (D1)				
RA2	RA1	RA0	RC2	RC1	RC0	Color	Color Value Reference
0	0	0	0	0	0	No light	0
0	0	1	0	0	1	Red	1
0	1	0	0	1	0	Green	2
0	1	1	0	1	1	Yellow	3
1	0	0	1	0	0	Blue	4
1	0	1	1	0	1	Purple	5
1	1	0	1	1	0	Cyan	6
1	1	1	1	1	1	White	7

Note: **Don't forget to add the TRISC command for this port C** in your code to force PORT C to be in output mode.

Show the operation of this part by changing the switch settings of SW1 and check that the color of the RGB LED D1 matches the combination of the switch. Make sure that the most significant bit of SW1 is on the left side of that switch and the LSB is on the right side.

PART 3)

Write an endless loop that will continuously generate the list of colors shown on Part 2) with about 1 second delay between each color. **There is no input to read**.

Hint:

Used the provided subroutine Delay_One_Sec() below:

```
#define         delay   17000

void Delay_One_Sec()
{
        for (int I=0; I <delay; I++);
 }
```

This delay routine is to generate the time delay. You would need an overall endless loop that will output the 8 different colors spaced by 1 second delay. To achieve this task, you will need to add a FOR loop with an 8-count value (count from 0 to 7). Use the counter value as the color of the output to be generated. In the FOR loop, after you have output the color to port C, add the Delay_One_Sec() routine to wait 1 second. Failure to have this delay will cause the colors to be generated at a very high rate so that your eyes cannot differentiate the changing of the color resulting in the effect that the LED will always on with an apparent WHITE color.

In addition, on the schematics, bit 0 of PORT C is shown to be connected to channel 0 of the logic analyzer (see signal RC0). We are going to use that signal to measure the time duration of the delay routine Delay_One_Sec() provided above. The variable 'delay' in that routine was assigned with an arbitrary value of 17000. That would provide about 1 second delay. To be more accurate, we will use the logic analyzer to change that value to have the exact duration of 1 second.

Run the analyzer software 'Logic 2' when Part 3 is running. Capture all the three channels LA0, LA1, and LA2 of the analyzer. Pay attention to channel 0. Click on the square waveform. The software will show the time length of the half period as well as the full period. If the period is not 1 second, change the value of the 'delay' in the routine 'Delay_One_Sec()' with a new approximate value, recompile the program, download the new one to the board and rerun the program. Execute another analyzer capture and perform again the measurement. Repeat the process until the recorded time is 1 second with a +/- **1 msec** tolerance.

PART 4)

Now, this part will add one additional RGB LEDs D2. D2 has three pins connected to **PORTD** bits 5 through 7.

Here is the sequence of colors for the LEDs D1 and D2.

Step #	**Color @RGB LED D1**	**Color @RGB LED D2**
0	White	Cyan
1	Purple	White
2	Cyan	Red
3	Yellow	Yellow
4	Green	Blue
5	No light (off)	No Light (off)
6	Blue	Green
7	Red	Purple

Due to the random assignments of the colors for both D1 and D2, it would be best to create arrays of values for each step value and use the FOR loop to output the color value for each D1 and D2 on each step.

To create an array of value, for each step, find the assigned color (see Part 2) Associate that color with the color value. Gather all the eight values and then create the array:

char array1[8] = {value1, value2, ….};

whereas value1 and value2 are the color values

After both arrays are created for D1 and D2, use the index of the FOR loop as the offset of the array to retrieve the value of the colors for the steps. Output the values from the arrays to the associated PORTs. For example: PORTC = array1[i]; where 'i' is the index used in the FOR loop and 'array1' is the array of value created.

Hint: the RGB LED D2 has its three pins connected to PORT D. However, these pins don't start from bit 0 to make it easy to enumerate the binary value for the colors. To evaluate the binary value, look at the color and use its associated number as indicated on part 2). Next, since the LED D2 does not start at bit 0 but rather at a different offset, determine how many bits the LED is off by. Next, do a shift operation to the value of the array before outputting to the appropriate PORT. The number of bits to be shifted depends on the offset observed from the schematics.

