Lab Sheet 4 – Onsite

PIC16F84A Input/Output & C Language & Interrupts

1. Overview

In the previous three weeks, we have studied the instructions set of the PIC16F84A. The set is divided into:

1. Byte-oriented instructions that operate on bytes of data and further divided into:
    a. Data transfer instruction,
    b. Mathematical and logical instructions
2. Bit-oriented instructions that operate on a single bit.
3. Literal and control instructions.

In this week we will write our first program to be downloaded to and run on the PIC hardware kit. Furthermore, we'll study the concept of interrupts. Four sources of interrupts are available in the PIC16F84A microcontroller:

1. External interrupt (RB0) (studied in this lab).
2. Interrupt on change (RB4-RB7) (studied in this lab).
3. Timer0 interrupt.
4. EEPROM write complete interrupt.

2. Simple Input/Output Program

Program 1: First of all, create a project for this lab as was shown in the first week. Write the following program in the source code file:

```
        BSF     STATUS, RP0
        BSF     TRISA, 0        ; Setup RA0 as an input pin
        CLRF    TRISB
        BCF     STATUS, RP0
        MOVLW   0xFF
        MOVWF   PORTB
LOOP    BTFSS   PORTA, 0        ; Check if button is pressed
GOTO    NO                      ; If button is not pressed
        CLRF    PORTB
        GOTO    LOOP
NO      MOVLW   0xFF
        MOVWF   PORTB
        GOTO    LOOP
```

Code 1. Program 1 Code

After you successfully build your project (production build not debug build), you will download the .HEX file to the hardware board available in the lab in order to run the program on the PIC16F84A microcontroller. Please follow these steps:

1. The program used to download the .HEX file to the PIC16F84A on the hardware board is PicProg. You can run it by double clicking on the .EXE file in "C:\ProgPicSoftware\ \ProgPIC.exe"**.**

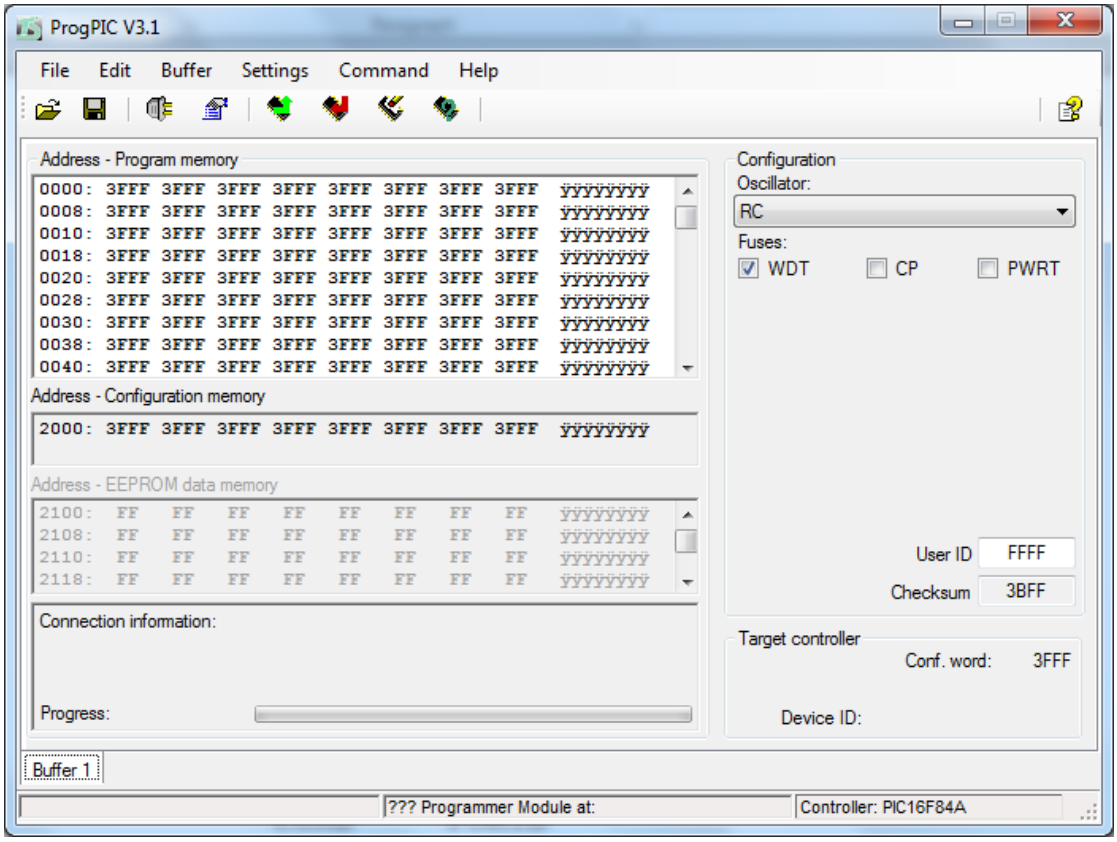

Fig. 1 The PicProg window.

2. Click "File→Open" to choose the .HEX file to be downloaded to the hardware kit, the contents of the .HEX file will appear in the "Address – Program memory" window as shown in Fig. 2 below.

3. **<u>If needed</u>**, change the configuration settings on the right to XT oscillator type, no WDT (WatchDog Timer) and no PWRT (Power-Up Timer) as shown in Fig. 2 below.

   Note that these settings could be adjusted in the .ASM file by manipulating the default line as follows:

   __CONFIG _CP_**OFF** & _WDT_**OFF** & _PWRTE_**OFF** & _**XT**_OSC

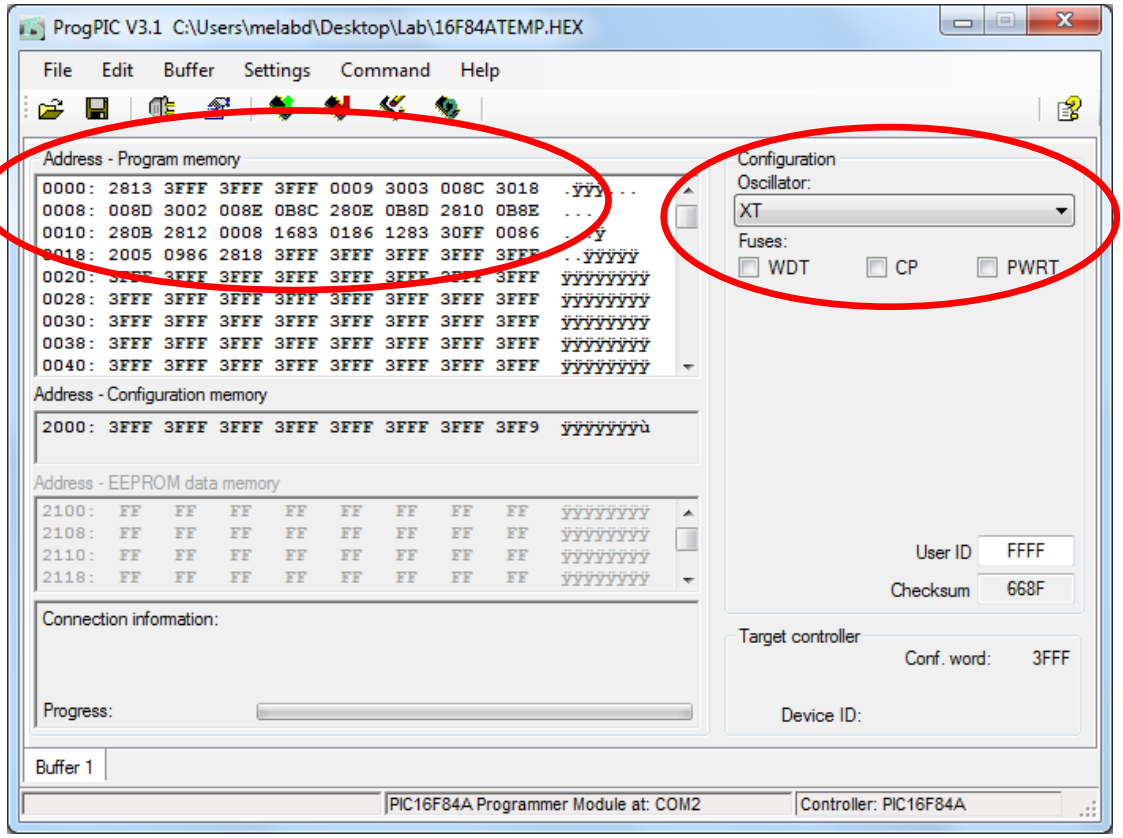

Fig. 2 Setting the configuration word.

4. **AT THIS POINT, MAKE SURE THE BOARD IS POWERED ON.** In order to test the connection to the hardware board, click on the "Test target" button  . If the connection is not detected, the error message shown below will appear.

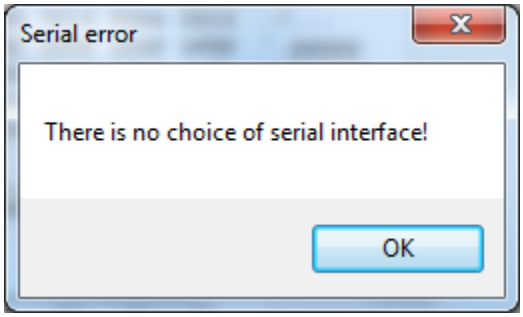

Fig. 3 Failure to detect the hardware board.

If this is the case, select “Settings→Programmer” and configure the connection as shown below.

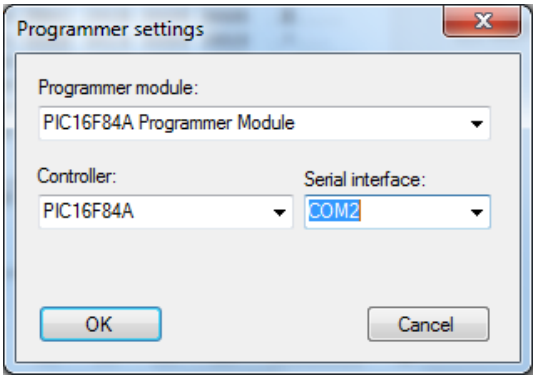

Fig. 4 Configuring the connection.

Re-run your connection test and if successful, you will read the message “Target controller available. type : PIC16F84A” in the “Connection information” section.

