CSUS EEE 174 Lab - Section 4 - Tuesday

Laboratory Experiment Number 3: Lab Report

**MicroChip PICkit3**



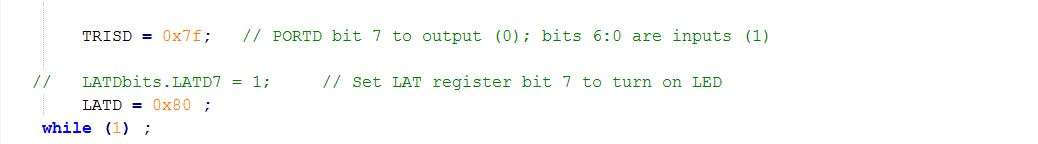
**Introduction**

The PICkit 3 programmer/debugger is a simple circuit debugger controlled by the MPLAB IDE (v8.20 or greater) software. The PICkit 3 programmer/debugger is an integral part of the development engineer’s tool suite. The PICkit3 is used along with the Microchip PIC microcontroller (MCU) and dsPIC Digital Signal Controller (DSCs) that are based on In-Circuit Serial Programming (ICSP). Besides that, the PICkit 3 is also used to be a simple IDE for software developer with powerful compiler.

1. **Introduction to the PICkit3 PIC18 Microcontroller**
   1. **Lesson 1: Hello LED**

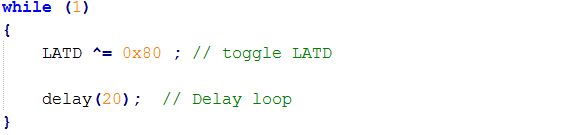
The objective of lesson 1 is to create a MPLAB C compiler project in the MPLAB IDE to turn on the demo board LED. Following the setup instruction, a new project is created before proceeding in the program.

By choosing the Lesson1\ Hello LED project found from given folder, we are ready for building the project. After clicking build all and receiving successful confirmation, the LED turns on as desired. Source code for lesson 1 is provided below:

****

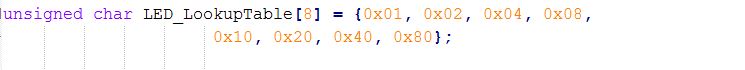
* 1. **Lesson 2: Blink LED**

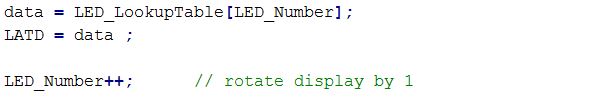
The objective of second lesson is to understand the Configuration bits of the PIC18F4K20 and the setup procedure in the MPLAB C source file. By following similar instruction, yet now for lesson 2, the LED was able to be turned on and blinking as desired delay time. Source code for lesson 2 is provided below:

****

* 1. **Lesson 3: Rotate LED**

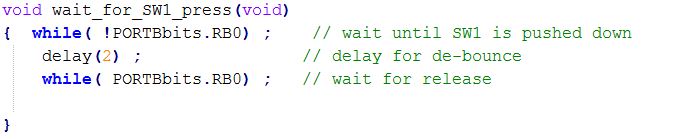
The objective of lesson 3 is to create a code that is able to rotate the LED display in some particular patterns. Rotation in LED basically means shifting the lights from 0 to 8 or vice versa. Once this one is up, the next one should be on. Following the similar instruction for lesson 3, the LEDs from 0 to 7 were able to be on in such order without having more than one lighted up simultaneously. Source code for lesson 3 is provided below:

****

****

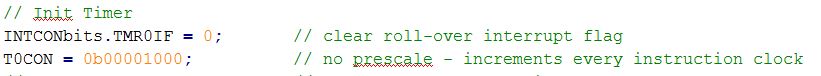
* 1. **Lesson 4: Switch Input**

The objective of lesson 4 is to combine the rotation function from lesson 3 with the integrated switch from the board. By implementing the switch along with rotation function, the user will be able to switch the rotation order, from left to right (0 to 7) or vice versa (7 to 0). By following the instruction in lesson 4, the LEDs were able to be lighted up and rotated as desired. Source code for lesson 4 is provided below:

****

* 1. **Lesson 5: Time Debug**

The objective of lesson 5 is to use Timer0 in time delays while rotating the demo board LED’s instead of using the program loop delays function. The demo board switch reverses the direction of the rotation. By following the instruction in lesson 5, we were able to implement the performance as desired. Source code for lesson 5 is provided below:

****

1. **PICkit 3 DEBUG**
   1. **Lesson 6: Debug EXPRESS**

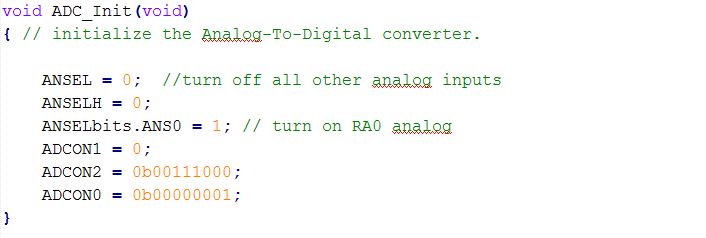
The main objective of this lesson 6 is to familiarize ourselves with the In-Circuit-Debugger (ICD) which uses the same MPLAB IDE workspace and project as lesson 5. The PICkit 3 Debug Express kit works in conjunction with the MPLAB IDE application to run, stop, and single-step through program. HALT functions, which halts the program and examines the next statement to executed, was examined at first in this step.

