
REQUIREMENTS NOT MET

N/A

PROBLEMS ENCOUNTERED

N/A

FUTURE WORK/APPLICATIONS

Converting analog to digital signals is used in applications such as:

Using PWM to control motors

Using inverters to convert AC to DC then back to AC

Controller photoresistors

Transmitting bits across lines

and other applications where an analog signal must be converted to a binary signal.

PRE-LAB EXERCISES

i. Why must we use the ADCA module as opposed to the ADCB module?

Our photoresistor is wired to port A of our ATX. Therefore, its not possible to use any other ADC but ADCA

ii. Would it be possible to use any other ADC configurations such as single-ended, differential, differential with gain, etc. with the current pinout and connections of the OOTB Analog Backpack? Why or why not?

No, we can only use differential with gain.

The reason why is because single-ended and differential without gain requires a 16 bit input port Our input port is only 7 bits, so we can only use differential with gain.

iii. What would the main benefit be for using an ADC system with 12-bit resolution, rather than an ADC system with 8-bit resolution? Would there be any reason to use 8-bit resolution instead of 12-bit resolution? If so, explain.

A 12 bit resolution allows a greater range of values we can measure.

However, this requires a longer conversion time.

An 8 bit resolution provides a pretty good range of values for conversion. And its quicker than 12 bit. So you may prefer that if you don't need a super accurate measurement.

iv. What is the decimal voltage value that is equivalent to a 12-bit signed result of 0x360, given a voltage range of -5V to +5V?

With a range of -5V to +5V, and with 4095 possible binary points, the voltage difference between each binary point is .00244 volts.

However, there are only 2047 points for both the positive, and negative side of the voltage axis.

