BME 580 Microcontrollers I

Lab 2: Hello World

Ross Rucho

Reading assignment:

Please read the following information before beginning the laboratory work.

https://en.wikipedia.org/wiki/Microcontroller

https://en.wikipedia.org/wiki/Embedded_system

The three most frustrating and difficult aspects of embedded system design (in my opinion) are:

(1) soldering the microcontroller properly into place on a printed circuit board

(2) basic wiring for power on the PCB without short circuiting anything or applying too much voltage

(3) programming or "burning" your code onto the microcontroller

The purpose of this laboratory exercise is to focus on these three simple but potentially very frustrating problems with the simplest possible laboratory exercise: make an LED flash under the control of a microcontroller.

Follow these instructions step-by-step

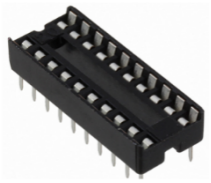
STEP 1: Circuit Diagram (schematic drawing)

Review the circuit information below, then draw a schematic in EagleCAD. Note that you will probably have to create several components for yourself using the FILE → New → Library function in EagleCAD. Since you have already done this in the EagleCAD tutorials, you should have no problem making new components as you need them. You should begin to build up your own library of components. HINT: I also suggest that you keep a separate backup of your EagleCAD component library files. Sometimes they get corrupted or lost. In general I suggest making your own components for EagleCAD, although it is possible to search around on the Internet to find component libraries. My suggestion is to build up a good set of generic components such as resistors, capacitors, diodes, then add the PCB layout for new variations of these generic components as you need them. Then, if you want to specify a specific brand or type or value of that component, call it out in a text note on your schematic near the component. This is better than building an entirely new component library for each available generic component. That would take forever, it would make your library file set huge, and would add almost no value.

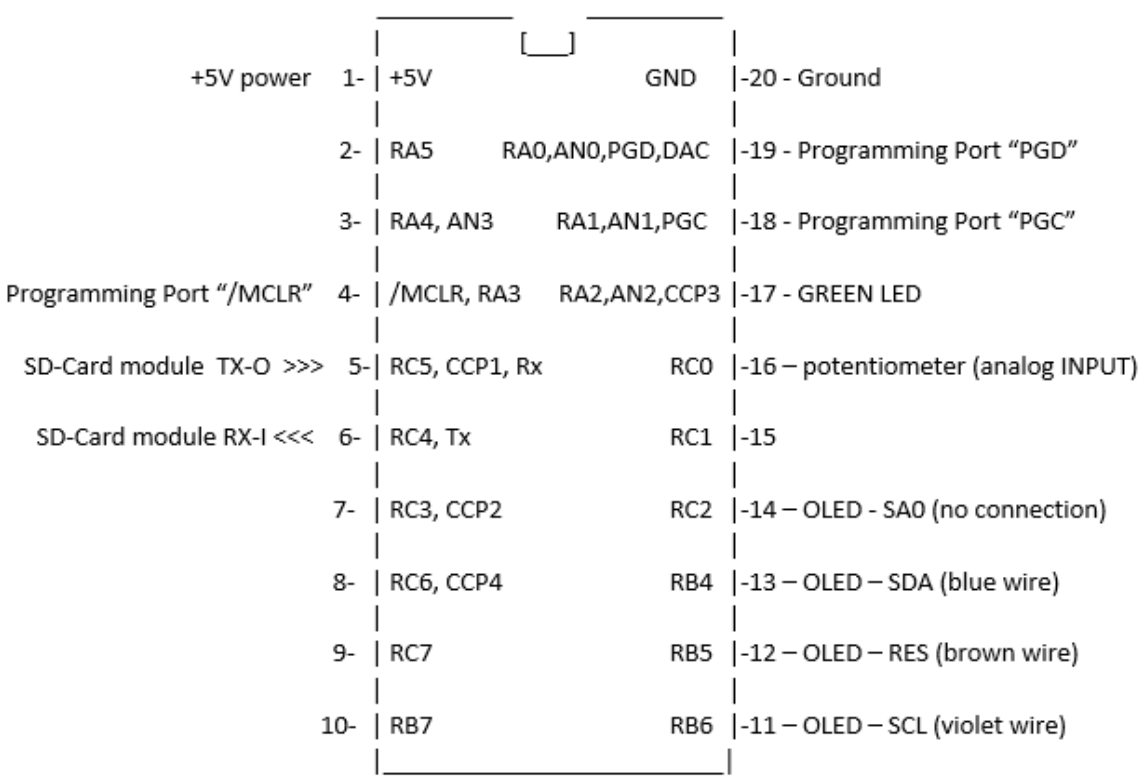
BME 580 Microcontrollers I

Circuit Components:

Prototyping PCB IC socket PIC16F1829 microcontroller programming header LED


PIC16F1829 8SOIC Pinout:



CONNECTIONS – from microcontroller …

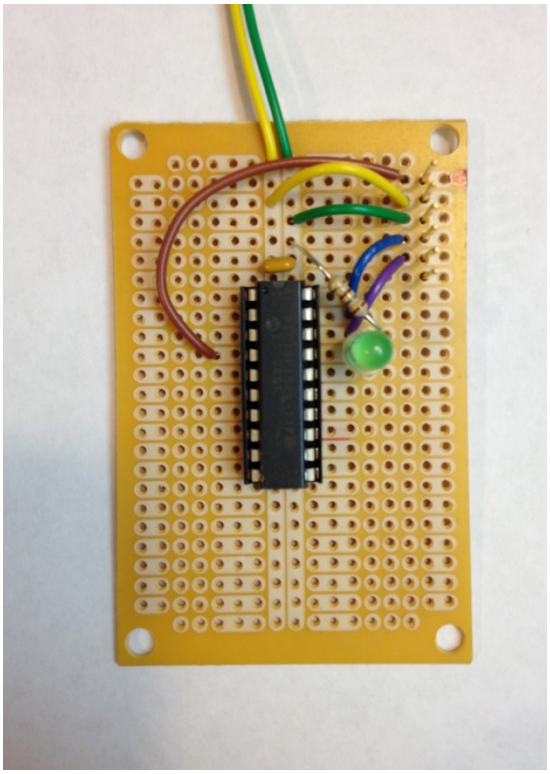
Pin 1 to +5V and to programming header pin #2
Pin 4 microcontroller "/MCLR", connect to programming header pin #1
Pin 17 to green LED, then through a resistor (about 100 Ohm) to GROUND
Pin 18 to Programming Port "PGC", same as programming header pin #5
Pin 19 to Programming Port "PGD", same as programming header pin #4
PIN 20 to GROUND to programming header pin #3

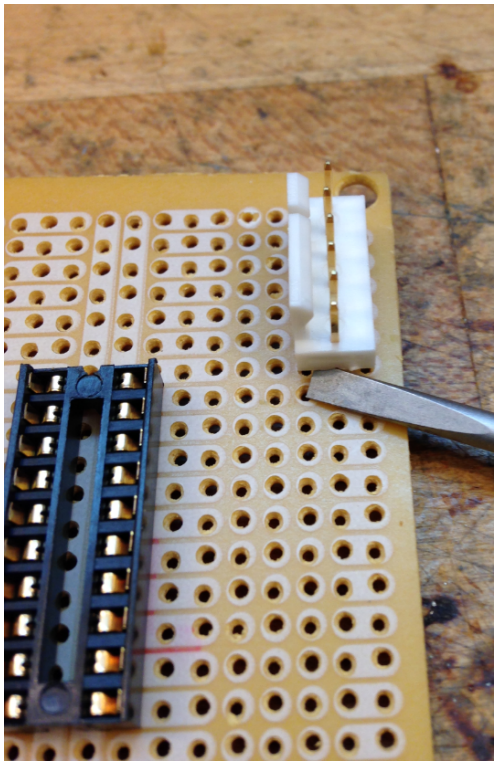
For the time being, do not connect the SD-card and the OLED display (pins 5, 6, 11, 12, 13, and 14)
Reserve the other pins for future use (pins 2, 3, 7, 8, 9, 10, 15, 16, for potentiometers, switches, etc.)

BME 580 Microcontrollers I

LAYOUT for your circuit board:

I suggest that you lay out your printed circuit board approximately as shown below. *You do not have to lay out your Hello World board this way, but if you follow this layout then you will have room to fit in all of the future components for future labs. Basically, you will continue to build on this board throughout this course.*





Note in the upper right of the first picture that, after soldering the programming port (6-pin gold header) into place, I have used a flat-blade screw driver to remove the white plastic support for the header. This allows the PICKit3 programmer to plug in fully. Doing this is optional, it just makes the programmer fit a bit better but you do not have to do it. MAKE SURE TO SOLDER ALL OF YOUR WIRES TO THE HEADER BEFORE REMOVING THE WHITE PLASTIC SUPPORT, or the gold pins will move around when you try to solder wires to them on the bottom of the PCB. The resistor to the LED can be anywhere in the range of about 100 to 510 Ohms. Can you tell the resistance from the color bars?

