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

MATERIALS:
- 2 RGB LEDs
- (4) regular LEDs (any color)
- 1 bank of 4 minimum DIP Switches
- (4) 1K or higher resistors

PART 1)
Write a program that will input from a bank of 4 switches (DIP switches) and output each corresponding input to an equivalent LED. There would be a total of 4 LEDs. The input switches are connected from PORT A bits 0-3 while the output LEDs are tied to PORT B bits 0-3.

Hint:
a) Use the downloaded datasheet of the 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 setup whether the corresponding pin is an input or output. A '1' will make the pin an input while a '0' makes it an output.

b) If you use PORTA or PORTB for the inputs and/or outputs, you need to study the definitions of the register ADCON1 (see the 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 setup in digital mode and not analog.

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 four inputs should be port B bits 0 through 3 while the four LEDs should be connected to port A bits 0 through 3.

Example:

```
void main()
{
char in;                              // Use variable 'in' as char
        TRISA = 0x??;                 // fill out the ?? with the proper values
        TRISB = 0x??;                 // fill out the ?? with the proper values
        ADCON1 = 0x??;                // fill out the ?? with the proper values

        while (1)
        {
                in = PORTA;           // read data from PORTA and save it
```

```
                                    // into 'in'
            in = in & 0x0F;         // Mask out the unused upper four bits
            PORTB = in;             // 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
0	0	0	0	0	0	No light
0	0	1	0	0	1	Red
0	1	0	0	1	0	Green
0	1	1	0	1	1	Yellow
1	0	0	1	0	0	Blue
1	0	1	1	0	1	Purple
1	1	0	1	1	0	Cyan
1	1	1	1	1	1	White

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:
> Write the subroutine Delay_One_Sec() below:
> void Delay_One_Sec()
> {
>   for(int I=0; I <17000; I++);
> }

Once this delay routine is written, use it 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 the 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.

PART 4)
Based on the code on Part 3), add the Common-Cathode RGB D2 to the PORT D bits 0
through 2.

Take the code from part 3) and add the needed coded to generate the colors for the LED D2. Notice the pattern of the colors for D2 with respect to the ones for D1 so that there is a simple formula to generate those colors for D2.

Color @RGB LED D1	Color @RGB LED D2
No light (off)	White
Red	Cyan
Green	Purple
Yellow	Blue
Blue	Yellow
Purple	Green
Cyan	Red
White	No light (off)

PART 5)
Now, this part will have a different sequence of colors for the LEDs D1 and D2.

Color @RGB LED D1	Color @RGB LED D2
No light (off)	Blue
Red	White
Green	Purple
Yellow	No light (off)
Blue	Red
Purple	Cyan
Cyan	Green
White	Yellow

Due to the random assignments of the colors for D2, I suggest the use of an array to contain the proper values.