Microprocessor Systems II & Embedded Systems

Course Number: EECE.4800

Lab 1: Sensor Design and Analog Digital Conversion

Instructor: Professor Yan Luo

Group #10

Derek Teixeira

October 2, 2017

Handed in October 2, 2017

Contributions

Group Member 1 – Derek Teixeira(Me)

Was the member that was at all of the necessary parts and meetings of the group project. Held onto the PICKit supplies and brought to every meeting in the labs. Main duty was to establish connection from the PICKit to the pic16 chip which was all hardware based. Setup all of the connections including the 3.3-volt power, ADC inputs and outputs, and the PWM motor connection. Figured to put a voltage divider of about 10k with the photo resistor; did with Hans. On the coding part was helping Hans with Initiating the ADC variables to get the ADC to work on the pic16 chip. Tried to help Kyle with the initializations on the PWM but our timer never worked.

Group Member 2 – Kyle Marescalchi

Was placed in charge of the PWM for the servo motor part of the project. Did his coding over the weekend and all hands-on deck tried to help him out on Monday to get the initializations to work. Never could get the initializations for the PWM to work so Kyle and Hans worked on a counter for the PWM. Main contributions was trying to get the PICKit PWM to work and helped in the end with Hans on our counters duty cycle with a counter loop.

Group Member 3 – Hans Edward Hoene

Was the main coder for the project, setup both a counter for the ADC conversion and the PWM module. Hans setup the GitHub which was essential in seeing each group members progress and helped out on troubleshooting each of our problems; hardware and software based. I helped Hans with the LED while Kyle helped Hans with the PWM module.

Lab Purpose

Lab 1 was to give the students a very sophisticated circuit that would test all of the group members to work together to get a big project done. This lab had design sensors, operations of ADC, embedded systems, PWM module, and a servo motor. If all of these units were synchronized into a circuit it would eventually create a system that would rotate a motor that would turn on and off an LED with a servo motor. Being able to understand registers, coding, and the hardware of the PICKit was essential in getting the lab 1 complete.

Introduction

Lab 1 is to get the student to figure out how to use the PICKit3 with the pic16 chip to do a couple a simple yet complicated actions. The first action is to turn and LED on and off with the effect from the light hitting a photo resistor. This is done by having a voltage divider calculated to figured out what resistor with be added with the photo resistor to get between 0 and 3 volts output. Once the ADC is configured a PWM module can be initialized or written with a delay or a counter. When differing duty cycles in the servo motor to the connected PWM module, it will rotate the motor 180 degrees left then right 180 degrees. Not much is given to the student and being able to read data sheets and understands the coding for I/O and the PICKit3 MPLAB helps tremendously.

Materials

- PICKit3 microchip, Model# 10-00424-R7, S/N# BUR142550918, Operating Voltage (1.8-5 Volts)
  Used to Communicate code and initializations between a PC interface and pic chip.
- Analog Discovery 2, Model# 210-321, S/N# 210321AZE323, Operating Voltage (5 volts)
  Was sometimes used to check voltages, square wave pulses, and supply voltages.
- Benchtop Power Supply-GWINSTEK GPD-3303P, S/N# EL920014.
  Used to power fully circuit. Outputted 3.3 volts to the circuit board to power the pic16 chip and outputted a separate 3.3 volts with a higher current to power the servo motor.
- Breadboard R.S.R Electronics
  Used for everything hardware related. Pins for pic16, PICKIT3, servo motor, Photoresistor, LED, and power/ground.
- Pic16f18857, Chip# 1621RW6
  Brains of the operation. Is the central unit for all processes sent to LEDS, Motors, or pins.
- Micro Servo Motor, Model name Longrunner MG 90s
  Can rotate a whole 180 degrees in each direction. Placed a sticky pad on to block and unblock a photoresistor connected to an LED.

Schematic

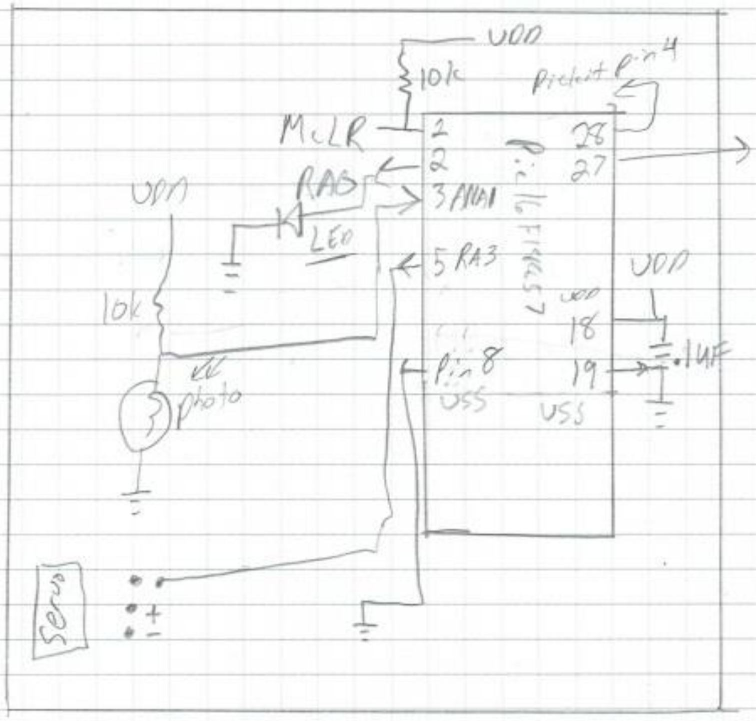
Written:

_________ - 121
BOOK PAGE

TITLE PROJECT

Continued from page

Schematic



Pickit3

- ▶ → MCLR
- → VDD
- → grond
- → pin 28/ ICSP Dat/ PGD
- → pin 27/ ICSPCLK/PGC
- X → Low Voltage not used

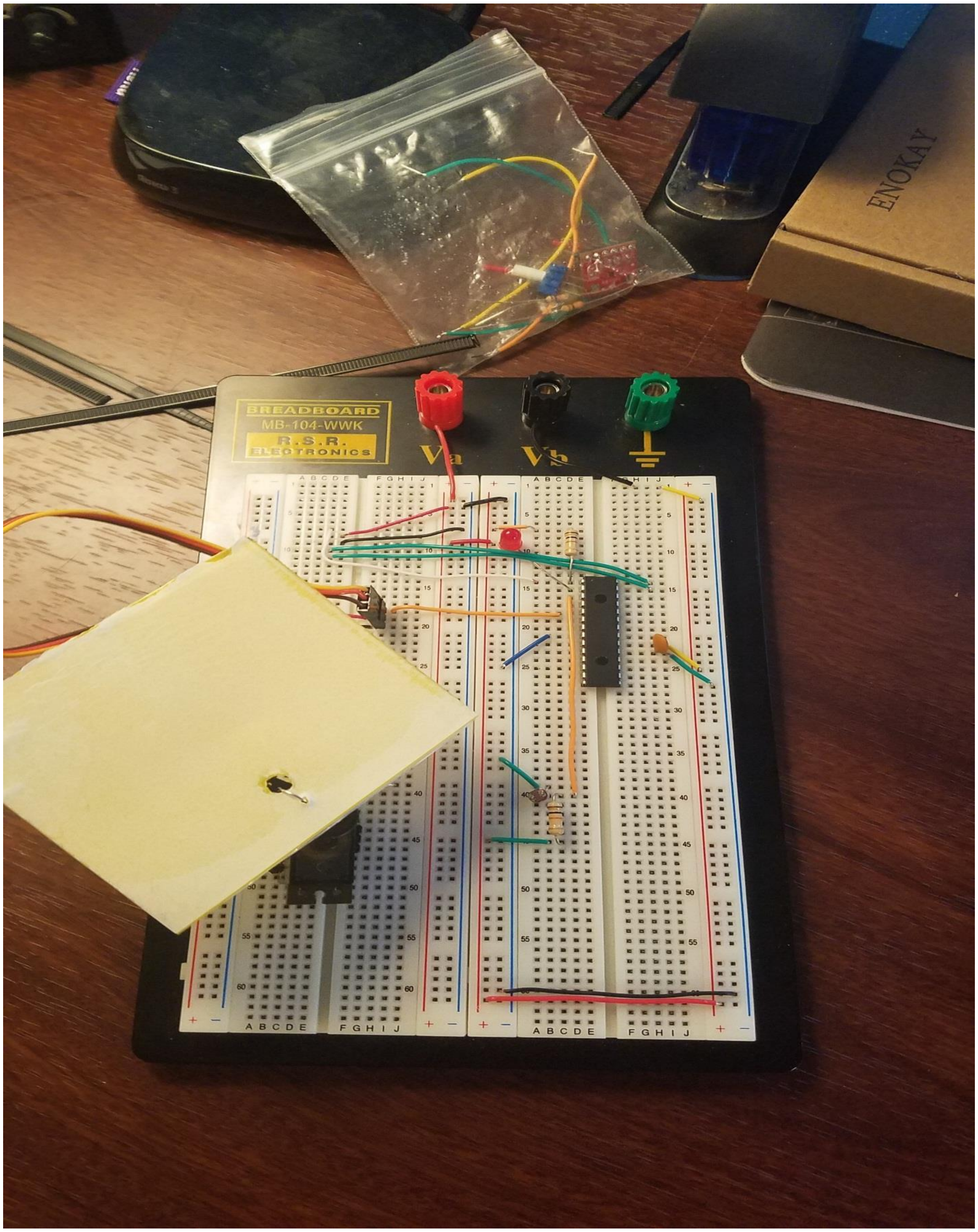
VDD is set at 3.3 volts from the power supply. Servo motor is powered by a separate 3.3 volts because it needs more amps than the pic16 chip.

Continued to page

SIGNATURE	DATE
DISCLOSED TO AND UNDERSTOOD BY	DATE

PROPRIETARY INFORMATION

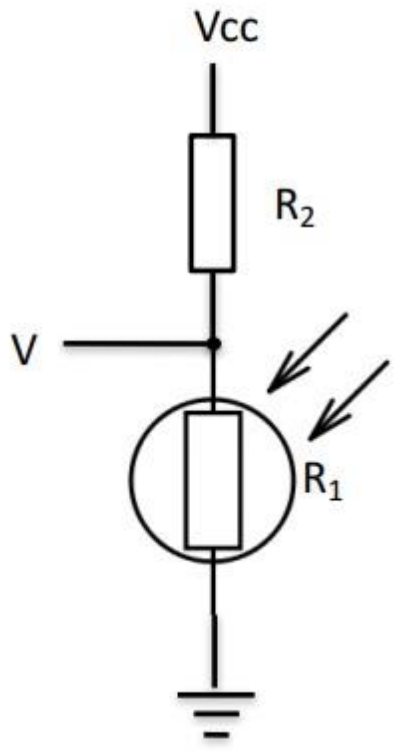
Breadboard



<u>Hardware Design</u>

The hardware was the backbone of the whole lab1 considering everything had to be put onto it. First to do the LED and photo resistor, the voltage divider had to be figured out (as scene bellow this paragraph). When the voltage would read around 1 volt when the resistor had light on it and 2.5 volts when in the dark, the group then could determine our LED Threshold. The R2 we found that would provide a good enough division was a 10k ohm resistor. This seemed to be a good working range once the ADC part of the circuit was setup correctly. This V from the diagram was fed into the 3 pin down from the right, also known as ANA1. The LED comes out of pin RAO1 which is in between the MCLR and analog input pins.

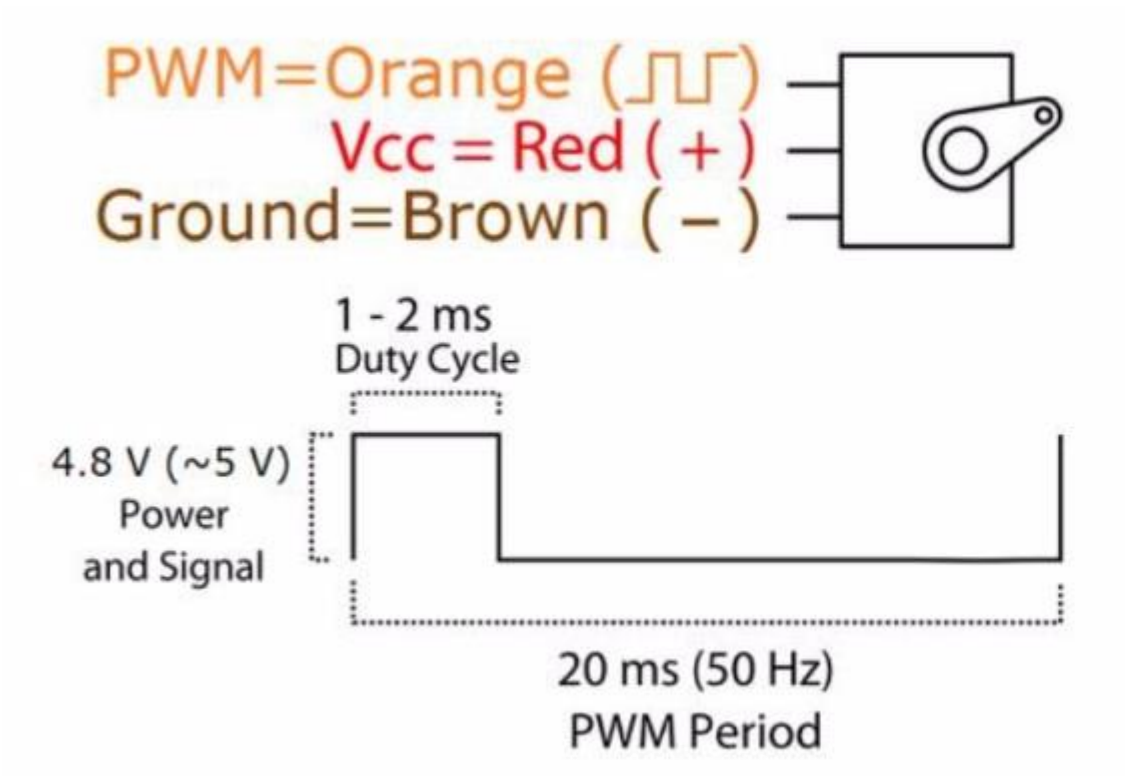
$$V=Vcc \times R_1/(R_1+R_2)$$



Some issues arouse when setting up the wires on the circuit. These will be talked about in the troubleshoot area. The working schematic is the one posted in the schematic sections. For a brief description of the circuitry I will start out with the PICKit3 to board interface.

The PICKit3 has 6 pins coming out of it and only the top five, from the white arrow down, are needed. The top pin, pin# 1 is used for the MCLR which is the power from the PICKit3 itself and needs a 10k resistor connected to the VDD to work correctly. This is plugged into the top left pin on the pic16 chip. Pin# 1. The next pin# 2 on the PICKit is VDD or power to the target. This will just be connected to our exterior power source that is needed to power the breadboard. Pin # 3 on the PICKit3 is the ground pin; also attached to the breadboard for a common ground for the circuit. Pin #4 is the data pin which will connect to pin 28 on the pic16 chip. Pin 5 will go right under pin 4 in the # 27 slow on the pic16 chip.

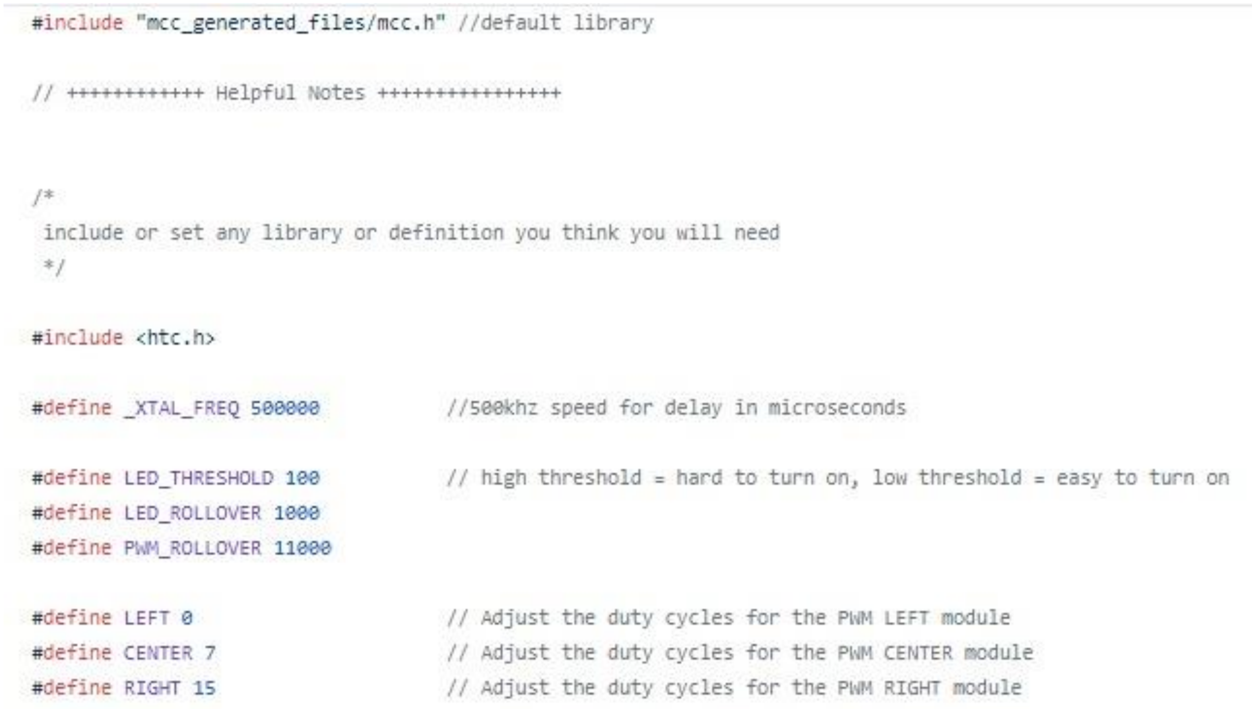
Since the PWM designated by the PICKit3 code was not working we see up pin RA3 to output our duty cycles to the servo motor. As you can see in the schematic the servo has the pin from the pic16 chip connected to its orange cord, while the power and ground are fed from the breadboard.



Software Design

Device Modules

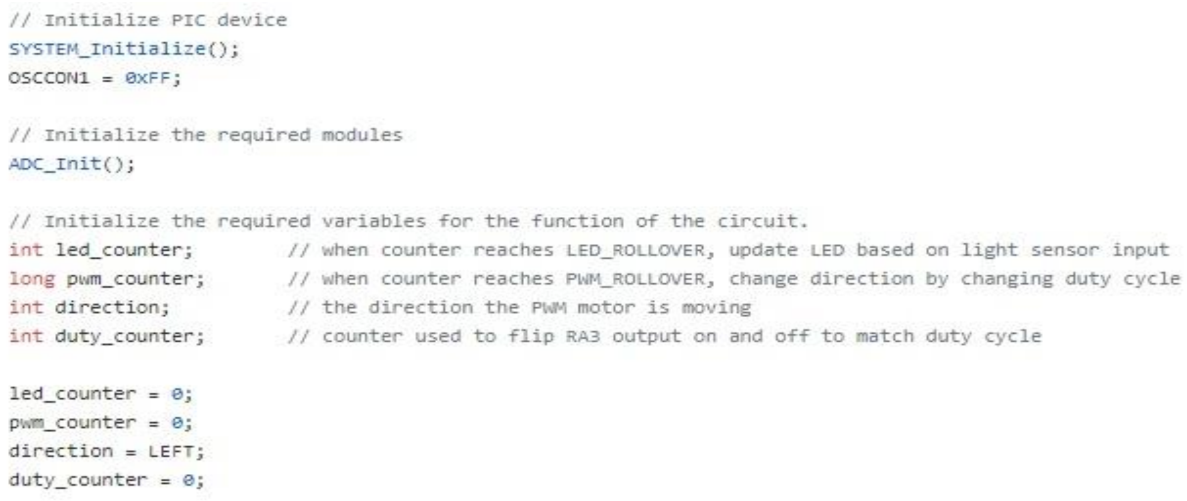
DEFINE VALUES



```
#include "mcc_generated_files/mcc.h" //default library

// ++++++++++++ Helpful Notes ++++++++++++++++


/*
 include or set any library or definition you think you will need
 */

#include <htc.h>

#define _XTAL_FREQ 500000          //500khz speed for delay in microseconds

#define LED_THRESHOLD 100          // high threshold = hard to turn on, low threshold = easy to turn on
#define LED_ROLLOVER 1000
#define PWM_ROLLOVER 11000

#define LEFT 0                     // Adjust the duty cycles for the PWM LEFT module
#define CENTER 7                   // Adjust the duty cycles for the PWM CENTER module
#define RIGHT 15                   // Adjust the duty cycles for the PWM RIGHT module
```

These defined values are the staple of what our system runs off of. These values keep controller of our counters and when changed can affect the circuit dramatically. Header files are included which connect the given skeleton code to our code. Changing the LED threshold would make the LED hard to turn on or off. The rollover values would extent the time that the code would stay counting in the loop. The LEFT, RIGHT, and CENTER were our duty cycle values.

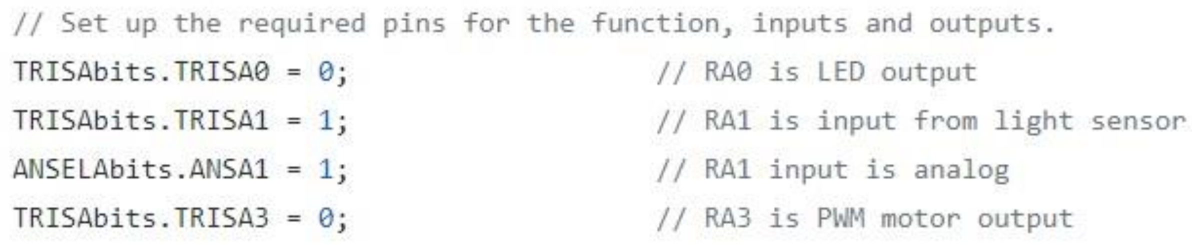
DEFAULTed VALUES



```
// Initialize PIC device
SYSTEM_Initialize();
OSCCON1 = 0xFF;

// Initialize the required modules
ADC_Init();

// Initialize the required variables for the function of the circuit.
int led_counter;        // when counter reaches LED_ROLLOVER, update LED based on light sensor input
long pwm_counter;       // when counter reaches PWM_ROLLOVER, change direction by changing duty cycle
int direction;          // the direction the PWM motor is moving
int duty_counter;       // counter used to flip RA3 output on and off to match duty cycle

led_counter = 0;
pwm_counter = 0;
direction = LEFT;
duty_counter = 0;
```

The defaulted values are ran at the beginning of our main loop and basically set everything to 0 or in the case of the PWM it sets the direction to left.

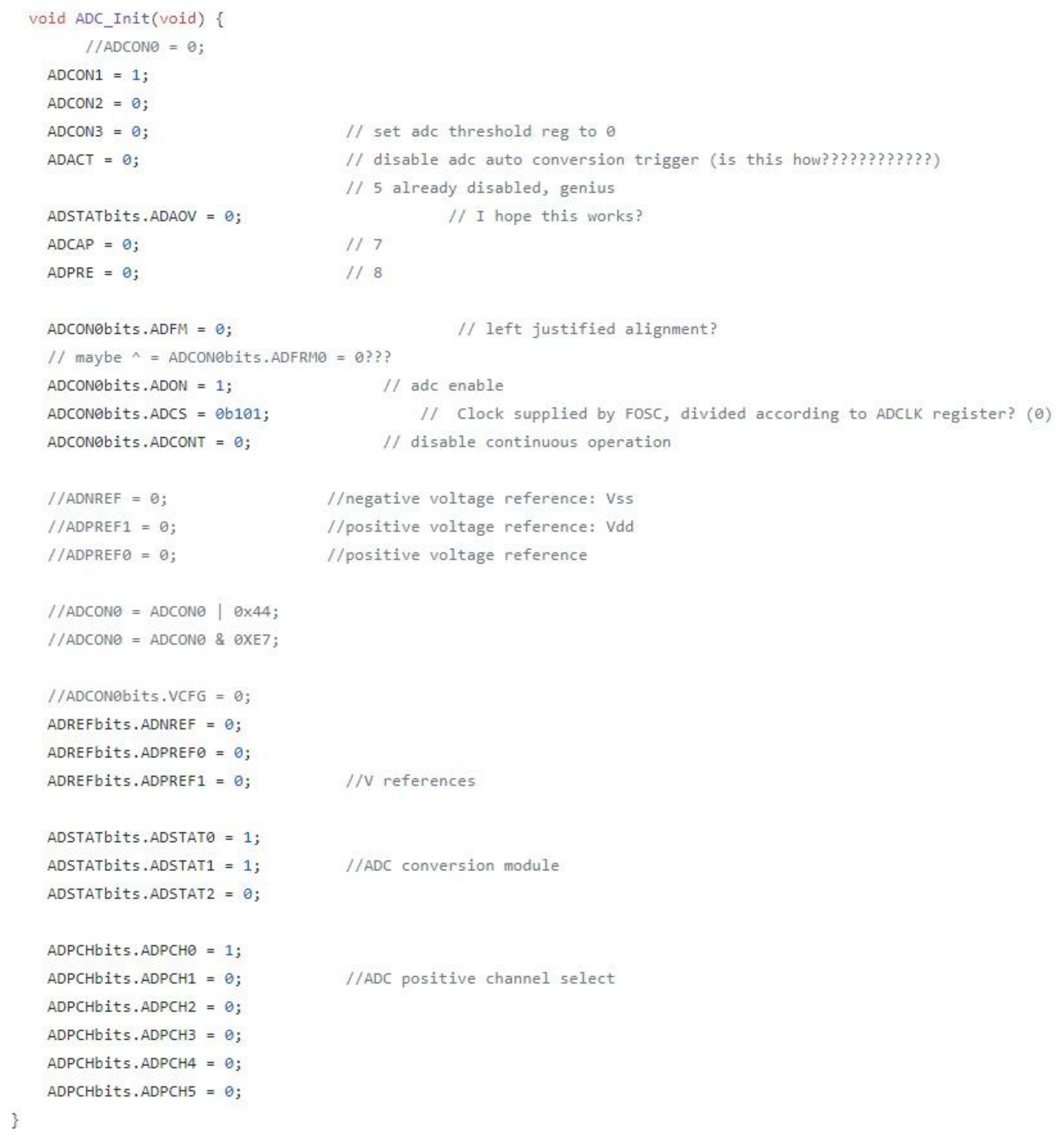
I/O



```
// Set up the required pins for the function, inputs and outputs.
TRISAbits.TRISA0 = 0;                   // RA0 is LED output
TRISAbits.TRISA1 = 1;                   // RA1 is input from light sensor
ANSELAbits.ANSA1 = 1;                   // RA1 input is analog
TRISAbits.TRISA3 = 0;                   // RA3 is PWM motor output
```

Without I/O pins nothing on the chip would read or write to our components. TRISA will enable the pins on the A register of the pic16 chip. While the TRISA0 chooses the number pin in the A register's region. Finally the = 1 will make that pin an input while the = 0 will make it an output pin.

ADC Initialization



```
  void ADC_Init(void) {
        //ADCON0 = 0;
    ADCON1 = 1;
    ADCON2 = 0;
    ADCON3 = 0;                     // set adc threshold reg to 0
    ADACT = 0;                      // disable adc auto conversion trigger (is this how????????????)
                                    // 5 already disabled, genius
    ADSTATbits.ADAOV = 0;                      // I hope this works?
    ADCAP = 0;                      // 7
    ADPRE = 0;                      // 8

    ADCON0bits.ADFM = 0;                        // left justified alignment?
    // maybe ^ = ADCON0bits.ADFRM0 = 0???
    ADCON0bits.ADON = 1;                // adc enable
    ADCON0bits.ADCS = 0b101;                //  Clock supplied by FOSC, divided according to ADCLK register? (0)
    ADCON0bits.ADCONT = 0;              // disable continuous operation

    //ADNREF = 0;                 //negative voltage reference: Vss
    //ADPREF1 = 0;                //positive voltage reference: Vdd
    //ADPREF0 = 0;                //positive voltage reference

    //ADCON0 = ADCON0 | 0x44;
    //ADCON0 = ADCON0 & 0XE7;

    //ADCON0bits.VCFG = 0;
    ADREFbits.ADNREF = 0;
    ADREFbits.ADPREF0 = 0;
    ADREFbits.ADPREF1 = 0;          //V references

    ADSTATbits.ADSTAT0 = 1;
    ADSTATbits.ADSTAT1 = 1;         //ADC conversion module
    ADSTATbits.ADSTAT2 = 0;

    ADPCHbits.ADPCH0 = 1;
    ADPCHbits.ADPCH1 = 0;           //ADC positive channel select
    ADPCHbits.ADPCH2 = 0;
    ADPCHbits.ADPCH3 = 0;
    ADPCHbits.ADPCH4 = 0;
    ADPCHbits.ADPCH5 = 0;
}
```

This part of the coding involves initializing the ADC to read a value coming from our voltage divider with a 10k ohm resistor and the photo resistor. Basically, these commands set the ADC in the PICKit3 to:

- ADCON1 = 1, ADCON2 = 0, ADCON3 = 0.
  These setup the ADC control registers for the pic16 chip

- ADACT = 0;
  Will removed the conversion auto trigger.
- ADACT = 0;
  Disables the ADACT
- ADSTATbits.ADAOV = 0;
  Cleared the ADAOV ACC
- ADFM = 0;
  Left justified alignments
- ADCS = 0b101;
  FOSC timer setting
- ADCIBT = 0;
  Continuous operation disabled.
- ADREF
  To set the + and – reference voltages.

- All of the ADSTATbits and ADPCHbits may not be needed but are kept to not mess with a working code.

PWM MODULE AND ADC CODE

Flow Chart

Trouble Shooting

ISSUE #1

The first issue arose when trying to get the PICKit3 to communicate with the pic16 chip on our breadboard. This was first introduced to the ground on Thursday the week before the demo and was still an issue on the next day which was Friday before the demo date. The group as a whole did not know how to get power to the chip and nothing was communicating between the PICKit3 and the targeted chip device.

This was solved by spending all day Friday working to get this done and eventually emailing the TA. He suggesting a website that is in the resources at the end of the lab report. The group stop trying to get the board to communicate on Friday and went home to fight another day. While I was home(Derek) I was able to follow the TA's instructions step by step and eventually got the board to communicate with our PICKit3.

ISSUE #2

Hans had made coding for the ADC module right after I was able to get the pic16 and PICKit3 to communicate. The LED would blink with his given code but could not receive signal from the photoresistor.

Looking back at the skeleton code, we were able to realized that the initiations suggested for the ADC conversion actually had to be down and researched. Another copy hours spent on learning the pic16 data sheet allowed us to successfully get the LED to power on and off through the ANA1 board on the pic16 chip

ISSUE #3

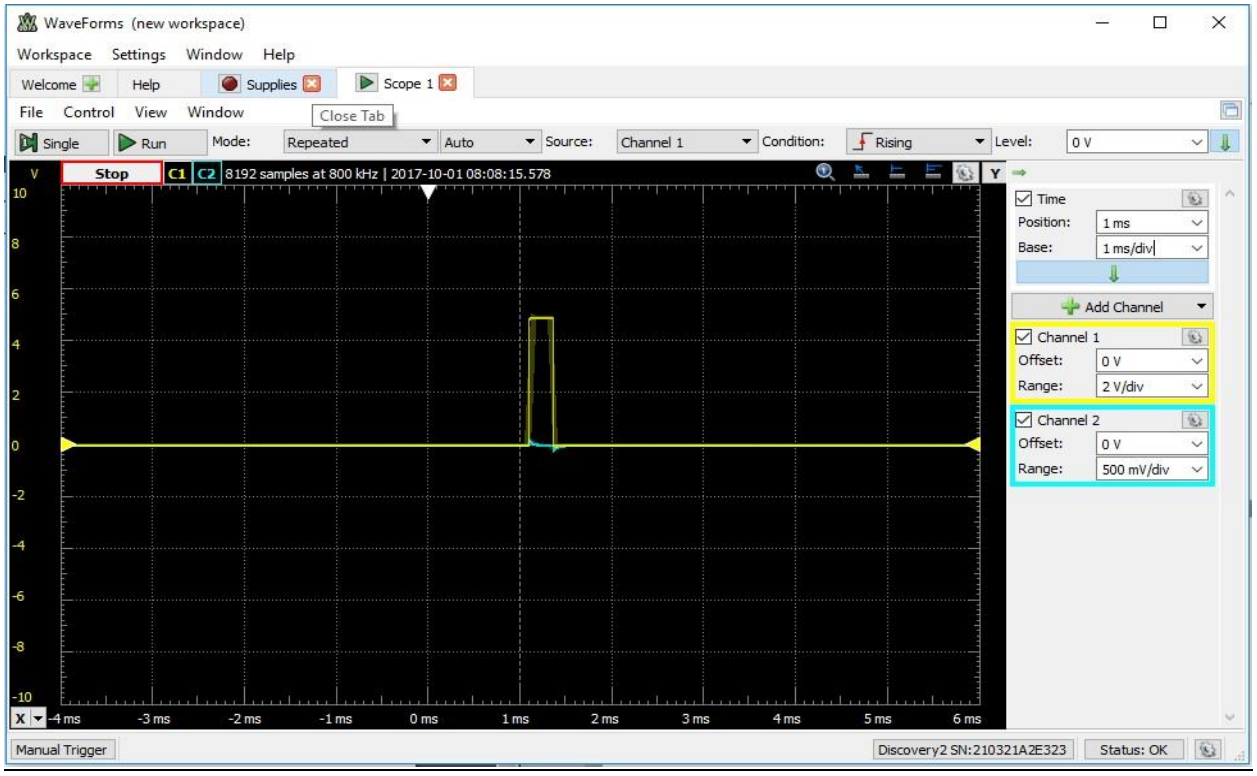
The dreaded PWM module was another on the list which I assumed was with most groups. This was another very pain staking feature that just was actually never figured out by the group as a whole. This was only remedied by not used the PICKit3's PWM module but by making a counter like the ADC module. This was not the hardest thing to do but after reading the next issue you will understand the group's frustration.

ISSUE # 4

Late night on the Tuesday before the demo was due our PICKit3 would compile the same code repeatedly but with different code in our MPLAB IDE console. We had a working pulse coming out of our pic16 chip and then for the next two or so hours while trying to debug and fix the duty cycle, the code was never changing. Called it a night on Tuesday and sent the TA a cry for help.

Another urgent email the TA would solve this issue like that of our first issue. The whole time the MCLR was connected with a 1k resistor. Woke up the next day and threw a 10k in its place and all of the sudden new code was compiling. Another disaster averted thanks to the TA.

OSCILLOSCOPE CAPTURES


WaveForms (new workspace)
Workspace Settings Window Help
Welcome Help Supplies Scope 1
File Control View Window
Close Tab
Single Run
Stop C1 C2
Add Channel
Manual Trigger

Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

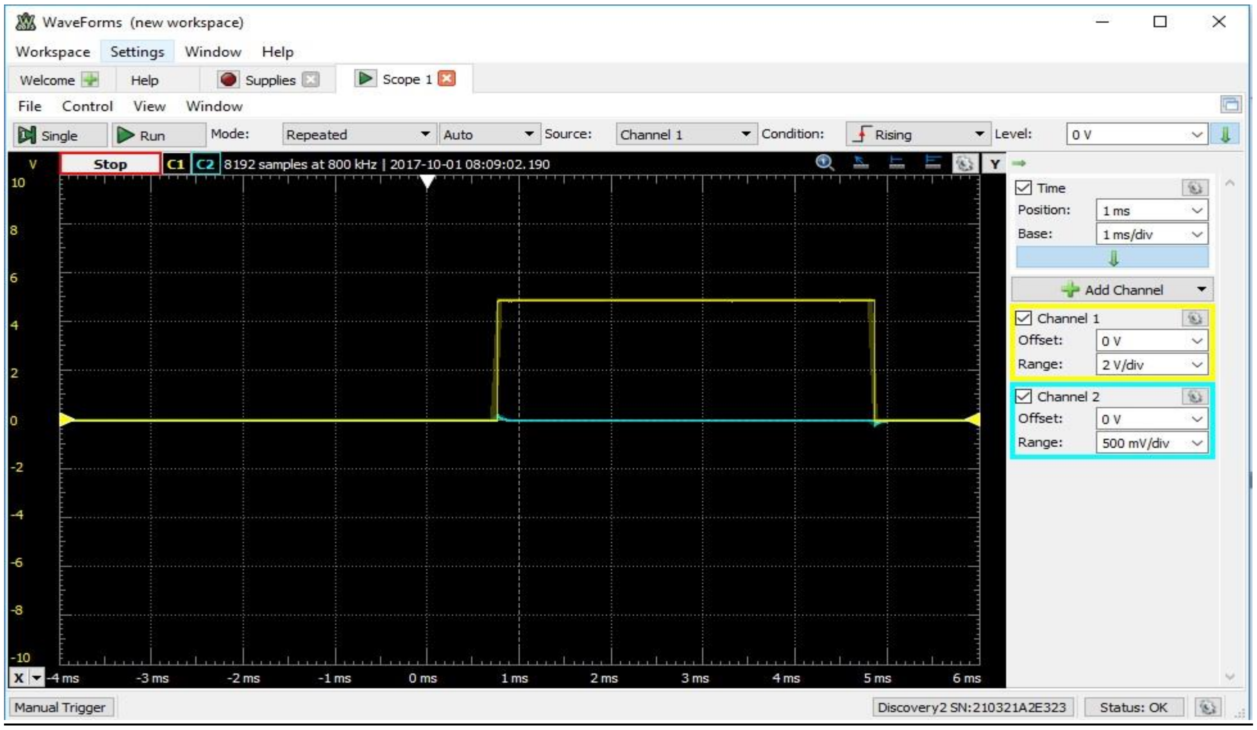
8192 samples at 800 kHz | 2017-10-01 08:08:15.578

Time — Position: 1 ms, Base: 1 ms/div

Channel 1 — Offset: 0 V, Range: 2 V/div

Channel 2 — Offset: 0 V, Range: 500 mV/div

Discovery2 SN:210321A2E323 | Status: OK


WaveForms (new workspace)
Workspace Settings Window Help
Welcome Help Supplies Scope 1
File Control View Window
Single Run
Stop C1 C2
Add Channel
Manual Trigger

Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

8192 samples at 800 kHz | 2017-10-01 08:09:02.190

Time — Position: 1 ms, Base: 1 ms/div

Channel 1 — Offset: 0 V, Range: 2 V/div

Channel 2 — Offset: 0 V, Range: 500 mV/div

Discovery2 SN:210321A2E323 | Status: OK

The top photo is that of the 1% duty cycle which would rotate the motor to the right for the rotation and the bottom is the 15% duty cycle that would rotate the servo to the left. The original plan was to use 10% and 20% instead of 1% and 15% but it never worked out for our motor.

Resources

http://www.instructables.com/id/Programming-PIC-Microcontrollers/

PICKIT3 Data Sheet

PIC16F18857 Data Sheet