Also note: I have used the following wire color code:

yellow = +5V green = ground brown = reset blue = digital data violet = digital clock

OPTIONAL decoupling capacitor:
You will also note that I added a small yellow component just above the microcontroller in the photograph to the left. This is a 1 uF capacitor. You can find these in the lab supply drawers It connects between the +5V and ground. It tends to smooth out the power supply to the microcontroller so that you do not get glitches if the power supply transiently fluctuates. This is one of the simplest things you can add to an embedded microcontroller circuit to make it more reliable, especially in electrically noisy environments

BME 580 Microcontrollers I

PICKit 3 Programmer:

When you wire up the programming port you will need to know the pinout for the PICKit 3 programmer, shown at right. The pinout is shown below

Make sure there is a PICKit3 programmer attached to a USB port on your workstation in the lab.

Look in Appendix 2 of this document to see the sample source code, which includes an ASCII sketch of the PIC16F1829 chip with each pin numbered and labeled. You need to wire up the programming port so that it connects the microcontroller to the PICKit 3 programmer correctly.

Here is how the pins connect:

PIC16F8129 pin	PICKit 3 pin
/MCLR →	Pin 1 (indicated by white triangle on PICKit 3)
+5V →	Pin 2
Ground →	Pin 3
PGD →	Pin 4
PGC →	Pin 5
No connection to	Pin 6

Make these connections by soldering the programming header to the PCB, then connect each pin on the programming header to the microcontroller using individual wires.

MAKING the EagleCAD schematic for this lab:

I suggest that each of you build your own EagleCAD component for the microcontroller (PIC16F1829). Although there are some publicly available components out there for EagleCAD, they are often inaccurate and can cause a lot more wasted time and effort than just taking 10 minutes to build your own component correctly.

As you build your PIC16F1829 component, you will be building the PCB layout for the standard "dual in-line" (DIP) chip package, which is the standard configuration for integrated circuits with little legs that stick down through holes in the printed circuit board.

To make the PCB footprint layout correctly for this component, you will discover that the dimensions for the holes on the PCB are not given in the last pages of the 400+ page datasheet for the PIC16F1829... they show every other footprint, but why not the standard footprint?!?! The reason is that the standard footprint is so standard that they just assume that everyone knows it. Here is the "standard footprint":

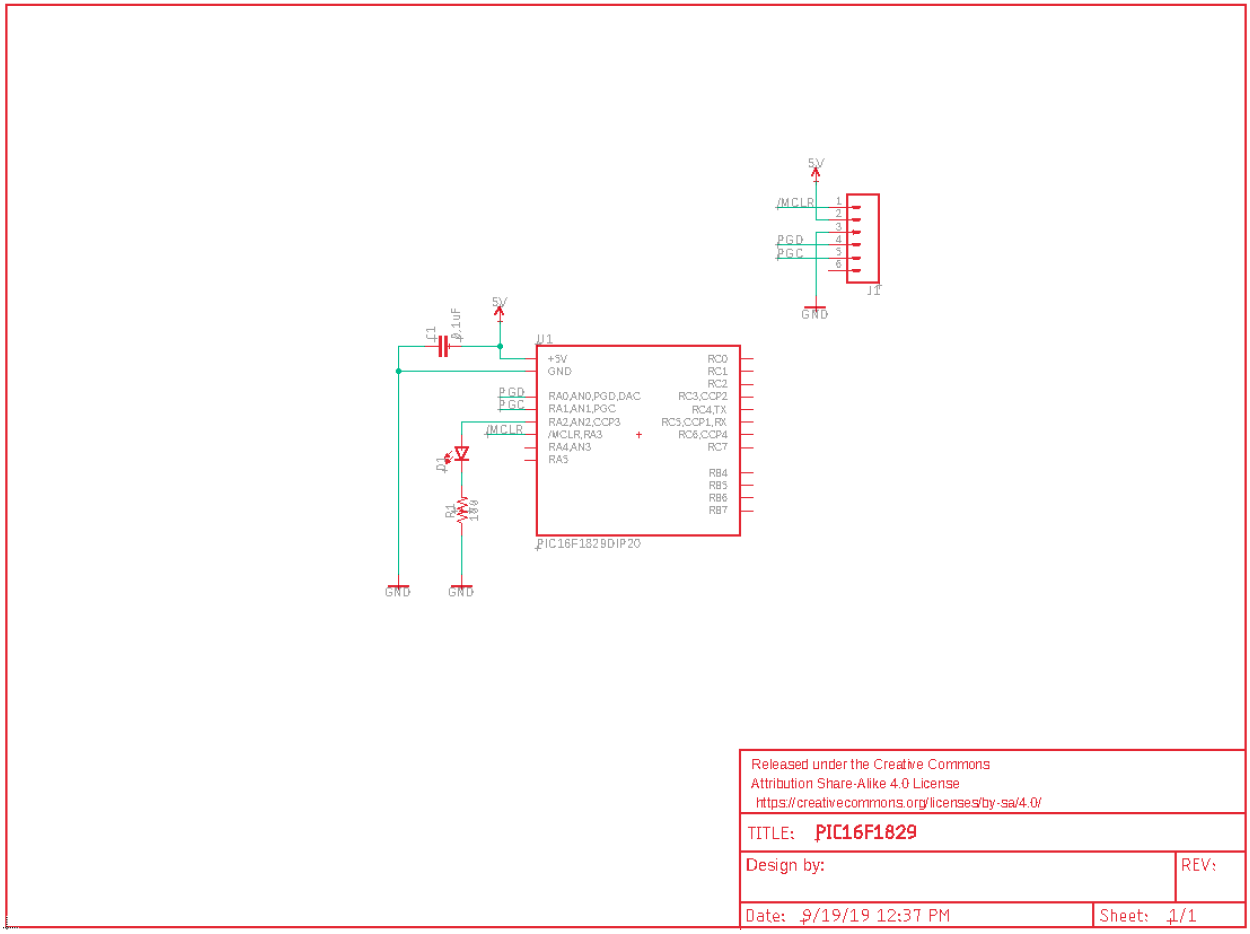
For standard ICs with legs that stick through the PCB, the holes all fit on a 0.100" grid (1/10" grid). This is why electronic prototyping breadboards all have holes on a 0.100” grid pattern.

BME 580 Microcontrollers I

So, since your PIC16F1829 IC has 20 legs, in 2 rows of 10 legs (pins), all these legs must fit onto a 1/10" grid. For "standard width" IC chips, the two rows are 0.300" apart, and each leg in each row is 0.100" from the adjacent leg. Some ICs are wider, 0.600" width is common for large memory and microcontroller ICs, but the PIC16F1829 is standard width, so the two rows of holes should be placed 0.300" apart. And you need to know the correct diameter for each hole. I suggest using about 0.031" diameter, or slightly larger.

FINALLY, I suggest that you use a SQUARE PAD to designate pin #1 of the component, and make all other pins with round pads. That way you will know which way to plug in the IC when you go to solder it onto the PCB.

Your EagleCAD version of the circuit schematic:



BME 580 Microcontrollers I

STEP 2: Get the components

Begin collecting the components. All of the main components you need are available from the instructor as a pre-sorted kit. When you need anything else such as wire or solder, you can find these in the laboratory. Also, standard components such as resistors and capacitors are available in the bins on the wall in the lab.

STEP 3: Build the circuit

Before you start, consider reading a short review on the basics of soldering electronic components to a printed circuit board or a prototyping breadboard. Here is an excellent one:

https://learn.sparkfun.com/tutorials/how-to-solder-through-hole-soldering

Find a lab bench where you can work. It must have a working soldering iron. Use an IC socket on the PCB. Do not solder the microcontroller directly onto the PCB so that you can swap out the microcontroller easily if you burn it out. It is usually a good practice to use different color wires for different signals on your prototype. Try this:

Green wire for GROUND
Yellow wire for +5V
Brown wire for /MCLR
Blue wire for PGD
Violet wire for PGC
White wire to the green LED

If you use a standard wire color code for all of your circuits, at least for power and ground, then in the future it will be very easy for you to pick up an old circuit and remember it quickly.

STEP 4: Check the solder joints

Ask the instructor to look at your circuit before powering it up or programming the microcontroller

BME 580 Microcontrollers I

STEP 5: Compile the source code into "hex" code that you can program onto the microcontroller

This will take some time to figure out, so please be patient. The instructor will help you with this.

1- Open the PICC Compiler application on a workstation in the lab
2- Select the 14-bit compiler on the top toolbar
3- Cut-and-paste the source code from Appendix 2 (below) into the PICC compiler
4- Save the new project as something like *yourname_hello_world_1*
5- PICC will save your source code into a file with the extension *.c
6- Press the "COMPILE" button on the top toolbar
7- Wait a few seconds
8- Look for a window that tells you if there were any errors during compiling
9- If there were errors, see if you can fix them. The sample code should compile without errors
10- If there were no errors, then the compiler generated a *.hex file from your source code
11- The *.hex file will have the same name as your source code file.
12- Locate the *.hex code. It should be in the same file as the source code.

STEP 6: Program ("burn") the sample source code onto the microcontroller

If you already are familiar with programming PIC microcontrollers then go ahead and use whatever method you are comfortable with. If you have never done this before it will take some time to figure out, so please be patient. Below you will see a step-by-step description of how to program a microcontroller on a PCB. The instructor will help you with this.

1- Open the MPLab IDE application (usually version 8.80, it may be different in the lab)
2- At this point do not connect your PCB to external power. We will use USB power.
3- Make sure the PICKit 3 programmer is attached by USB to the workstation
4- Plug the PICKit3 into the programming header on your PCB
5- In the top toolbar: CONFIGURE -> SELECT DEVICE…, scroll down the list to find: PIC16F1829
6- Click OK
7- If it shows an error, for now you can ignore the error
8- In the top toolbar: PROGRAMMER -> Select Programmer -> PICKit 3
9- The OUTPUT window will show that the programmer is connecting to the microcontroller
10- If there is a CAUTION about 3.3V devices, click OK to clear the window
11- The OUTPUT window will probably tell you "You must connect a target device to use PICkit"
12- In the top toolbar: PROGRAMMER -> Settings… -> click the tab that says "POWER"
13- Click the box that says "Power target circuit from PICkit 3"
14- Verify that the voltage is set to 5.000 Volts
15- Click APPLY
16- If there is a CAUTION about 3.3V devices, click OK and clear the CAUTION window
17- Click OK again if necessary to clear the Settings… POWER tab
18- The Output window may show that the microcontroller is properly connected
    The window will display a line message such as "Device ID revision = ……."
19- If there is still an error message, check the physical connection to the programmer-header on your PCB, then on the top toolbar: PROGRAMMER -> Reconnect

BME 580 Microcontrollers I

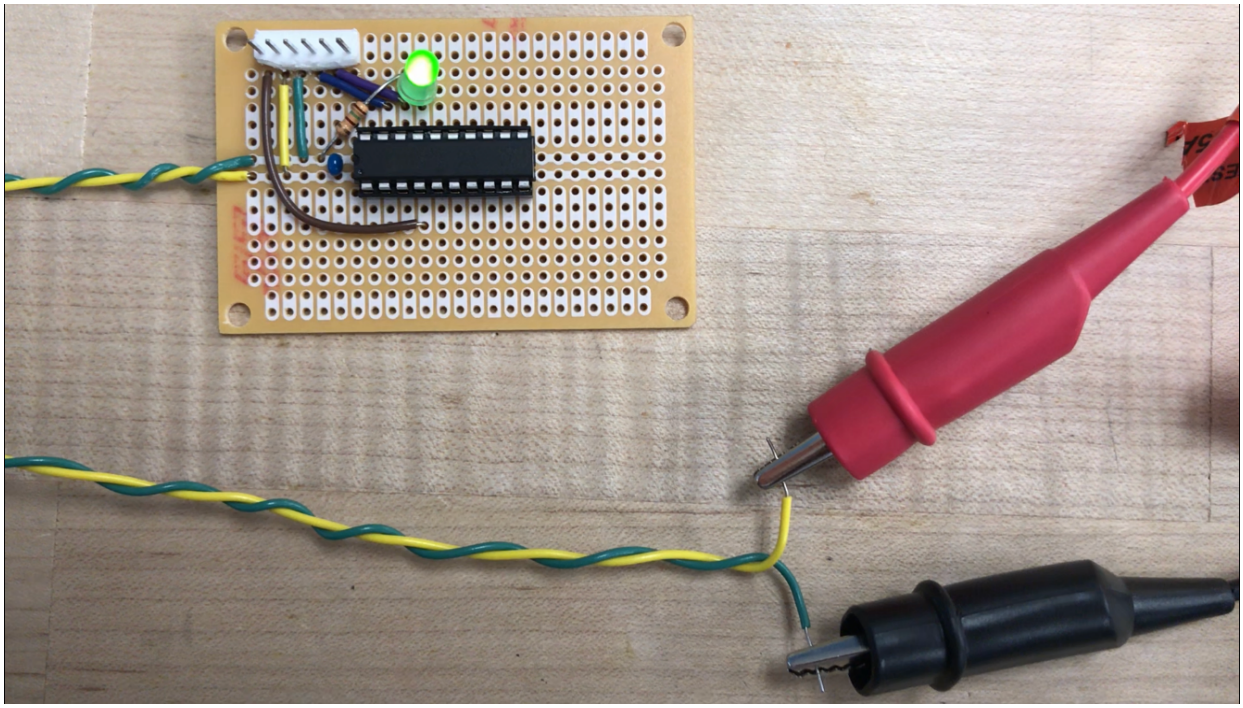
20- You may see a CAUTION window again. If so, clear it by clicking OK
21- Once you see the message "Device ID revision = ……..", you are good to go!
22- In the top toolbar: VIEW -> Program Memory.
23- The new Program Memory window will be filled with blank opcode lines: 3FFF
24- Select FILE -> IMPORT and browse to find the *.hex file that you just generated
25- Click on the *.hex code you want to load onto the microcontroller and click OK
26- You will see the *.hex code in the Program Memory window. Each line is now different. This is the compiled machine code you will burn onto the microcontroller
27- The Output window will change to display the line "Loaded:…"
28- Mouse over the icons in the top tool bar to find the ones that say:
"program" "read" "verify" "erase" and "blank check"
Each button allows you to interact with the code on the microcontroller to program it, read the code from the microcontroller, verify that the code in the program window is the same as the code on the microcontroller and that it was correctly programmed, to check that the microcontroller is blank, or to erase the microcontroller memory and reset it to default.
29- Left click on the little icon that says "program"
30- There will be a delay during which the Output window will display "Programming…"
31- After a few seconds, if the microcontroller is properly programmed, the window will say