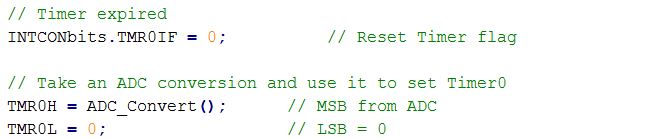
The next command STEP, also known as stepping allows the code to be executed one statement at a time or the user to debug and observe the behavior of the code line by line. However, the step command can be divided into three options, which are step into, step over, and step out. The first step function, step into, will step through the statements one block of code at a time until a function is called. On the other hand, step over function will only step to the next statement after the function call but will not step into the function. The last one, step function, step out allows for a complete execution of the current functions and steps into the next statement after the function call.

Besides discovering halt and step commands, another method for debugging called breakpoints was also examine. When breakpoints are used during debugging and the program is instructed to run, it will not halt the target until it reaches the breakpoint statement.

1. **ADC, Interrupts, and PWM**
   1. **Lesson 7: ADC**

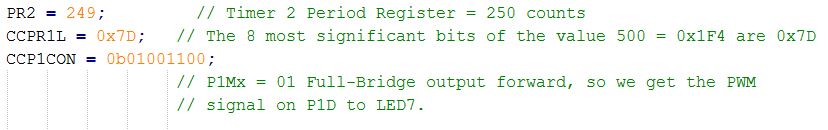
The objective of this lesson is to apply all the fundamental concepts learned from previous lessons to build an Analog-to-Digital Converter by a potentiometer. This lesson blinks and rotates the lights with an input switch to reverse the rotation direction and a potentiometer to change the speed of rotation. To accomplish the control of speed with the potentiometer a timer clock is used. The clock controls the speed of the rotation and the potentiometer controls the clocks pre-scale timer. So as the potentiometer is varied, the analog value of the potentiometer is input into the pre-scale timer of the clock. This changes the speed of the LED rotation. To initialize the Analog Digital Converter, the line *ADC\_Init()* is used. Then that value is put into *TMR0H* to vary the speed of rotation.

****

****

* 1. **Lesson 12: CCP PWM**

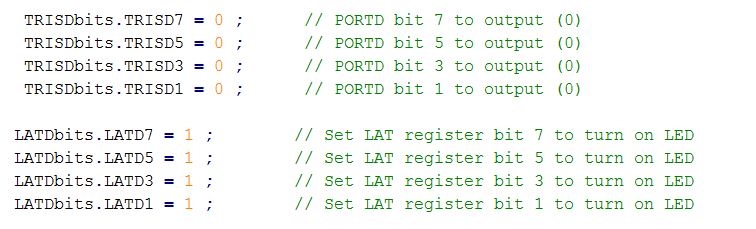
The objective of this lab is to familiarize with Pulse Width Modulation function. It is assigned to be used to dim and brighten a LED with a pre-scaled timer clock. The PWM is accessed with the *PR2, CCPR1L,* and *CCP1CON* with their values to set the PWM frequency, read the 8 most significant bit, and signal out to the specified LED respectively. The value of the brightness will be incremented with the timer and output to the LED with *CCPR1L* line, which differs from a normal high and low signal. This will allow the light to shine brighter and dimmer.

****

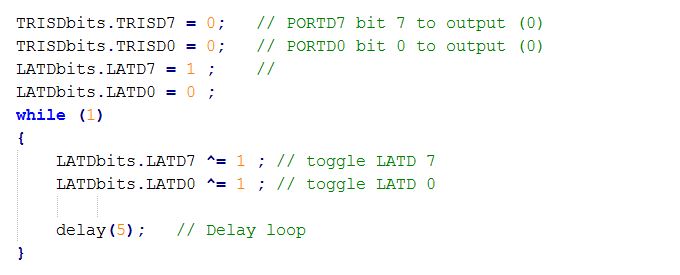
**12_1.jpg**

1. **MicroChip Lab Assignments**
   1. **Lab 1: Turn on 3 LEDs**

By applying the concept from lesson 1, the LEDs were turned on by not using the single *TRISD* line of code. Instead, the registers were individually turned to port to output one by one as shown in the code below. Once we got the desired LEDs to become outputs, we turned them on using a *LATD* line of code for each LED and specifying which ones to turn on.