PART 5)

Now we will use the logic analyzer probe to check our implementation. Connect the 6 channels 0 through 5 of the analyzer to the 6 pins that are the sources for the two RGB LEDs D1 and D2. See schematics.

Run Part 4). Capture the timing of the analyzer's channels. Make sure to adjust the capture window large enough so that all the eight steps of Part 4) are recorded. In addition, rearrange the channel number in the following order:

Ch2
Ch1
Ch0

Ch5
Ch4
Ch3

The reason is for each group of 3 signals it represents the three colors of each RGB LED. The reason for the order shown above is that the higher channel number is the most significant bit of the binary combination of the number.

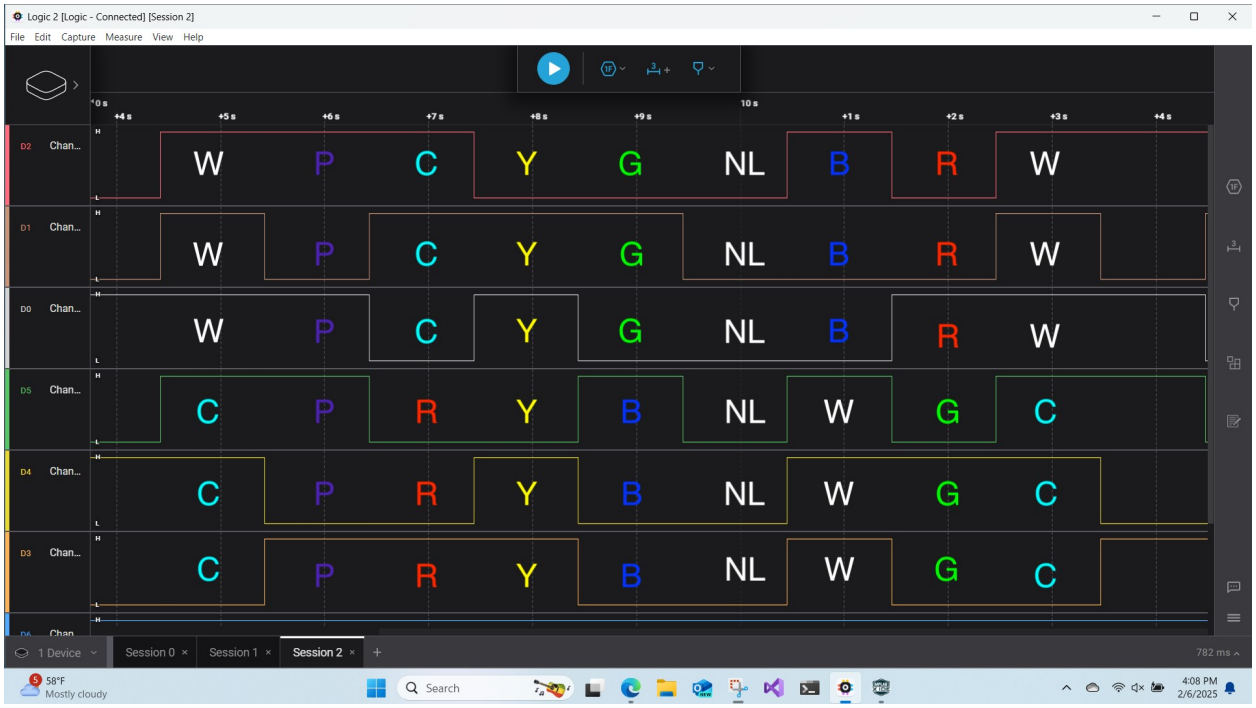
Once the capture is complete, use a marker to slide over each step and check the logic state of each channel and create the following table:

Step #	Ch2	Ch1	Ch0	Ch5	Ch4	Ch3
0	x	x	x	x	x	x
1						

‘x’ is the logic value observed.

Once the table is completed, translate the binary value into decimal value. Then, translate the decimal values into color values. Finally, verify that the recorded values match with the table indicated on Part 4).

Here is an example of the captured values. Notes: the example does not show the same values as the lab requires



General reminder:

Make sure that for this lab create a general folder marked as ‘lab2’. From there, create sub-folders one for each part. Since we are going to have five parts, then we should have five sub-folders marked for example lab2p1, lab2p2, … lab2p5.