MODULE 09

MICROCONTROLLERS I

PREREQUISITES: MODULE 07: DIGITAL CIRCUITS.

OUTLINE OF MODULE 09:

What you will learn about in this Module:

(Based upon the Microchip PIC16C/Fxx families):

- Basic Architecture of Microcontrollers
- Programming fundamentals
  - Introduction to PIC-C (an inexpensive and popular C compiler)
  - Introduction to MP-Lab
- Loading compiled code into a microcontroller
- Hardware considerations: how to implement a microcontroller

What you will build in the lab:

You will build a very simple circuit that will allow you to test the function of a microcontroller IC. Mostly, you will learn the basics of programming a microcontroller, and the essential components that you need to include to get it to function (power and an external oscillator).

INTRODUCTION:

Back in prehistoric times people used to control everything with clever mechanisms, such as an hourglass to time an event (as sand runs out of the mechanism, a lever loses weight and eventually moves to a new position), mechanical clock mechanisms, rotating cams to control the timing of complex devices, etc. The automatic transmission in a car is essentially an hydraulic computer, and some cars still use tuned mechanisms to set shift points based on RPM and throttle position. But the advent of the Apollo space program changed everything: it became clear that the world would come under the control of tiny, highly integrated microcontrollers. What is the difference between a *microcontroller* and a *microprocessor*? Well, they can have different architectures, but in fact they are quite similar conceptually. Microprocessors are built with the intention of carrying out computation, whereas a microcontroller is built to interact with the outside world to monitor and control things. Thus, microcontrollers tend to be small, cheap, and extremely simple compared to modern microprocessors. It has been estimated that the average American interacts with several dozen microcontrollers every day before they even get to the office. This is because microcontrollers are pretty much everywhere: coffee makers, automobiles (several in each modern car), garage door openers, traffic lights, radios, security gates…they are even in the little chip implanted by aliens into your brain to control your every thought and action. At this point it is probable that more humans have interacted with microcontrollers than have seen snow or have become literate. You simply can not be a good engineer without knowing something about microcontrollers. That being said, lash yourself to the mast for a stormy ride, because after the next three modules on microcontrollers you will never see the world in the same way. It will be much, much better.

READINGS FROM HOROWITZ AND HILL (H&H): *ART OF ELECTRONICS*

Chapter 10: Introduction (note the quaint terminology in some places)
10.01-10.02
10.17 (assembly language, compilers…we will use a C compiler)
10.23 (number formats)

ADDITIONAL READINGS & INTERNET RESEARCH:

Visit the Microchip web site and look around a bit. What sorts of engineering and technical support tools do they provide?:

Now, download the datasheet for the PIC16F84A microcontroller, since we will use that device for the next several modules. Read only pages 1-5, since the datasheet is about 88 pages in length.

MichiganEngineering

MODULE 09

SELF QUIZ

1: How many digital I/O (input/output) pins does the PIC16F84A microcontroller have available for data I/O?

2: What is the operating voltage range for the PIC16F84A microcontroller?

3: What is the maximum "clock speed" for the PIC16F84A microcontroller?

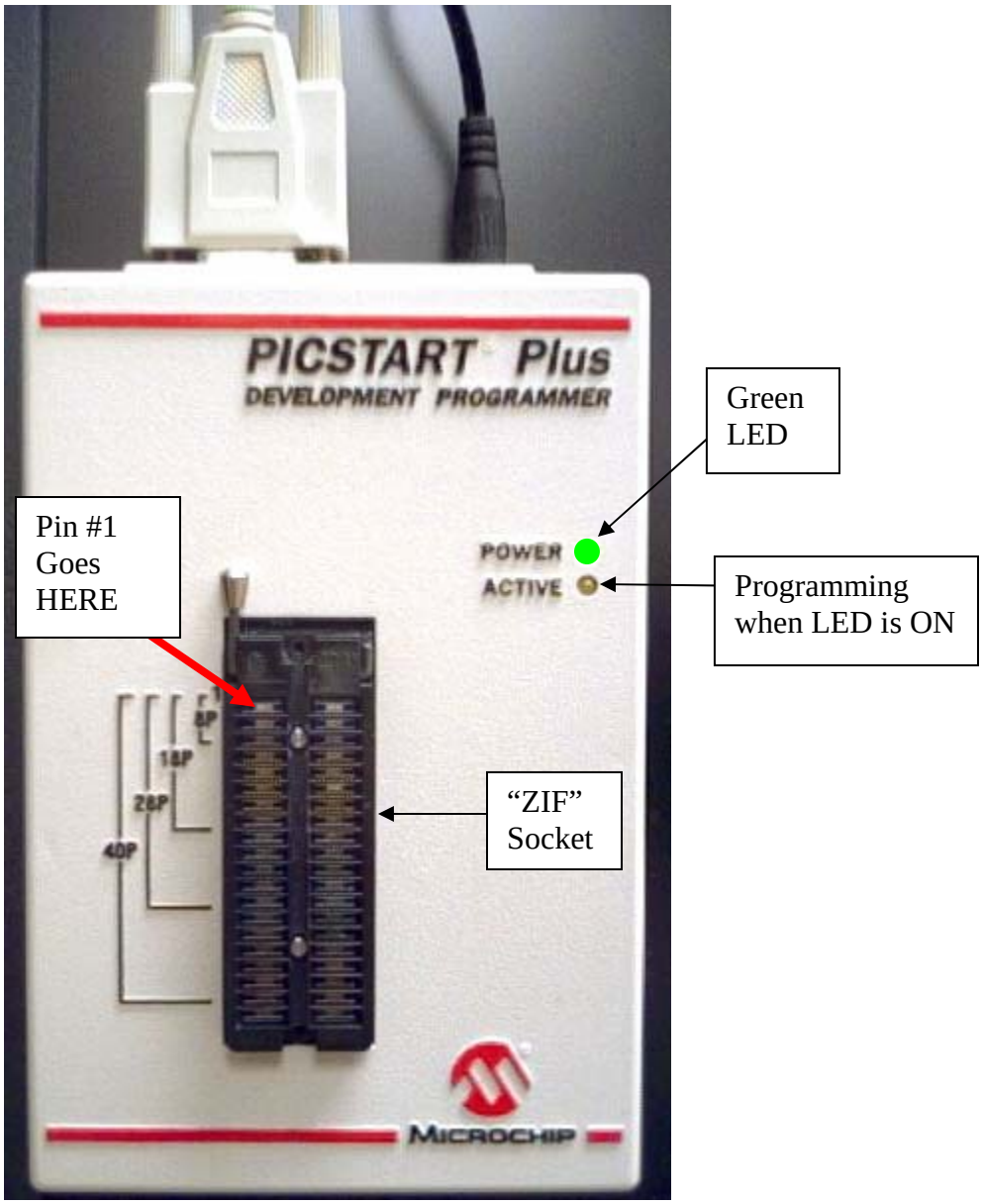
4: What variables might you alter in your circuit to get the microcontroller to use the MINIMUM amount of electrical power (there are several ways to do this)? Be explicit, and use the tables and graphs on the datasheet for the microcontroller to support your answers.

PLEASE ANSWER THE ABOVE QUESTIONS AND E-MAIL TO THE INSTRUCTOR
"I have neither given nor received aid on this examination, nor have I concealed any violation of the Honor Code"

X_____

LABORATORY PROJECTS

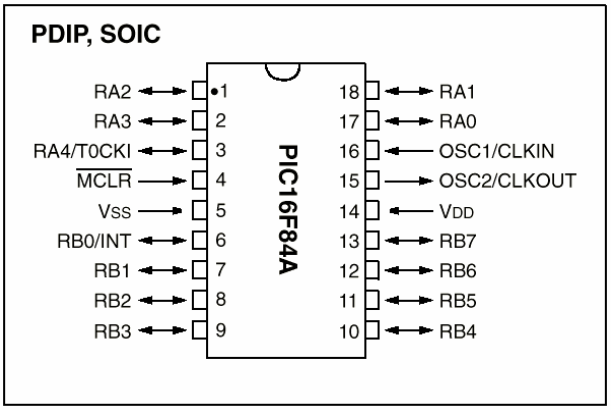
1- You will be using the PicStart plus microcontroller programmer shown below:



Basic external anatomy of the PIC16F84A microcontroller:

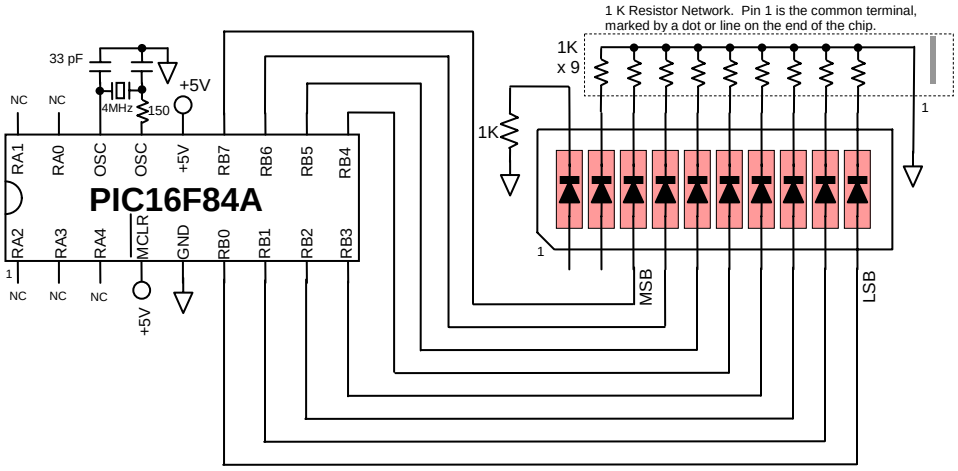
Note that each pin is numbered on this diagram: 1-18
Pins 15 & 16 are the Oscillator. They set the clock speed (0-20 MHz)
Each internal instruction executes in 4 clock cycles
Pin 14 is +5V positive power
Pin 5 is Ground
Pin 4 is the Master Clear (resets the microcontroller if it goes to GROUND)
Pins 17, 18, 1, 2, & 3 are Register A (RAx): They are just 5 bits of digital I/O
Pins 6-13 are Register B (RBx): This is a full 8-bit digital I/O (input/output) port

<u>What can you do with this (or any other) microcontroller?</u>
INPUT: You can **read** the digital state of any of the register pins (RAx or RBx).
OUTPUT: You can **write** a digital value to any or all of the register pins (RAx or RBx).
Logical Operations: conditional branching, time delays, simple mathematical operations, etc., based on the input states or on internal numerical values.



Build the circuit shown in the schematic below:

Use IC Sockets for both the PIC16F84A and the 10-segmet LED Bar Display so you can remove and replace each of these components easily.



10-segment red LED Bar Display
Black dot or chamfered corner on side denotes pin 1. This side has all of the diode anodes. NOTE: you should use a 20-pin IC Socket so that you can flip the Bar Display around if you put it in backwards!

You can base your circuit layout on the example circuit that I have provided in the parts cabinet.

Note the resistor array. This is just a simple and compact way to add a row of resistors to a circuit. It is a narrow strip with one pin indicated by a bar or dot. The indicated pin is the “common” pin, which you would generally connect to ground or +5V. In this case, we connect this pin to ground so that we can establish a ground point for the 1K resistors that will limit the current flow through each of the diodes in the 10-segment LED bar display. Resistor networks can be purchased with almost any number of resistors, but we just happen to have 9-element (9 resistor) arrays, so we need to add one extra single 1K resistor to the LED bar display, since there are 10 LEDs in the display.

Programming the Microcontroller:

Once you have built this circuit, you are ready to program your microcontroller. The code is located on the computer in the lab (D:/ME499 – Bob/module09.c). You can also download the source code from the ME499 web page.

To program the microcontroller you will need to use two types of software. The first is called a "C" compiler, located on the desktop of the computer in the lab. It is available from Custom Computer Services, Inc. (http://www.ccsinfo.com/picc-referall.shtml). You will use this compiler to write all of your source code in "C". The compiler will then generate the "*.hex" code that the programmer can understand. The second software is called MPLAB, also on the desktop of the lab computer. This software is free (available from www.microchip.com) and is used for transferring the *.hex code that you wrote earlier onto the microcontroller chip. To do this, you place the microcontroller chip into the PicStart Plus Microcontroller Programmer and "burn" the code onto the chip. A detailed description of this process is given below. For this module the source code has already been written, all you will need to do is:

1- Locate the source code on the computer in the lab (D:/ME499 – Bob/module09.c),
2- Transfer the file "module09.c" to your own folder (create one for your self) on the D drive,
3- Open the code using the PIC C Compiler software, compile the code (it will generate a file called "module09.hex".
4- Plug in the PicStart Plus Microcontroller Programmer.
5- Start the MPLAB software
6- Select the correct device type in the dialog box (PIC16F84A)
7- Import the hex code: module09.hex
8- Press "PROGRAM"
9- Wait for the programming to complete
10- Press "VERIFY" to verify that the code was burned in
11- Remove the microcontroller from the programmer
12- Plug the microcontroller into your circuit.
13- Turn the power on for your circuit.
14- Observe the LED Bar Graph, and carefully watch what happens.

For more detailed directions, ask an instructor or read the following pages.

It is very important that you read the source code (module09.c) and be sure that you understand the program flow. The code is very simple and has plenty of comments to tell you how it functions. You will modify this code in future modules. You can look up the commands that are available in the "C" compiler using the CCSC Help File, which is available within the PIC C Compiler software. By burning the code to the microcontroller, plugging it into your circuit, and verifying the correct function, you have done the most difficult aspects of employing embedded microcontrollers. The rest is really just a matter of learning to program and adding useful hardware to your circuit. You will do this in subsequent modules.

Detailed Instructions for burning a program onto a microcontroller:

Programming in "C":

Note: The icon for the PIC C Compiler is on the desktop. The program can also be accessed through the Windows Start menu → Programs → PCW → PIC C Compiler.

Procedure:

1. Make a folder for yourself on the D drive on the computer in the lab.
2. Locate the file "module09.c" on the D drive, then place it in your new folder.
3. Start the compiler by double clicking on the PIC C Compiler icon on the desktop or by accessing the program through the Windows Start menu.
4. Once the program is open, verify that the small window on the menu bar reads "Microchip 14 bit". If it does not, use the pull down menu to set it to "Microchip 14 bit". This tells the computer which compiler to use (obviously, the 14-bit compiler).
5. Go to File → Open
6. Locate the file "module09.c" in your new folder, and open this file. Alternatively, you can just download the file for "module09.c" from the web page and copy and paste the text into the text editor window of the C Compiler software.
7. The "C" source code is just an ASCII text file: you need to compile it so that the microcontroller can understand it. Located on the menu bar is a "Compile" pull-down menu. Access that menu and compile your program (There is only one selection available in the pull-down menu and that is → Compile)
8. While your program is compiling you should see a compile window appear on the screen. This window tells you the progress of the compiling function.
9. If your program compiles successfully, there will be a "No Errors" statement at the bottom of the Compile Window. Click "Ok", minimize PIC C Compiler and proceed to the MPLAB instructions listed below.
10. If you program does not compile successfully you have an error in your code. Don't panic, the compiler will usually put the cursor on the line where your error is located. Sometimes however, the error may actually be in the line just above or just below the line where the cursor is positioned.
11. Correct your error, save your file and re-compile.

Downloading your program onto your microcontroller:

Ok, now that you have compiled your C source code you are in a position to download your code onto your microcontroller. Look in your new folder. In addition to the original file (module09.c) there will be about 6 new files that were created by the C Compiler. These are very useful for different purposes, but the file you will burn onto the microcontroller will be called "module90.hex". This is a file that contains machine language that your microcontroller can understand. It is in hexadecimal format.

MODULE 09

Now you will use the software MPLAB and the PicStart Programmer to burn the *.hex code onto your microcontroller.

Note: The icon for MPLAB is on the desktop (upper right hand corner). The program can also be accessed through the Windows Start menu → Programs→ Microchip MPLAB → MPLAB.

1. Plug the PICSTART PLUS Development Programmer power cord into the nearest wall socket (This is the black power cable that has a wall transformer on one end and the other end is plugged into the programmer). Once the power cord is plugged in, the green LED on the programmer should illuminate.
2. Verify that the serial cable is plugged into both the PicStart programmer and the desktop computer.
3. Start MPLAB by double clicking the MPLAB icon on the desktop or access the program through the Windows Start menu.
4. Along the top of the MPLAB window are several pull down menus. The following selections will guide you through enabling the programmer, importing a hex file, and downloading the hex code onto your microcontroller.
5. PICSTART Plus → Enable Programmer (Three additional windows should appear on the screen. They are: "Program Memory Window", "Configuration Bits", and the "PICSTART Plus Device Programmer")
6. I have set the default device in the Device Programmer window to be the PIC16F84A, which is the microcontroller that your motor-control circuit is equipped with. If the device window does not say PIC16F84A, use the pull-down menu to locate and select this device.
7. PICSTART Plus → Erase program Memory (This assures that no other code is in the memory of MPLAB before you Import your HEX code). Disregard what the dialog box says when you do this for the first time.
8. When you compiled your code, the compiler actually produced about 6 different files. The only files you are interested in, are your "C" source code and the file with the .HEX file extension.
9. File → Import → Import to Memory (a small "explorer-like window labeled "Import Emulation Window" will appear)
10. Using the "Directories" side of this window, navigate to the folder where your HEX file is stored (same file you put the "module09.c" source code into).
11. Select your HEX file and choose "OK".
12. Your code should now appear in the Program Memory Window. It will be in a very confusing hexadecimal format, but this is good.
13. Verify that the configuration bits are set to the following values:
    - Oscillator → HS
    - Watchdog → ON
    - Power up to → OFF
    - Code Protect → OFF

14. The PicStart Programmer has a special socket on it to hold many different sizes of integrated circuits. This is called a ZIF socket (it is labeled on the picture of the PicStart programmer, above). "ZIF" stands for Zero Insertion Force. The way it works is this: You push the little metal lever on the ZIF socket down, and you will notice that every little slot opens up to accept a pin from an IC chip. Pull the lever pack up (vertical) and each slot closes again. Go ahead and try it.
15. Now, push the lever down to open up the ZIF socket so you can put your microcontroller in.
16. Notice that right next to the ZIF lever is a small numeral "1". This is where pin #1 of the microcontroller must go. This shows you how to orient your microcontroller in the ZIF socket.
17. Place your microcontroller chip in the Programmer. Remember that the end of the chip with the small hemispherical depression on it goes to the top of programmer socket (Pin #1).
18. Pull the programmer socket lever to the vertical position to lock your chip into the ZIF socket.
19. Return to the MPLAB program, you are now ready to program the chip
20. In the PICSTART PLUS Device Programmer window select "Program"
21. A small window will appear that shows the progress of the programming function.
22. If the programming function completes successfully, the progress window will disappear. Additionally, in the PICSTART PLUS Device Programmer window under "Program Statistics" there will be a "PASS" indication.
23. Push the ZIF lever back down, remove your chip from the programmer and insert into the IC socket on your board. Remember to make sure that end of the chip with the small hemispherical depression is aligned with the end of the socket with the small hemispherical notch in it. Make sure that you have all of the chip's pins aligned with the socket cavities before you apply to much downward pressure to seat the chip in the socket.
24. If you have problems downloading your code onto the chip, check that your chip is locked into the programmer and repeat these steps beginning with step 17. If you continue to have problems please contact an instructor by email (fulcrum@umich.edu, yoda@umich.edu) or in our lab (Room 2178 G.G. Brown)

If you want to change the way the microprocessor functions, you just change the source "C" code and repeat the above steps. This is all much easier than it seems to you at the moment. For example, you can change the speed or direction of the counting in your source code. In the next module we will focus on programming.

Note: When re-programming your chip you do not have to erase the chip. Simply import your new HEX file into the program memory and hit program. The chip's memory will be overwritten with the new code.

FEEDBACK

Was this Module useful and informative?

_____

Is there a topic that should get more or better coverage?

_____

In what way can this Module be improved:

Content: _____

Depth of Coverage: _____

Style: _____

Any additional comments that will help us to improve this course:

_____

_____

_____

If you prefer, you may e-mail comments directly to Bob Dennis: yoda@umich.edu