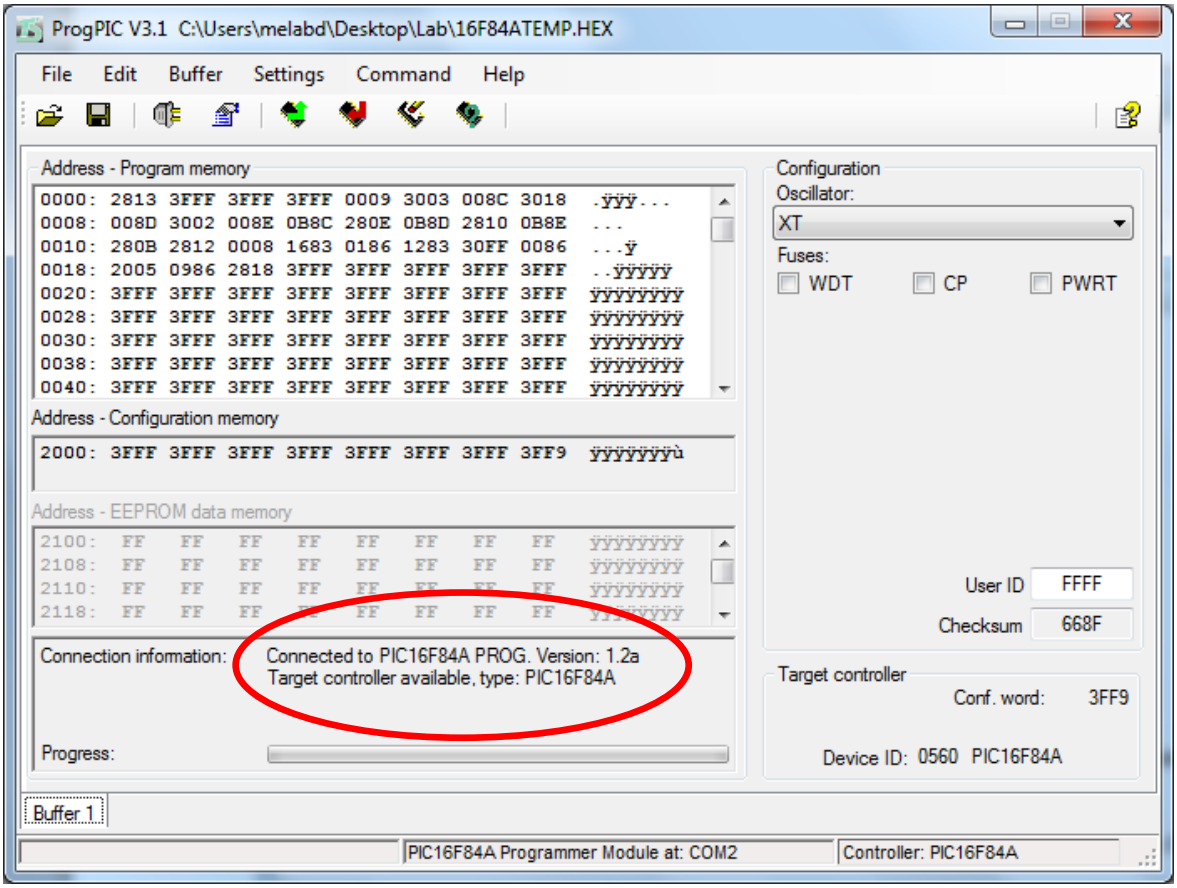

Fig. 5 Connection detected successfully.

5. To download the .HEX file to the microcontroller, click the “Write all” button  . If successful, you will read the message “Writing memory successfully!” in the “Connection information” section.