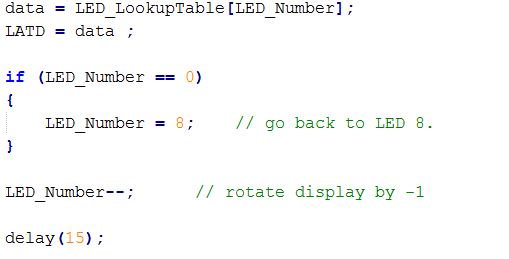
****

* 1. **Lab 2: Change Blinking Speed from LED**

By applying lesson 2 of the programs to this lab, we can simply change the delay speed of the code to change the blinking speed from the LED. The source code for lab 2 is provided below:****

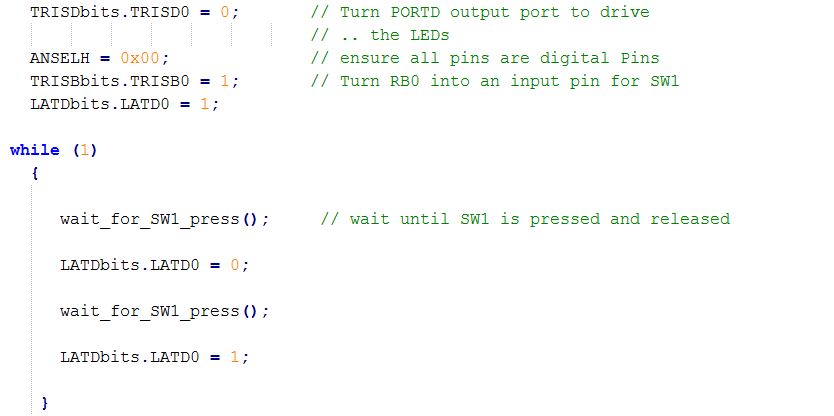
* 1. **Lab 3: Change Direction in which LEDs are Rotated**

To accomplish this part of the experiment, we had to change the rotation of the LEDs. By decrementing the LED instead of incrementing and changed the ‘if’ statement so that the LED would jump from the opposite side to the beginning, it worked as the assignment requires. The source code for lab 3 is provided below:

****

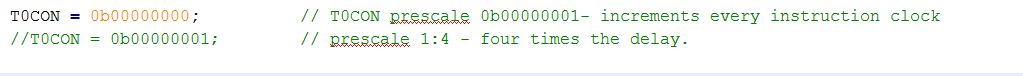
* 1. **Lab 4: Change from Rotate to “Toggle”**

The objective of lab 4 is to combine lesson 1 through 4 to toggle a LED from its rotation mode with an input switch button. The source code is provided below:

****

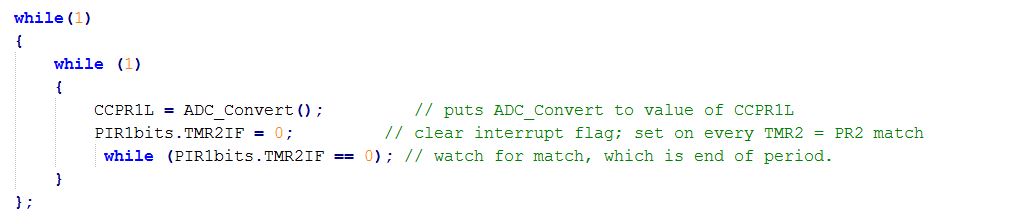
* 1. **Lab 5-6: Lower Rotation Speed**

The objective of this lab is to lower the rotation speed of the LEDs. By changing the pre-scale timer of the clock, its speed would be proportionally reduced, but complicated in the fact that you have to know what pre-scale you want. Changing the value of *T0CON* will change the speed, but it really is the pre-scale value that is important.

****

* 1. **Lab 7, 12: Combination**

The objective of this lab is to ensure that changing the ADC value (turning the POT) will change the brightness of the LED as well. In order to do that, combining lesson 7 and lesson 12 are recommended to achieve the desired performance. Lesson 7 implements the previous methods from lesson 1 through 6 and uses the on-chip Analog-to-Digital Converter to read the demo board potentiometer voltage. An ADC converts an analog voltage level into a digital number representing the voltage. As a result the ADC converter varies the LED time delay so the potentiometer controls the LED rotation speed. Lesson 12 introduces the idea of the Pulse Width Modulation (PWM) functionality of Capture/Compare/PMW (CCP) peripheral. The Pulse Width Modulation is a square wave of a given frequency where the duty cycle of the period is varied. The duty cycle is a ratio of how long the signal is high to the total length of the period. The function of this lesson is to control the brightness of the demo board LED by the output of the PMW. Upon implementing the PWM the LED will only turn on during the high portion of the PWM period, and off during a low period. Revised code for implementing the PWM on the LED with ADC(Analog to Digital Converter), so that by turning the Potentiometer the brightness of the LED will change. The source code for this lab is provided below:

****

**Conclusion**

Microchip PICkit3 is a very interesting microcontroller to explore with a lot of powerful functions. By providing a great integrated development environment along with thorough and deep instructions, Microchip PICkit3 has claimed its position and how serious it is in the microcontroller family. The most interesting concept learned from this lab is to be able to code in register/memory level in the simple and user-friendly IDE. Doing that way helped me to learn both programming skills and memory understanding.