$$.00244 * 0x360(864) = +2.11 \text{ volts}$$

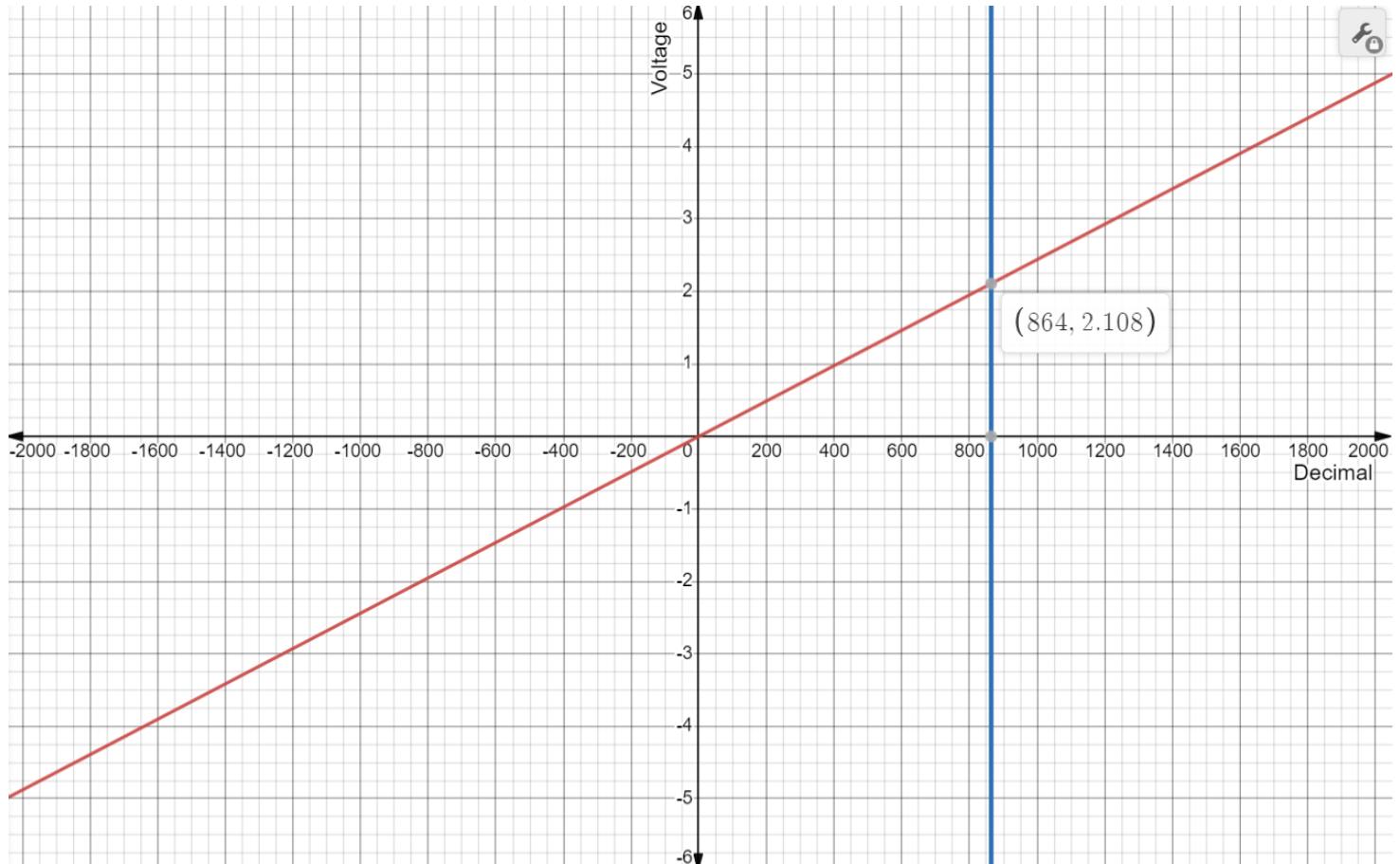


Figure 1: Voltage vs Decimal graph

v. Given an 8-bit signed ADC system with a voltage reference range of -1V to +2V, express the expected digital value in terms of the analog input voltage, using the form $V_D = f(V_A)$.

With a range of 3 volts, and with 255 possible values, we have a slope of .012 volts/decimal.

With $V_A = 2$ volts, and with a slope of .012, then the vertical intercept (b) is equal to -1.06

Therefore, $V_D = .012(V_A) - 1.06$

PSEUDOCODE/FLOWCHARTS

SECTION 1

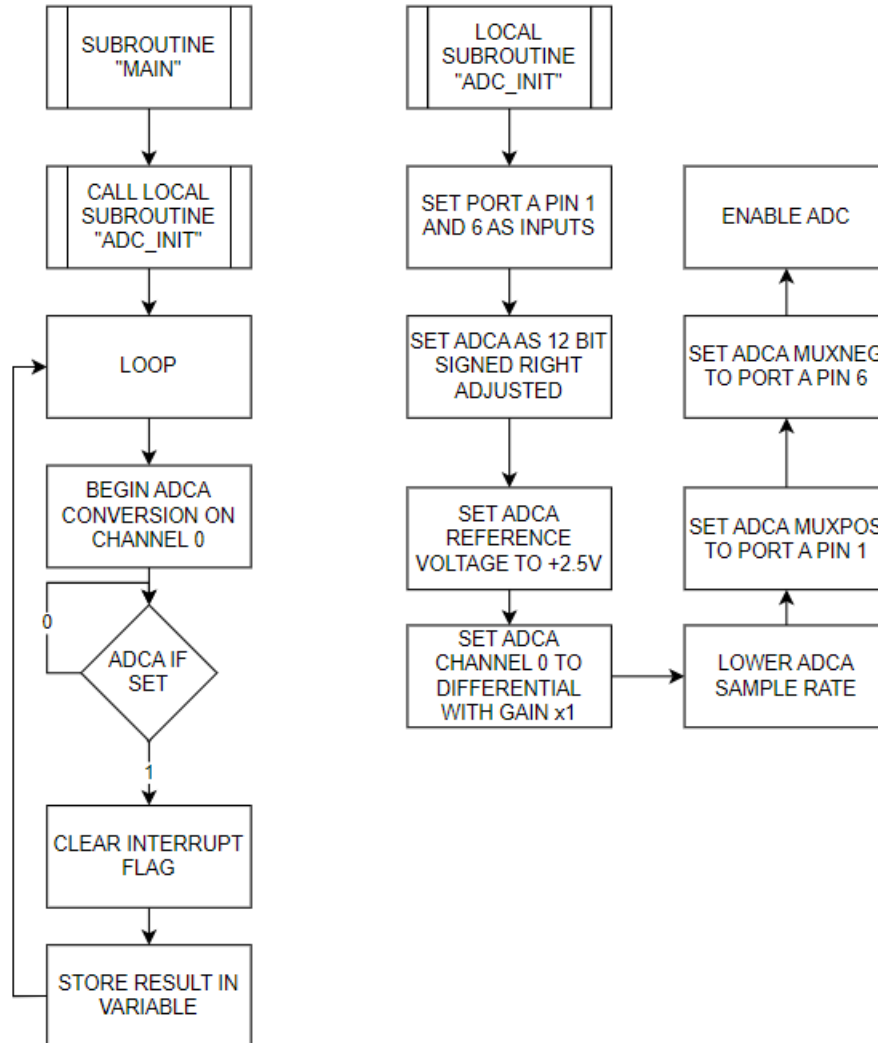


Figure 2: Flowchart for "lab7_1.C"

SECTION 2

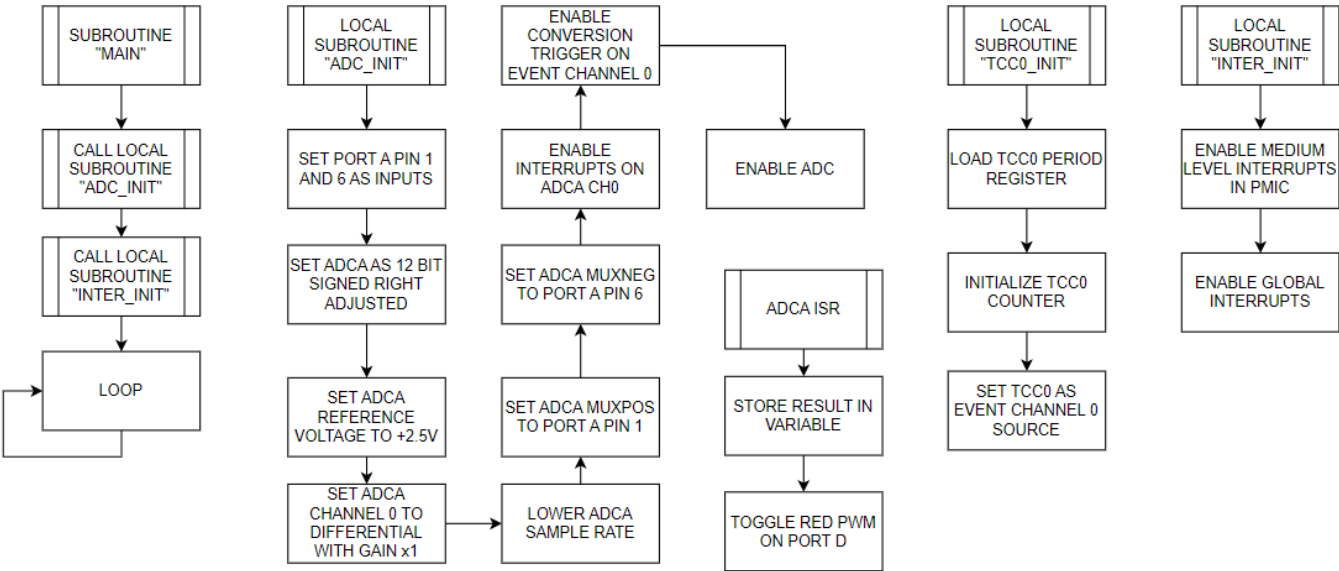


Figure 3: Flowchart for “lab7_2.C”