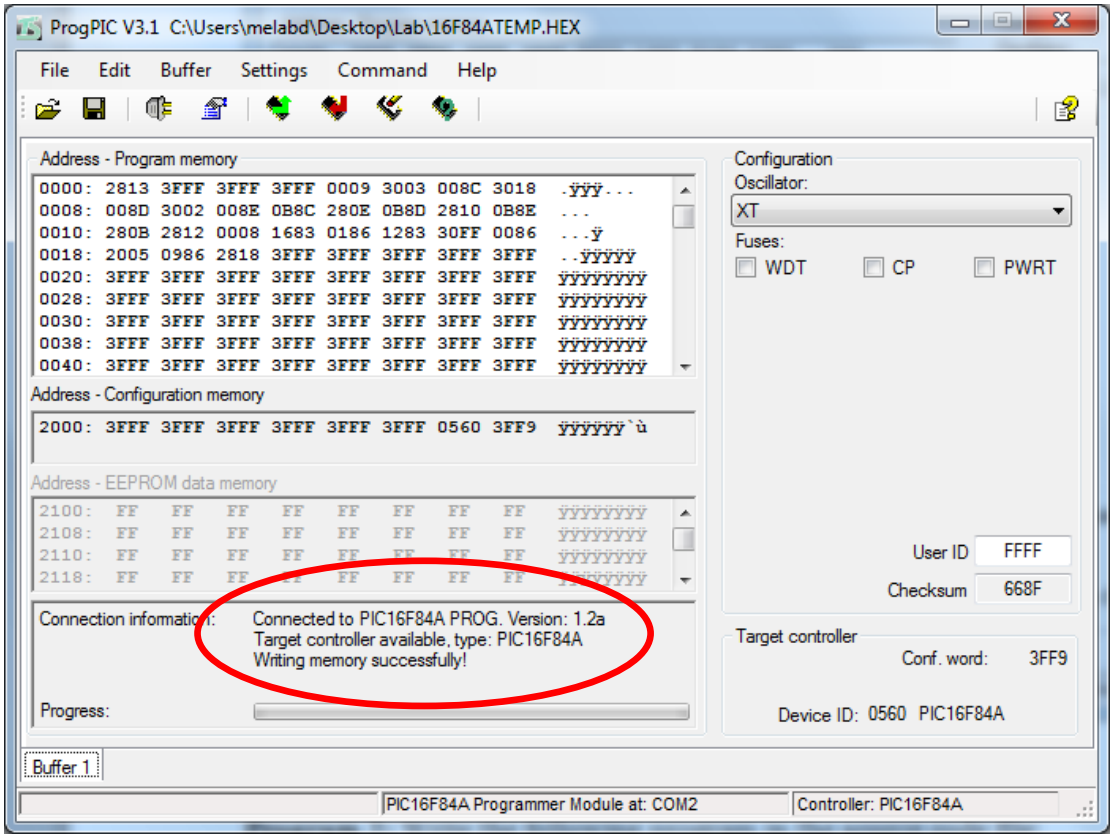

Fig. 6 .HEX file opened and downloaded successfully.

6. At this point, your program should be automatically working on the PIC16F84A. In order to sense the input at pin 0 of PORTA, please use a wire to connect “RA0” on the board to the red push button “Q0” then download and run your program. You can also connect it to the switch “S0”.

   Answer the following questions:

- What does the program do?
- What is the difference between connecting “RA0” to “Q0” or “S0”?

Exercise 1: Modify program 1 such that it increments PORTB when a push button connected to RA0 is pressed and released.

Program 2:
Create a C Project (as shown in Lab Sheet 1). Once your project is ready add a new source file and name it Lab4C.c and write the following code:

```
#include <htc.h>
#define _XTAL_FREQ 4000000
void main()
{
  TRISB=0X00;
  PORTB=0X00;
  while(1)
  {
    PORTB =~PORTB;
    __delay_ms(500);
  }

}
```

Code 2. Program 2 Code

Build the project (debug build) and check the generated assembly code.

(Go to Window → Debugging → Output → Disassembly Listing File)

- Compare the generated assembly code with Program 1
- What are the similarities?
- What are the differences?

3. Interrupts

Program 3: Write the following program in the source code file (**Note that this program requires the Delay.inc include file**):

```
        BSF     STATUS, RP0
        CLRF    TRISA
        BSF     TRISB, 0
        BSF     INTCON, GIE     ; Enable Interrupts in General
        BSF     INTCON, INTE    ; Enable External Interrupt
        BCF     STATUS, RP0
LOOP    CLRF    PORTA
        GOTO    LOOP

 ISR    BCF     INTCON, INTF    ; Clear External Interrupt Flag
        COMF    PORTA
        CALL    delay500ms
        RETFIE
```

Code 3 – Program 3

Please make sure to replace reftie with GOTO ISR at the top of your code before the main. Run your code and answer the following questions (**To run the code, make sure that RB0 is connected to INT**):

4. Report Exercises

1. Write an assembly program that
    a. Continuously reads inputs from PORTA (RA0-RA3)
    b. Outputs the summation of these numbers to PORTB
    c. The program knows there is a new input available at RA0-RA3 when a push button connected to RA4 is pressed and released.
    d. It keeps reading inputs from PORTA until the input is zero.

2. Write an assembly program that accepts two input signals on pins RA0 and RA1. According to the read inputs, the program performs a mathematical operation on variables X and Y and produces an output on PORTB according to the following table.

RA0	RA1	PORTB
0	0	X + Y
0	1	X – Y
1	0	X * Y
1	1	Quotient of X / Y

Table 3 Table for the fifth report problem.

3. Modify program 3 such that it increments the value of PORTA every time an interrupt signal is received.