Name _____

Lab Section _____

PIC – GPIO to Blink a LED

Lab 4

Introduction: The equivalent of the traditional "Hello World" program on an embedded system is often a blinking LED. In this lab you will work on a blinking LED application using a time wasting loop. The time wasting loop is not an efficient method for measuring long segments of time; however it does serve as a good introduction to microcontroller programming.

Lab Requirements:

1. Modify the supplied source code so the LED on your PIC Dev board blinks at the rate of the last digit of your RedID in Hz. If your RedID ends in 0 do 10Hz.

2. Connect a momentary push button switch to one of the GPIO pins. Detect when the button is pressed and double the rate of the blinking LED.

3. Demonstrate the proper frequencies (±0.1Hz) of oscillation using the oscilloscope.

4. Submission of your neatly formatted source code.

Last Digit of Red ID_____ Demo Check (JK)_____

Getting Started: MPLAB X is an Integrated Development Environment "IDE" for developing firmware to run on Microchip microcontrollers. The IDE contains the tools needed to write and build code as-well-as program and debug a microcontroller. A demonstration on how to use the tools will be given in the lab. The basic steps in setting up a new project are shown below:

File -> New Project…	
Microchip Embedded -> Standalone Project	***Next >***
Device: PIC16F18324	***Next >***
Select Tool: PICkit3 or Simulator	***Next >***
Select Compiler: XC8 (ver.)	***Next >***
Select Project Name and Folder: Name and Location	***Finish >***

Device Configuration: Before we can begin to write code for the PIC microcontroller we need to consider how we want the device to be configured at power on. The Device configuration sets many of the important operating characteristics of the part such as what to use for a clock source and how nMCLR is driven. The *#pragma config* directive is used to indicate to the compiler that the following definitions refer to configuration settings:

```
#pragma config RSTOSC = HFINT1   // Power-up default value for COSC bits->HFINTOSC
#pragma config MCLRE = ON        // MCLR/VPP pin function is MCLR; Weak pull-up enabled
```

We will discuss the various settings for configuring the device in class and look at the automated tools for generating the *#pragma config* directives. For today's lab you will use the configuration settings in the supplied source code.

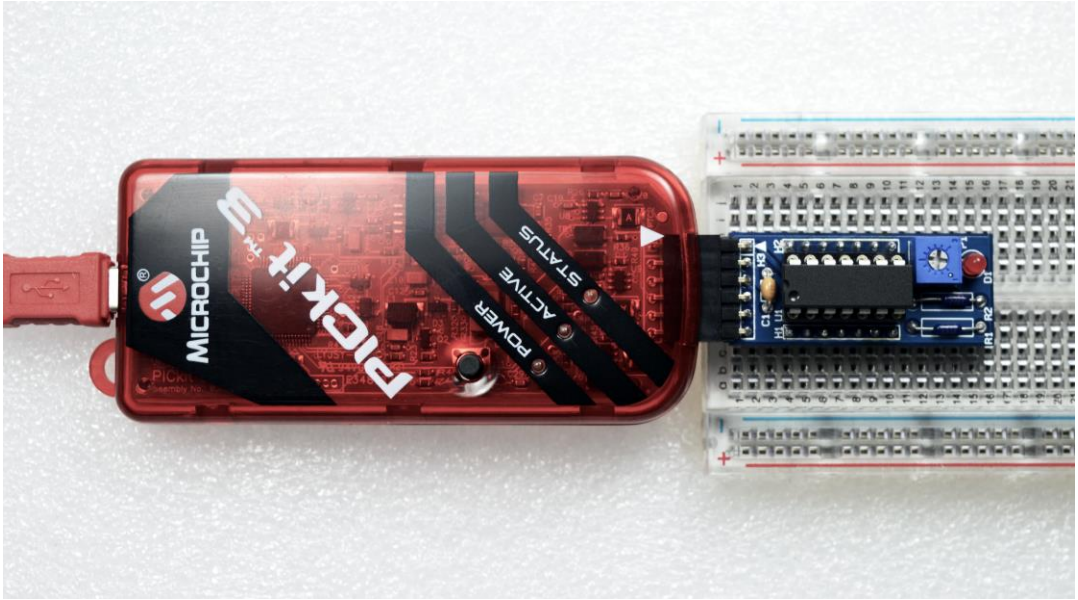
Header Files: It is important that you remember to include the header file with the register definitions for the microcontroller you are using. An easy way to do this is to include xc.h which will point at the header file for the device being used (pic16f18324.h).

```
#include <xc.h>
```

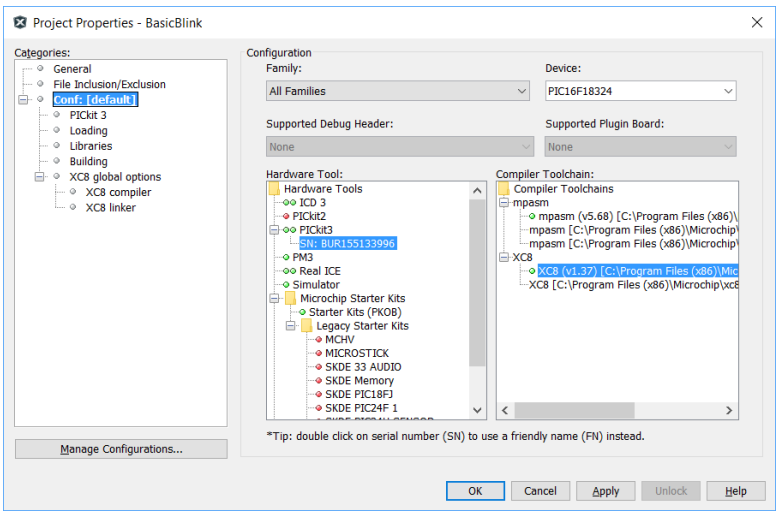
Blinking the LED*:* The sample code at SeniorDesignLab.com will blink the LED using a time wasting loop at an unspecified rate. Your assignment is to modify the code to blink the LED at a rate that matches the last digit of your RedID. To do this you will need to make minor changes to the code supplied and test for proper operation.