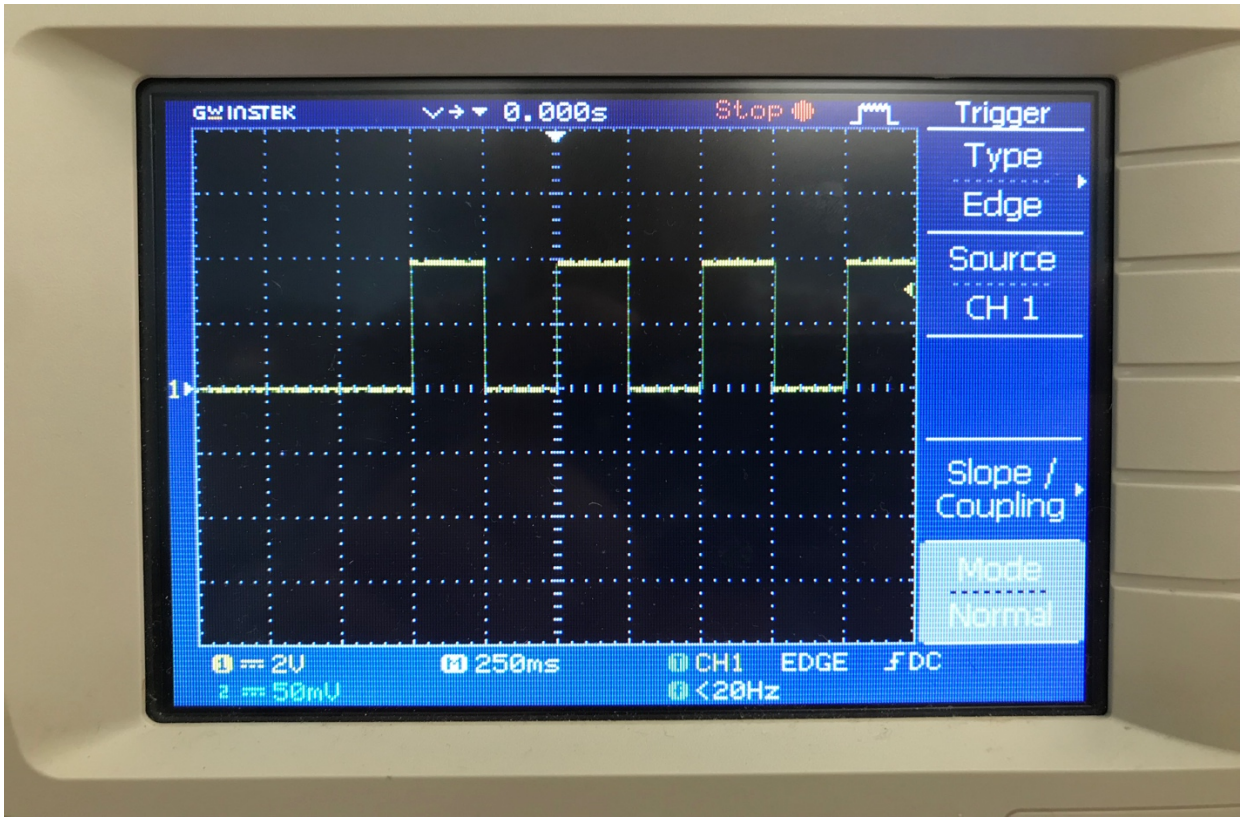
"Programing/Verify complete"

32- This means that the programmer has burned your code onto the microcontroller, then verified that the code was correctly programmed.
33- If all went well, your chip is programmed and the green LED should start blinking
34- Right now your circuit is being powered by the PICKit3 programmer
35- Disconnect the PICKit3 programmer from the programming header
36- Set the bench top power supply to deliver +5.0 volts
37- Power your circuit from the benchtop power supply and verify that it works.
38- Measure the voltage signal on the LED using an oscilloscope.
39- Take a photo of the oscilloscope trace and paste the image where indicated below.
40- Time the flashing of your LED. Does it flash exactly twice per second?
Hint: you can use an oscilloscope to test the voltage on the green LED

BME 580 Microcontrollers I

Photograph of your completed circuit:





BME 580 Microcontrollers I

APPENDIX 1: Resources

Microcontroller Programming Example #1: Hello World

To program your microcontroller you should have the following materials handy. This is a lot of stuff, so please plan ahead or you will probably not be able to program and debug your embedded system.

SOFTWARE--

S1: You need the C compiler. We use the PIC C Compiler in this class

S2: MPLAB IDE, this is the freeware from Microchip that programs the microcontroller
This software is also available on the lab computers, and is accessible as a Desktop Shortcut
You can try to program the microcontrollers directly from within the compiler (PIC C)
but I suggest instead you use MPLAB as a stand-alone device programmer driver. I further suggest that you use version 8.80 of MPLab unless you are familiar with a newer version

S3: Source code from programs you plan to use as an example. You can cut-and-paste from the source code.

S4: EagleCAD and your schematic and layout files, so that you can design and update your hardware design.

HARDWARE--

H1: The PICKit3 programmer and USB cable. Drivers for these are installed on the lab workstations.

H2: Your embedded circuit. You need it handy so you can program it.
Also, glitches may arise from differences between your schematic and the actual hardware you built.

H3: Programming cable (PICKit3 to your device), unless you included a PICKit3 header on your PCB

H4: Soldering iron and rework supplies to build and correct your hardware circuit.
During programming you will often discover mistakes in your hardware wiring, especially at first

H5: An eye loupe or magnifier, especially if you are working with small-pitch surface mount devices

DOCUMENTATION-- (put these on your laptop, generally in the form of *.PDF files unless otherwise noted)

D1: Datasheet for the microcontroller you are programming (example: PIC16F1829)

D2: PIC-C User Manual, which is located on each lab computer that has the C compiler installed

D3: Datasheets for each of the components you have built into your embedded system
You should find and keep copies of datasheets for each of the components you use

D4: The *.h file for the microcontroller you plan to program: 16F1829.h
These files tell you the commands and pin references for each device, see below
Save a copy of this file for yourself but save it as a *.TXT file (do not corrupt the original)

D5: Example source code. This is the very best way to learn how to program microcontrollers
It is usually best to start every project with at least one template example source code
Hard-copy is OK, but source code in the form of a text file allows you to cut-and-paste

D6: Application Notes, if you have found any that relate to your project

BME 580 Microcontrollers I

Getting Started Step-by-Step (sort of)

1- Familiarize yourself with the overall contents of the PIC-C User Manual, which is located on each computer in the lab that has the C compiler installed, in a location similar to:

C:\Program Files (x86)\PICC\Data Sheets

The User Manual is a file named: CCS_C_Manual.pdf

2- Start with a bit of example code, such as this document (See Appendix 2 for the sample source code), so you can get the right header files in place to set up the microcontroller properly. This is also sometimes called "template code" or "boiler plate"

3- I usually use a bit of crude ASCII art to sketch out the device and the intended use for each pin on the microcontroller, see the sample source code in Appendix 2.

4- Review your reference copy of the*.h file (header file) for the microcontroller you plan to use.

The file will include the name of the microcontroller, for example: 16F1829.h

PLEASE REMEMBER: locate the file *16F1829.h* and save it as a *.txt file for future reference

DO NOT ALTER THE ORIGINAL *.hex file, and DO NOT MOVE THE *.h FILE TO A DIFFERENT LOCATION

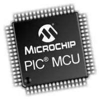
*.h files are located on the hard drive in a location that looks like:

C:\Program Files (x86)\PICC\Devices

Open the file using a text editor and then re-save it as “16F1829-h.txt” in a convenient location (usually near your source code). This is for your reference, and the saved *.txt file helps prevent you from corrupting the original *.h file which is actually linked and compiled with your source code.

You should read through the 16F1829-h.txt reference file before you begin programming, just so you can see what's there. You will probably refer to this reference file many times to see what options are available for configuring your microcontroller, and to see the correct syntax for each configuration. You will learn a lot about your microcontroller and its advanced capabilities as you grow familiar with the contents of this header file.

5- Starting with your template source code, you can start writing down a draft of your new source code. Even if you do not have access to the C compiler (for example, if you are away from the lab) you can still work by using any text editor to draft your code. Then just copy and paste the code one section at a time into the PICC compiler/editor, try to compile it to see if there are any errors, then continue to copy/paste your draft code until it has all been transferred into the compiler/editor.



BME 580 Microcontrollers I

APPENDIX 2: Source Code for this lab

Cut and paste the source code (below) into the PICC Compiler:

```
/*  This is template code that I generated that will accomplish the first objective of your first "Hello World"
project.   You should be able to map this into your first draft of your source code, but please be sure to check
your device (microcontroller) type and pin-out to make sure it is correct.
//        uC-test-1.c      RGD 7/14/2012//
//
//  First sample template program for the PIC16F1829 microcontroller for BMME580 class
//
//
//  Use PCM 14 bit compiler
//  Circuit configuration:
//  Internal Oscillator with I/O on PGM and PGD, osc speed = 4.0 MHz, 1 us instruction cycle

                              PIC16F1829  8SOIC Pinout:
                            ___________      ___________
                            |           [___]          |
            +5V power   1-  | +5V                  GND |-20 - Ground
                            |                          |
                        2-  | RA5      RA0,AN0,PGD,DAC |-19  - Programming Port "PGD"
                            |                          |
                        3-  | RA4, AN3     RA1,AN1,PGC |-18  - Programming Port "PGC"
                            |                          |
Programming Port "/MCLR" 4- | /MCLR, RA3  RA2,AN2,CCP3 |-17  - GREEN LED
                            |                          |
  SD-Card module  Rxi  >>> 5- | RC5, CCP1, Rx      RC0 |-16 – potentiometer (analog INPUT)
                            |                          |
  SD-Card module Txo <<<  6- | RC4, Tx            RC1 |-15
                            |                          |
                        7-  | RC3, CCP2          RC2 |-14 – OLED - SA0
                            |                          |
                        8-  | RC6, CCP4          RB4 |-13 – OLED - SDA
                            |                          |
                        9-  | RC7                RB5 |-12 – OLED - RES
                            |                          |
                        10- | RB7                RB6 |-11 – OLED - SCL
                            |__________________________|
*/
//  SAMPLE SOURCE CODE from Bob Dennis
```

BME 580 Microcontrollers I

```
//  SIMPLE Hello World demonstration, with ability to flash an LED if the microcontroller is programmed properly

#include <16F1829.h>          //  you have to tell the compiler which microcontroller device you are programming
#include <MATH.H>             //  look in the PIC C User Manual to see what this library contains.  Do you need this?
#use delay (clock=4000000) //  4 MHz, 1 us instruction cycle, CHECK TIMING DELAYS!
#fuses INTRC_IO, NOWDT, NOMCLR, PUT
                              //  Set for internal oscillator operation with RA6 and RA7 as digital I/O,
                              //  Disable the WDT.  Later setup WDT = 2048 ms WDT,  ref 16F1829.h to find syntax
                              //  Disable the /MCLR pin so that it can be used as general I/O
                              //  Enable the Power-Up Timer, allows power supply to stabilize at start-up

#define green_LED 98          //  RA2, a.k.a pin #17 on the PIC16F1829 device is in memory location 98
                              //  the 16F1829.h file is where you find all of the pin locations in memory
                              //  also note that RC0 (pin #17) has another function:  CCP3.  You will use this later.

////////////////////////////////////
// SUBROUTINES,  Here is where you set up your I/O ports and other features of the microcontroller
//  This is where it is very handy to have a *.txt copy of the *.h file for the microcontroller:    16F1829.h

void init_ports(void) {
  SETUP_OSCILLATOR(OSC_4MHZ);
                        //  Look for this syntax in the *-h.txt file under  "Constants used in setup_oscillator() are:"
                        //   also note that for this microcontroller the internal oscillator can be changed
                        //   on the fly as needed during program execution.
}

////////////////////////////////////
// PIC16F1829 goes here at RESET

void main() {
  init_ports();           //  Initialize ports

CYCLE:
     OUTPUT_BIT(green_LED, 1);     //  start by turning the green LED ON
               delay_ms(250);          //  delay for 250 ms, basically the microcontroller sits there doing nothing
     OUTPUT_BIT(green_LED, 0);     //  then turn the green LED OFF after 1/4 second delay
               delay_ms(250);          //  delay another 250 ms
goto CYCLE;                          //  execute this loop without stopping

} // main
```