INTRODUCTION TO MICROCONTROLLERS – PROGRAMMING THE PIC16F84A

Having known about the architecture of the PIC16F84A, lets dive into learning how to actually program the controller. As you all know(If you have read the post about getting started with PIC16F84A) the port B pins RB6 and RB7 along with the MCLR pin will be used to program the controller(To quote again, RB6 is pin 12, RB7 is pin 13 and MCLR is pin 4 of the microcontroller). So, whats the big deal about programming? It just this simple – these are the steps to program the controller:

1. Connect the MCLR(4th pin) pin to 13 volts
2. RB6(pin 12) is used for serial programming clock
3. RB7(pin 13) is used for serial programming data
4.

As soon as you connect the MCLR pin to 13 volts, the controller "knows" that its going to receive serial programming clock via RB6 and serial programming data via RB7 . Now, here don't worry about how to send the serial programming clock and data – there are a dozen available applications that can do this for you.

Building the Programmer:

The PIC16F84A is compatible with the serial port of our computer (You can find it in desktop computers or old laptops – a blue colored 9 pin connector). It is designed to take data from the serial port of the computer. So, if you are going to use the USB port, you need a serial to USB converter chip. Here, I will point out a good working circuit which works with the serial port. For those who don't have serial port, you can buy a PICKIT2 programmer from Microchip. Belive me – its worth investing in this programmer and you can program a wide range of PIC controllers with the programmer and it is USB compatible.

Here is a very low cost serial port programmer by JDM for the PIC16F84A (I tested it and it works fine).

You can buy the Microchip PICKit2 programmer from the microchip webiste.

Okay, now having constructed/bought your programmer, lets start programming the PIC. Initially, I would like to progress in showing how to program this controller in assembly language. We all do know that C is indeed very easy to learn and needs very less codes to write the program. But the reason behind choosing assembly is that, while using assembly, you will understand the controller more and get better grip over the hardware of the controller (Roughly saying, programming in

assembly is like driving your own car. Programming in C is like hiring a driver for driving it – hope you got the idea).

Writing and assembling the code:

Microchip MPLAB downloadable from microchip website is a good IDE(and freeware) which integrates a assembler. But, I would strongly recommend PIC simulator IDE from oshonsoft (Its not a freeware but you get a 30 time launchable trial – which is enough for getting started.)

After you open the PIC simulator IDE from oshonsoft, set the microcontroller to PIC16F84A, and clock frequency to 10MHz. Now, from tools menu, open the assembler. Here is the code – you will learn it as you progress.

```
BSF STATUS,05H
MOVLW B'00000000'
MOVWF TRISB
BCF STATUS,05H
LOOP MOVLW B'10000000'
MOVWF PORTB
GOTO LOOP
```

If you dont know what are TRISB, STATUS please read the post about Getting started with PIC16F84A.

Here goes the explanation:

What the above program does is that, it delivers a high output on RB7 and can be checked with a multimeter or by connecting a LED. To achieve the aforementioned result, these are the steps

1. First, we have to set the port B pins as output and to achieve this, TRISB register must be used and to access TRISB, you have to navigate to bank1 and you know what must be done to navigate to bank1 – set the 5th bit of the STATUS register. For this, we use the instruction BSF – Bit Set F [BSF STATUS,05H]
2. Now, we are in bank 1 here, say, you wanna set the portB pins as output. To achieve this, first move the value 00H or B'00000000' (in binary format for easy understanding) to W register (Remember? W register is the only register to which we can directly write to/ read from). For this we use the instruction MOVLW – Move literal value to W[MOVLW B'00000000']
3. Then move the contents of W register to TRISB. For this we use the instruction MOVWF – move the contents of W register to F [MOVWF TRISB]

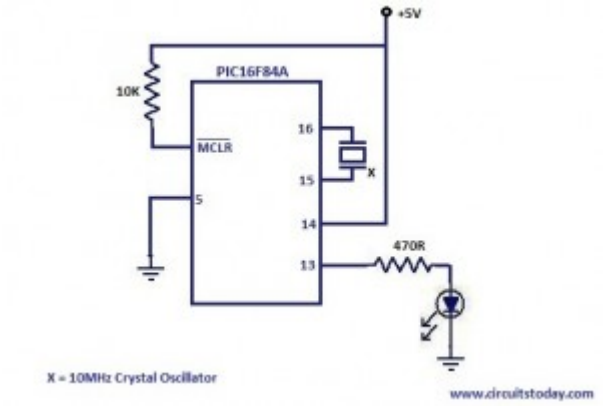
4. Having finished the operation of setting the portB pins to output, we have to toggle to bank 0 to access port B and for this we need to clear the 5th bit of STATUS register. For this we use the instruction BCF – Bit Clear F [BCF STATUS,05H]
5. Now, we are in bank 0 here, move the value 80H or B’10000000′ to W register [MOVLW B'10000000']
6. Then move the contents of W register to PORTB [MOVWF PORTB]
7. Look, what the controller does is that it executes these instructions once and then keeps quiet. But thats very fast. Indeed you cannot see the high output on RB7. So, to see what we have done, we have to make the microcontroller keep RB7 high for longer time or indefinitely. So, we are putting it in a loop. You can pretty much understand the loop thing.
   NOTE: Here, while typing the program in the simulator’s assembler, leave a blank space before each instruction except for loop label. Because, It considers all the words that appears without a space as labels.
   After writing the program, in the assembler window, goto tools and select assemble and load option. This option will assemble the program and load it into the simulator. Then select the microcontroller view from tools menu in the simulator main window. When you select simulation – start, you must see the result in the microcontroller view. This is the software verification part.

“Burning” the PIC:

Having built the programmer, now, load the controller into the programmer and then connect it to the serial port of your PC or to your USB port if it is a USB programmer. Download and install ICPROG or Pony prog software (which is a programming software for the PIC). Now, when you select the correct COM port, your software must be able to read the device (The PIC16F84A in your case). Now, the file you need to write into the controller is the .hex file. You can find this file in the directory in which you saved the .asm file when assembling and loading the PIC simulator. Now, open and browse to the .hex file and then select the write option and when it displays “programming successful”, Voila! your PIC is programmed! (Programming the microcontroller is generally called as “burning” the program)

Checking the Program:

Below is a very simple setup in order to check your programming. Just pop out the PIC from the programmer and wire it in a solderless breadboard as show below.



If the LED glows, then Bingo! you’re done!

Source : http://www.circuitstoday.com/introduction-to-microcontrollers-programming-the-pic16f84a