SECTION 3

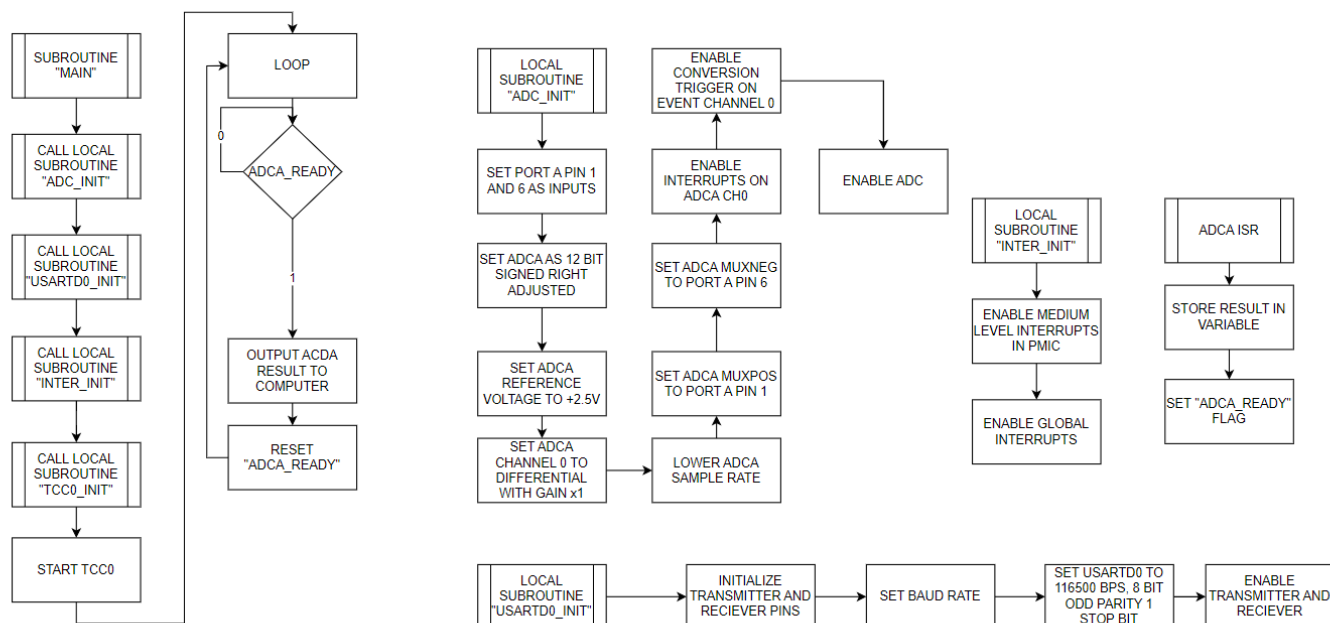


Figure 4: Flowchart for “lab7_3.C”

SECTION 4

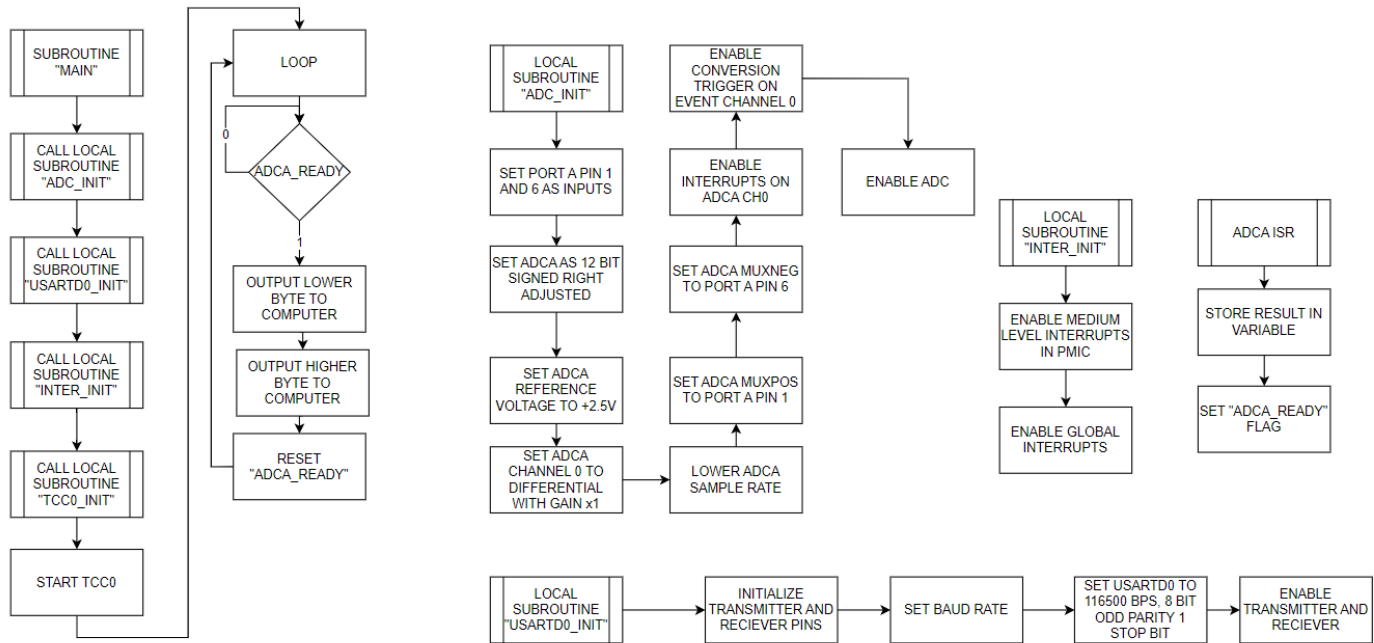


Figure 5: Flowchart for “lab7_4.C”

SECTION 5

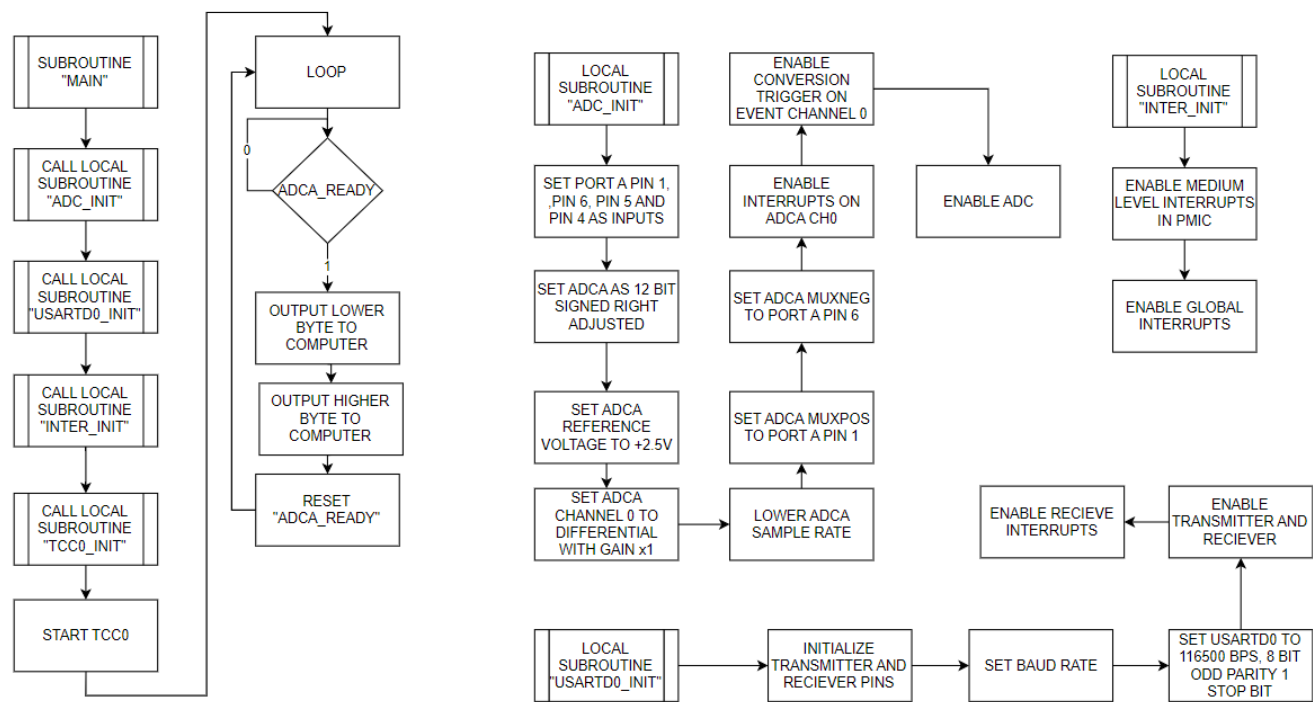


Figure 6: Flowchart for “lab7_5.C”

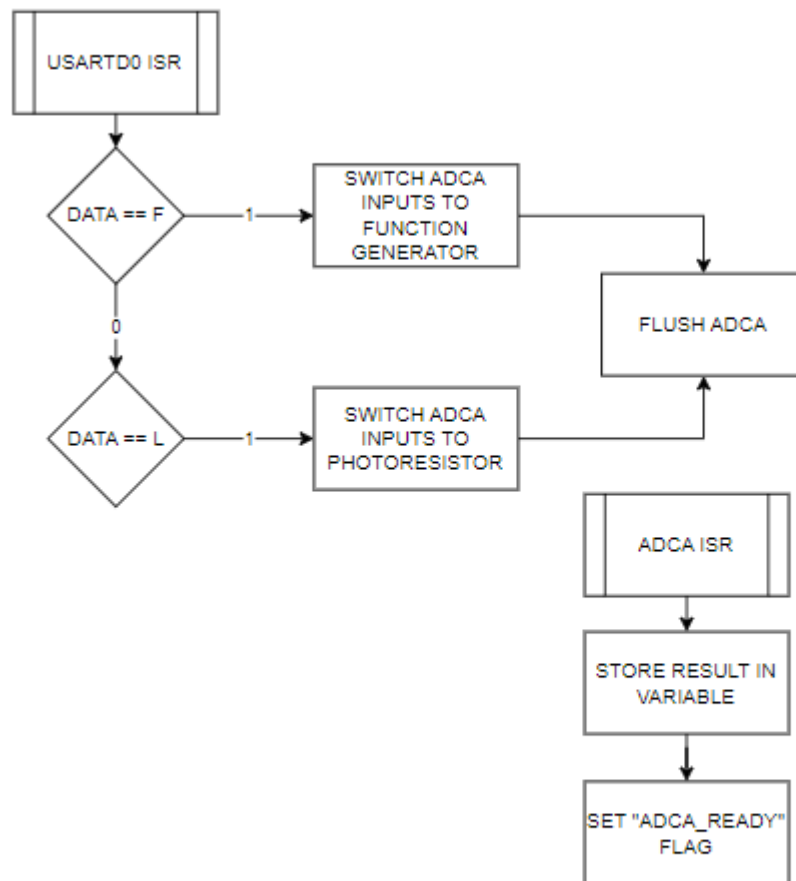


Figure 7: Flowchart for “lab7_5.C”

PROGRAM CODE

SECTION 1

```
//*****
//Lab 7, Section 1
//Name: Steven Miller
//Class #: 11318
//PI Name: Anthony Stross
//Description: setups photoresistor to connect to ADC module A
//*****

#include <avr/io.h>
//definitions
int main(void)
{
    int16_t upperbyte = 0;
    int16_t lowerbyte = 0;
    int16_t data = 0;
    //initialize ADC
    adc_init();
    while (1)
    {
        //begin adca conversion on channel 0
        ADCA.CH0.CTRL |= (ADC_CH_START_bm);
        //check if interrupt flag set
        while(!(ADCA.CH0.INTFLAGS & ADC_CH_CHIF_bm))
        {
            //do nothing
        }
        //clear interrupt flag
        ADCA.CH0.INTFLAGS = ADC_CH_CHIF_bm;
        //store result
        data = (ADCA.CH0.RES);
    }
}

void adc_init(void)
{
    //set port a pin 1 and 6 as inputs
    PORTA.DIRCLR = (PIN1_bm|PIN6_bm);
    //set adca as 12 bit signed right adjusted
    ADCA.CTRLB = (ADC_CONMODE_bm|ADC_RESOLUTION_12BIT_gc);
    //set adca reference voltage to +2.5V
    ADCA.REFCTRL = (0|ADC_REFSEL_AREFB_gc);
    //set adca channel 0 to differential with gain x1
    ADCA.CH0.CTRL = (ADC_CH_INPUTMODE_DIFFWGAIN_gc);
    //lower ADCA sampling
    ADCA.PRESCALER = ADC_PRESCALER_DIV512_gc;
    //set adca muxpos to port A pin 1 and pin 6
    ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN1_gc|ADC_CH_MUXNEG_PIN6_gc);
    //ENABLE ADC
    ADCA.CTRLA = (ADC_ENABLE_bm);
}
```

SECTION 2

```
//*****
//Lab 7, Section 2
//Name: Steven Miller
//Class #: 11318
//PI Name: Anthony Stross
//Description: samples the photoresistor six times per second
//*****
//includes
#include <avr/interrupt.h>
#include <avr/io.h>
//variables
volatile int16_t result;
int main(void)
{
    //initialize ADC
    adc_init();
    //initialize interrupts
    inter_init();
    //initialize tcc0
    tcc0_init();
    //init red led
    PORTD.DIRSET = PIN4_bm;
    //start timer counter
    TCC0.CTRLA = TC_CLKSEL_DIV1024_gc;
    while (1)
    {
        //do nothing
    }
}

void adc_init(void)
{
    //set port a pin 1 and 6 as inputs
    PORTA.DIRCLR = (PIN1_bm|PIN6_bm);
    //set adca as 12 bit signed right adjusted
    ADCA.CTRLB = (ADC_CONMODE_bm|ADC_RESOLUTION_12BIT_gc);
    //set adca reference voltage to +2.5V
    ADCA.REFCTRL = (0|ADC_REFSEL_AREFB_gc);
    //set adca channel 0 to differential with gain x1
    ADCA.CH0.CTRL = (ADC_CH_INPUTMODE_DIFFWGAIN_gc);
    //lower ADCA sampling
    ADCA.PRESCALER = ADC_PRESCALER_DIV512_gc;
    //set adca muxpos to port A pin 1 and pin 6
    ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN1_gc|ADC_CH_MUXNEG_PIN6_gc);
    //enable interrupts on ADCA ch0
    ADCA.CH0.INTCTRL = (ADC_CH_INTMODE_COMPLETE_gc|ADC_CH_INTLVL_MED_gc);
    //enable conversion trigger on channel event 0
    ADCA.EVCTRL = (ADC_EVSEL_0123_gc|ADC_EVACT_CH0_gc);
    //ENABLE ADC
    ADCA.CTRLA = (ADC_ENABLE_bm);
}
```

```
void tcc0_init(void)
{
    uint8_t period = 159;
    uint8_t offset = 6;
    //load tcc0 period register
    TCC0.PER = period + offset;
    TCC0.CNT = 0;
    //set tcc0 as event channel 0 source
    EVSYS.CH0MUX = EVSYS_CHMUX_TCC0_OVF_gc;
}

void inter_init(void)
{
    //enable medium level interrupts in pmic
    PMIC.CTRL = PMIC_MEDLVLEN_bm;
    //enable global interrupts
    sei();
}

ISR(ADCA_CH0_vect)
{
    result = (ADCA.CH0.RESH<<8 | ADCA.CH0.RESL<<0);
    //toggle red pwm led on port D
    PORTD.OUTTGL = PIN4_bm;
}
```

SECTION 3

```
//*****
//Lab 7, Section 3
//Name: Steven Miller
//Class #: 11318
//PI Name: Anthony Stross
//Description: samples the photoresistor every second and outputs it to the computer
//*****
//includes
#include <avr/interrupt.h>
#include <avr/io.h>
//definitions
#define byte1_bm (15 << 4)
#define byte0_bm (15 << 0)
#define resolution (.00122)
#define intercept (.00061)
//variables
volatile int16_t result;
volatile int8_t bsel = 5;
volatile int8_t bscale = -6;
volatile uint8_t adca_ready;
int main(void)
{
    uint8_t int1 = 0;
    uint8_t int2 = 0;
    uint8_t int3 = 0;
    float result2 =0;
    float result3 =0;
    //initialize ADC
    adc_init();
    //initialize tcc0
    tcc0_init();
    //initialize interrupts
    inter_init();
    //initialize usartd0
    usartd0_init();
    //start timer counter
    TCC0.CTRLA = TC_CLKSEL_DIV1024_gc;
    while (1)
    {
        if(adca_ready)
        {
            //output ADCA result to computer

            //*****
            /*OUTPUT VOLTAGE IN DECIMAL VALUE*/
            //*****
            //output positive or negative sign
            if(result < 0)
            {
                result = result*-1;
                USARTD0.DATA = '-';
                while(!(USARTD0.STATUS & USART_DREIF_bm))
                {
                    //do nothing
                }
            }
            else if(result > 0)
            {