Connecting the Programmer: Connect the PICkit 3 programmer to the 6-pin programming header with the arrow on the programmer facing pin 1 of H3 (see photo below). You do not need to connect the *PIC Dev 14* to a DC power supply when using the PICkit 3. The PICkit 3 can provide up to 30mA to power the target board.

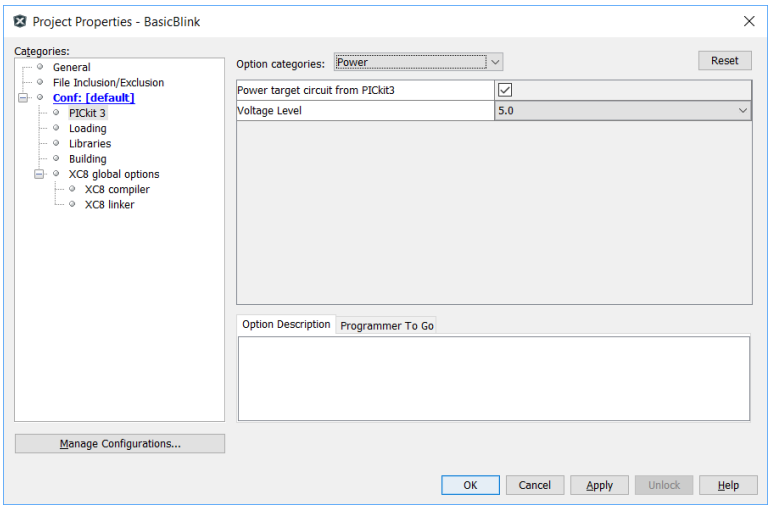
PICkit 3 Programmer:



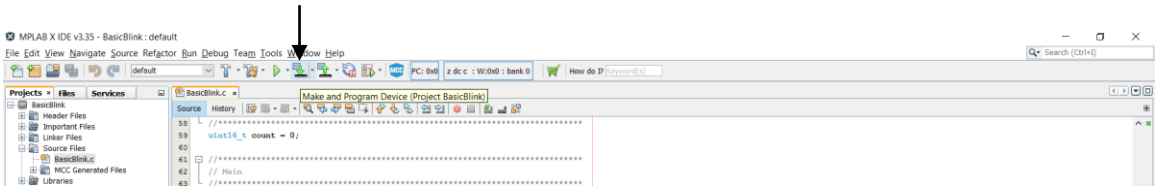
To power your board from the PICkit 3 programmer you will need to set the output voltage and enable the output power. Select **Project Properties** from the **File** drop down menu



Set up the PICkit 3 programmer by selecting:
PICkit 3 under the **Options categories:** Select **Power**

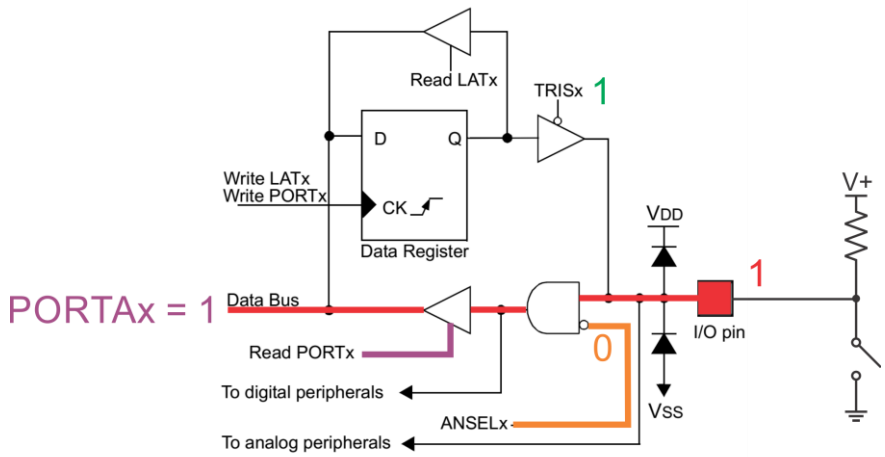


Set the operating voltage to 5.0 and check the box to power target from PICkit 3. The output window will now display the Device Revision of the connected device when you press the **Make and Program Device** button.



Programmer to target power is enabled - VDD = 5.000000 volts.
Target device PIC16F18324 found.
Device ID Revision = 2003

Connecting the Push Button: Connect a momentary switch to one of the GPIO pins by using a 10k pullup tied to Vdd as shown below:



Read from the PORT Write to the LAT

To read the state of the switch you can check the value of the individual Port bit where you connected the switch. Make sure the corresponding ANSEL and TRIS bits are set to the correct values.

PIC Dev 14 Schematic: