UNIVERSITY OF NEVADA LAS VEGAS – DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design Assignment 4

CPE 301 Fall 2016

Luis Ruiz

PART 0

The assignment took me about 5 hours to do. I had to read the datasheet and debug a lot.

1. PART A: Description

DC

My design consists of a C program that drives a DC motor using PWM. The program takes advantage of Analog to Digital converter to read a voltage off a potentiometer that is creating a voltage divider with a resistor. The value of the resistor is 330 ohms as for the potentiometer, it has a max value of 2000 ohms. The program is very responsive to the change of voltage and outputs a PWM based off the voltage read in through the ADC. The logic used is very simple, once a voltage is read in the code has pre-established half way point which determines if the DC motor is to go forward or backwards. The speed of the motor is determined on the read in voltage if it is going forward, the higher the voltage the faster the motor; however, in the backwards direction, the lower the voltage read in the faster the motor goes.

SERVO

My design consists of a C program that drives a Servo motor using PWM. The program takes advantage of Analog to Digital converter to read a voltage off a potentiometer that is creating a voltage divider with a resistor. The value of the resistor is 330 ohms as for the potentiometer, it has a max value of 2000 ohms. The program is very responsive to the change of voltage and outputs a PWM based off the voltage read in. The voltage read is converted using a formula in order to get the range allowed in 180-degree servo motor. Based on the voltage read in, the C code then determines what PWM to output. This PWM is what determines the servo motors degree.

PART B: Code

DC

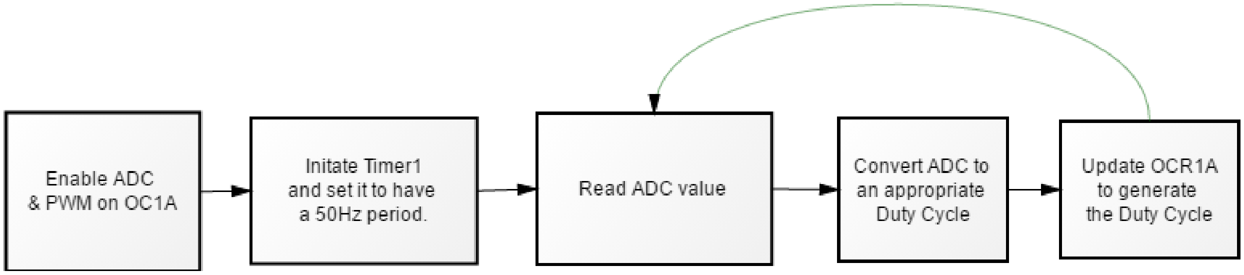
```
/*
 * DA4_DC.c
 *
 * Created: 2/20/2017 2:08:50 PM
 * Author : Luis
 */

#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000UL

unsigned short getADC();                        //get the value read from ADC0
void initateTimer();                            //initialize the Timer and set PWM with a
50Hz
void ADC0init();                                //initialize ADC0 as an input
void update_OC1A(unsigned char);                //update the value of the DC

int main(void)
{
       //temporarily hold the value from the analog channel
       unsigned short val;

       //DUTY CYCLE
       unsigned char DC;

       //set the a nibble as output
       DDRD = 0x0F;

       ADC0init();
       initateTimer();

       while (1)
       {
              //Start conversion
              ADCSRA |= (1 << ADSC);
              //wait conversion to finish
              while((ADCSRA & (1 << ADIF)) == 0);
              //Get value that was converted
              val = ADC;

              if(val > 512)
              {
                     //enable power to 1,2EN
                     //P.D0 and PD.1 for 1A HIGH and 2A LOW
                     //FOWARD
                     PORTD = 0x03;

                     //GET THE DUTY CYCLE BASED ON VOLTAGE IN
                     //Then update OC1A to put DC
                     DC = (unsigned char)((100.0*val)/1023);
                     update_OC1A(DC);

              }
              else
              {
                     //enable power to 1,2EN
                     //P.D0 and PD.1 for 1A LOW and 2A HIGH
                     //BACKWARD
                     PORTD =0x05;

                     //GET THE DUTY CYCLE BASED ON VOLTAGE IN
                     //Then update OC1A to put DC
                     DC = (uint8_t)((100.0*(1023-val))/1024);
                     update_OC1A(DC);
              }
              _delay_ms(700);
       }

       return 0;
}

void initateTimer()
{
       //Set PORTB1 pin as output
       // make OC1A as output.
       DDRB |= (1<<DDB1);
       // Output compare mode on OC1A. Fast PWM with top = ICR1.
       // Clear OC1A on Compare match and set at bottom.
       TCCR1A |=
(1<<COM1A1)|(0<<COM1A0)|(0<<COM1B1)|(0<<COM1B0)|(0<<FOC1A)|(0<<FOC1B)|(1<<WGM11)|(0<<WGM1
0);

       // Start timer with pre-scaler 64
       TCCR1B |=
(0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(1<<WGM12)|(0<<CS12)|(1<<CS11)|(1<<CS10);

       //TOP =  (F_CPU / (N * F_pwm)) - 1, where N is the prescaler = 64, and F_pwm is
the desired 50Hz frequency.
       ICR1 = 4999;
}
void ADC0init()
{
       DDRC   &= ~(0<<DDC0);       // SET PC.0 as an input
       ADCSRA = 0x87;              // Enable ADC and CLK/128
       ADMUX  = (1<<REFS0);        // VCC reference, ADC0 single ended input
}
void update_OC1A(unsigned char DC)
{
       //SET OC1A to create desire Duty Cycle based on the value passed in
       OCR1A = (unsigned short)((DC * 4999.0)/100.0);
}
```

Servo

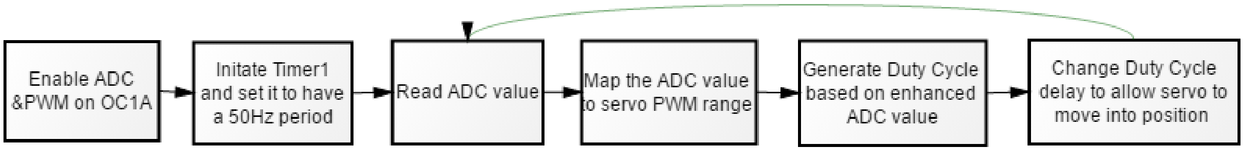
```
/*
 * DA4_Ser.c
 *
 * Created: 2/20/2017 12:32:11 PM
 * Author : Luis
 */

#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000L
#define SERV_MIN 97
#define SER_MAX 535

void initateTimer();                      //initialize the Timer and set PWM with a 50Hz
void ADC0init();                          //initialize ADC0 as an input
float position(unsigned short);           //get the servo position

int main(void)
{
       //temporarily hold the value from the analog channel
       unsigned short val;
       //get the value of Vin
       float servo;

       ADC0init();
       initateTimer();

    while (1)
    {
              //Start conversion
              ADCSRA |= (1 << ADSC);
              //wait conversion to finish
              while((ADCSRA & (1 << ADIF)) == 0);

              if(!(val == (ADCH<<1))){
                            //take the value from the upper bits of the ADC
                            val = ADCH<<1;
              }

              //Get the Vin being read
              servo = position(val);
              //this will determine the position
              servo = (servo*438)+97;

              //The OCR1A value is based on what value ADC0 is reading
              //then develop a PWM based on ADC0 reading
              if(servo < SERV_MIN)
                     OCR1A = SERV_MIN;
              else
                     OCR1A = (int) servo;
    }

       return 0;
}

void initateTimer(){

//TOP = ((Focnx*N)/F_cpu)-1
//Desire Focnx = 50 Hz
ICR1 = 4999;
//ICR1 = (int)((16e6/(50*64))-1);
//SET Timer 1 to have the top to be ICR1
//FAST PWM reading the OCRA in non-inverting mode
//A prescalar of 64 => 16MHz / 64
TCCR1A = (1<<COM1A1)|(1<<WGM11);
TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
}
void ADC0init(){

       //SET ADC0 as an input
       DDRC |= (0<<PC0);

       //PWM pin (OC1A)
       DDRB |= (1<<PB1);

       //SET AVcc with external capacitor at AREF
       //and ADC0 as an input MUX[3:0] = 0b0000
       ADMUX = 0x60;

       //Turn on the ADC for conversion
       //CLKadc/128 = ADPS[2:0] = 0b111
       ADCSRA = 0x87;
       //= (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
       //Run in Free mode
       ADCSRB = 0x0;
}
float position(unsigned short val){
       //ADC = Vin*1024.0/Vref
       //Vin = ADC * 5(Vref)/1024.0
 return ((val*5)/1024.0);
}
```

PART C:Flow Chart

DC

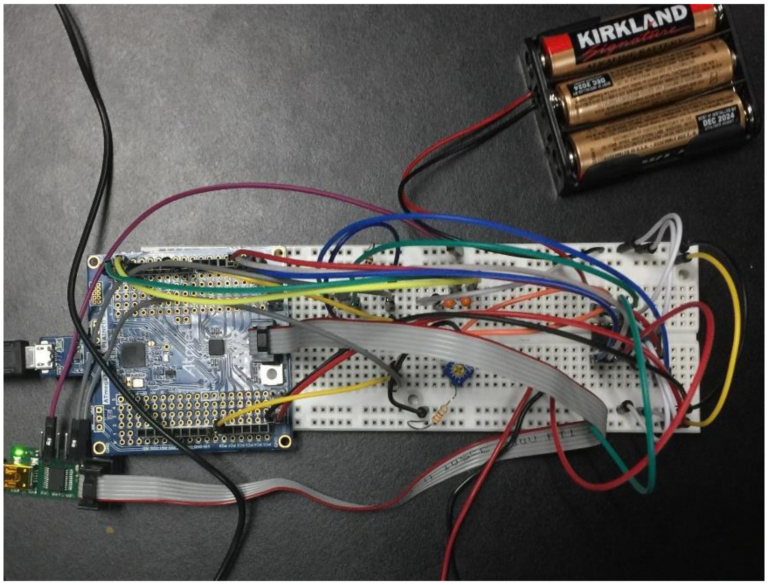


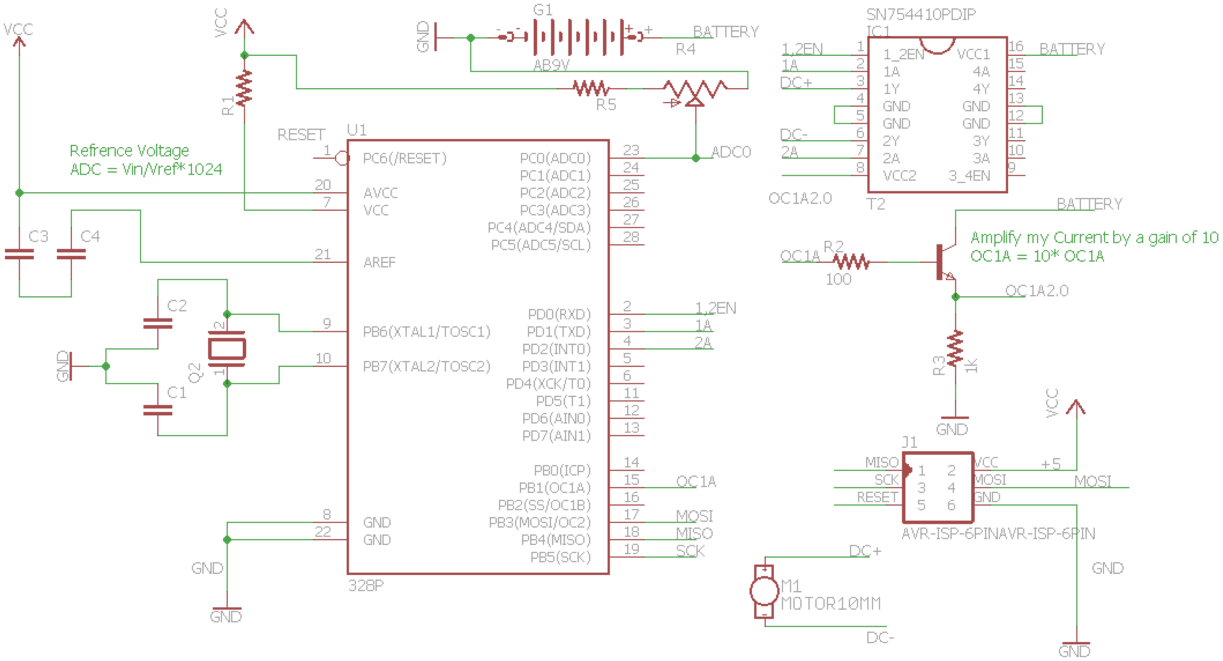
Servo


Enable ADC &PWM on OC1A
Initate Timer1 and set it to have a 50Hz period
Read ADC value
Map the ADC value to servo PWM range
Generate Duty Cycle based on enhanced ADC value
Change Duty Cycle delay to allow servo to move into position

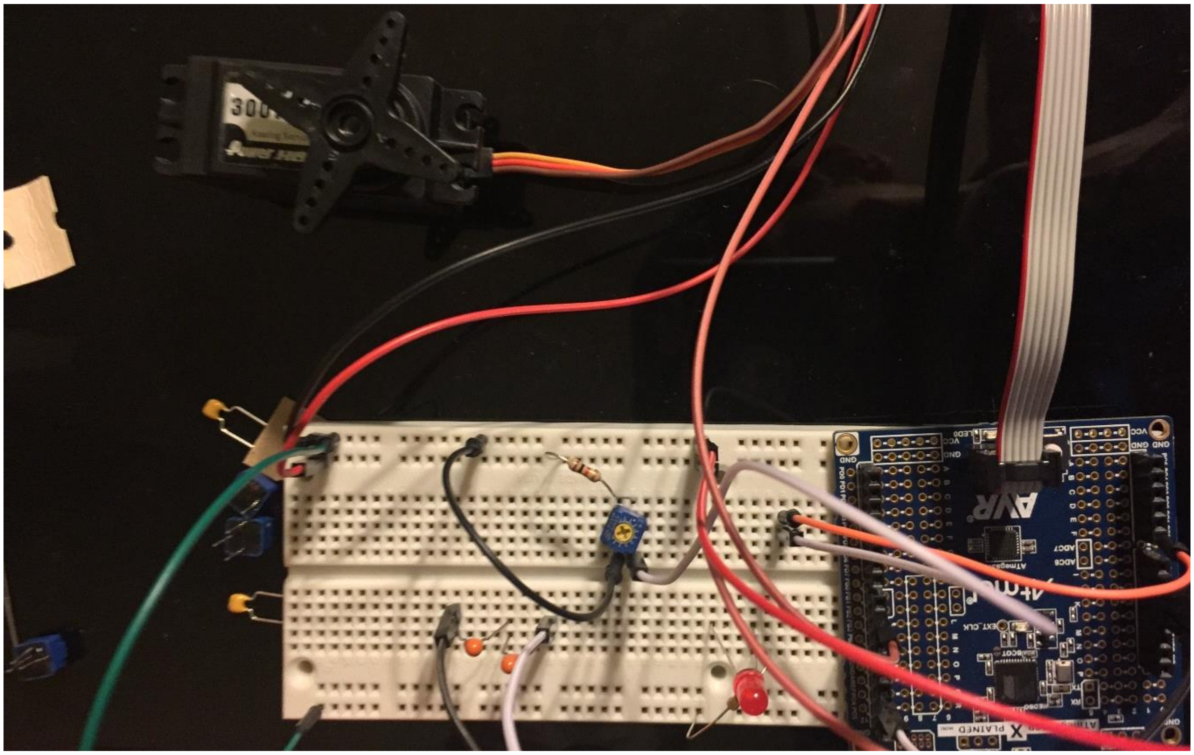
PART D: Schematic

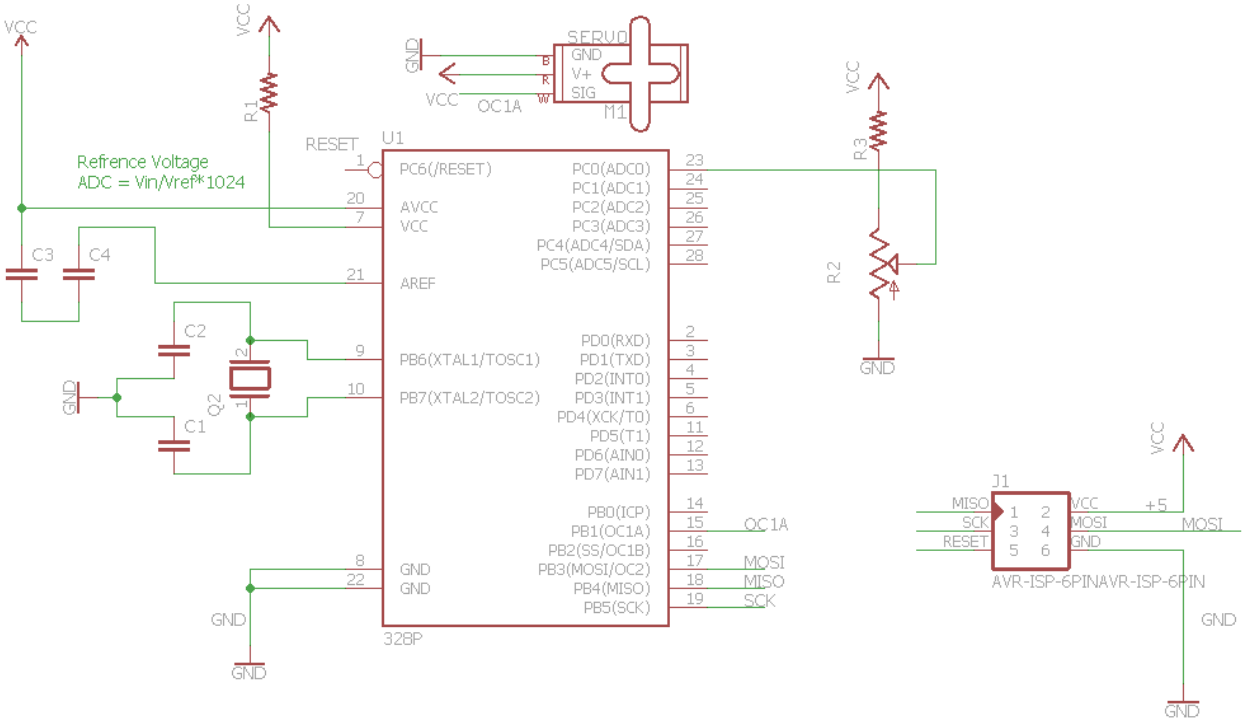
DC





Servo




VCC
VCC
R1
SERVO
GND
V+
SIG
B
R
W
M1
GND
VCC
OC1A
VCC
R3
RESET
U1
Refrence Voltage
ADC = Vin/Vref*1024
1
PC6(/RESET)
20
AVCC
7
VCC
21
AREF
C3
C4
C2
C1
Q2
GND
9
PB6(XTAL1/TOSC1)
10
PB7(XTAL2/TOSC2)
8
22
GND
GND
GND
GND
328P
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
23
24
25
26
27
28
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
2
3
4
5
6
11
12
13
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
14
15
16
17
18
19
OC1A
MOSI
MISO
SCK
R2
GND
J1
MISO
SCK
RESET
1
2
3
4
5
6
VCC
MOSI
GND
+5
VCC
MOSI
AVR-ISP-6PINAVR-ISP-6PIN
GND
GND

PART E: Video

URL Video of Design Assignment 4: https://youtu.be/ycIpYCUF2V0