Laboratory #3
Simple I/O Port Interfacing

Objective: To use the PIC32 I/O ports for interfacing simple digital I/O devices such as pushbutton switches and LED displays.

Task:

The Explorer 16 Development Board comes with a few on-board I/O devices, the simplest of which are four push-button switches and eight LEDs. In this lab exercise you will be writing a C program to interface these simple I/O devices.

For this lab, you will be using two of the pushbutton switches (S3 and S4) for program input and the 8 LEDs for output. Pushbutton S3 is connected to RD6 (i.e. pin 6 of PORTD), and pushbutton S4 is connected to RD13 (i.e. pin 13 of PORTD). The 8 LEDs are connected to RA7..RA0 (i.e. the lower 8 pins of PORTA).

The pushbuttons are low-asserting; i.e. when you push a button it drives the corresponding logic signal low, otherwise the signal remains high. The LEDs are high asserting, i.e. sending a logic high signal turns on the light.

In this exercise, your program is required to monitor the status of pushbutton switches S3 and S4 and react to their activation as follows:

1. If S3 is pressed, your program enters a SET mode of operation and remains in this mode while S3 pressed. While the program is in the SET mode, it will allow the user to increment a counter by using S4. Every time you press and release S4 the counter will increment by one. Your program is required to display the current state of the counter on the eight LEDs as the count gets incremented.
2. When S3 is released, your program switches to a RUN mode of operation and remains in this mode as long as S3 is not pressed. While in the RUN mode, the program will count down the counter at a rate of about once per second. It will also continuously update the current state of the counter on the LED displays. Repeat the count down operation until the counter value reaches zero.

Set your program to run in an infinite loop executing the above sequence indefinitely.

While testing your program, observe the switch bouncing problem when you interact with the pushbuttons. You can avoid this problem by implementing a switch de-bouncing technique. Assume in the worst case the switch bounce lasts no more than 25 ms.

For this lab project, since we have not covered the timer module yet, you are free to use software idle loop to implement time delay routines.

MCU Configuration information: To help properly configure your PIC32 MCU for operation at 80 MHz clock frequency, include the following directives at the beginning of your program code:

```
// configuration bit settings, Fcy=80MHz, Fpb=40MHz
#pragma config POSCMOD=XT, FNOSC=PRIPLL
#pragma config FPLLIDIV=DIV_2, FPLLMUL=MUL_20, FPLLODIV=DIV_1
#pragma config FPBDIV=DIV_2, FWDTEN=OFF, CP=OFF, BWP=OFF

#include <p32xxxx.h>
```

Apparently, even with the above configuration of the clock frequency, other default settings of the PIC32 MCU actually introduce a few wait states when accessing the flash memory where the program resides (because the flash memory's access speed is lower that the 80 MHz clock), data RAM, peripheral bus, etc. So the actual instruction execution speed could be slightly lower than the system clock frequency.

The MPLAB C32 provides the following library function that you can use to get your program configure the CPU for optimum performance, minimizing or eliminating the wait states when possible.

SYSTEMConfigPerformance(80000000);

When using this function, you need to add the include file **<plib.h>** at the beginning of your program.

Notes:

- When you create your project remember to select the right device (PIC32MX360F512L).
- Use the "Debugger -> Select Tool" menu to choose the "REAL ICE" as your debugger.
- If you get error messages when MPLAB tries to connect to the REAL ICE, or program the MCU, or debug a program, try the following techniques to resolve the issues:
    - Reset the Explorer 16 board (by unplugging the power and plugging it back)
    - Reset the REAL ICE using its RESET button
    - Disconnect and re-connect the USB cable
    - If all the above steps do not fix the problem, re-download the firmware using the "Debugger -> Settings…" menu and then selecting the "Configuration" tab and pressing the "Manual Download" button. Choose the "RIFW_xxxxxx.jam" file from "C:\Program Files\Microchip\MPLAB IDE\REAL ICE" folder

Checkout and report:

Demonstrate your working program to the lab instructor and hand in a lab report that includes the following:

- Title page: course #, course title, Lab# & title, date, professor's name, names of group members
- This lab handout.
- Printout of the C source code.
- Printout of the disassembly listing file. Since the *SYSTEMConfigPerformance()* function assembles to many pages of assembly instructions, please comment it out when generating the disassembly listing file for your submission.