```



```
////////////////////////////////////

////////////////////////////////////
/*OUTPUT VOLTAGE IN HEXADECIMAL VALUE*/
////////////////////////////////////
USARTD0.DATA = '(';
while(! (USARTD0.STATUS & USART_DREIF_bm))
{
    //do nothing
}
USARTD0.DATA = '0';
while(! (USARTD0.STATUS & USART_DREIF_bm))
{
    //do nothing
}
USARTD0.DATA = 'x';
while(! (USARTD0.STATUS & USART_DREIF_bm))
{
    //do nothing
}
//CONVERT TO HEX
//get byte 0
int1 = (result & byte0_bm);
if(int1 > 9)
{
    int1 = int1+55;
}
else
{
    int1 = int1+48;
}
//get byte 1
int2 = (result & byte1_bm);
int2 = (int2>>4);
if(int2 > 9)
{
    int2 = int2+55;
}
else
{
    int2 = int2+48;
}
//get byte 3
int3 = (result >> 8);
if(int3 > 9)
{
    int3 = int3+55;
}
else
{
    int3 = int3+48;
}
USARTD0.DATA = int3;
while(! (USARTD0.STATUS & USART_DREIF_bm))
{
    //do nothing
}
USARTD0.DATA = int2;
while(! (USARTD0.STATUS & USART_DREIF_bm))
{
    //do nothing
}
```



```
    }
    USARTD0.DATA = int1;
    while(!(USARTD0.STATUS & USART_DREIF_bm))
    {
        //do nothing
    }
    USARTD0.DATA = ' ';
    while(!(USARTD0.STATUS & USART_DREIF_bm))
    {
        //do nothing
    }
    ////////////////////////////////////
    /*END OUTPUT VOLTAGE IN HEXADECIMAL VALUE*/
    ////////////////////////////////////

    ////////////////////////////////////
    /*CARRIAGE RETURN*/
    ////////////////////////////////////
    USARTD0.DATA = '\r';
    while(!(USARTD0.STATUS & USART_DREIF_bm))
    {
        //do nothing
    }
    ////////////////////////////////////
    /*END CARRIAGE RETURN*/
    ////////////////////////////////////

    ////////////////////////////////////
    /*LINEFEED*/
    ////////////////////////////////////
    USARTD0.DATA = 10;
    while(!(USARTD0.STATUS & USART_DREIF_bm))
    {
        //do nothing
    }
    ////////////////////////////////////
    /*END LINEFEED*/
    ////////////////////////////////////

    //reset adca
    adca_ready = 0;
}
}
```

```
void adc_init(void)
{
    //set port a pin 1 and 6 as inputs
    PORTA.DIRCLR = (PIN1_bm|PIN6_bm);
    //set adca as 12 bit signed right adjusted
    ADCA.CTRLB = (ADC_CONMODE_bm|ADC_RESOLUTION_12BIT_gc);
    //set adca reference voltage to +2.5V
    ADCA.REFCTRL = (0|ADC_REFSEL_AREFB_gc);
    //set adca channel 0 to differential with gain x1
    ADCA.CH0.CTRL = (ADC_CH_INPUTMODE_DIFFWGAIN_gc);
    //lower ADCA sampling
    ADCA.PRESCALER = ADC_PRESCALER_DIV512_gc;
    //set adca muxpos to port A pin 1 and pin 6
    ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN1_gc|ADC_CH_MUXNEG_PIN6_gc);
    //enable interrupts on ADCA ch0
    ADCA.CH0.INTCTRL = (ADC_CH_INTMODE_COMPLETE_gc|ADC_CH_INTLVL_MED_gc);
    //enable conversion trigger on channel event 0
    ADCA.EVCTRL = (ADC_EVSEL_0123_gc|ADC_EVACT_CH0_gc);
    //ENABLE ADC
    ADCA.CTRLA = (ADC_ENABLE_bm);
}

void inter_init(void)
{
    //enable medium level interrupts in pmic
    PMIC.CTRL = PMIC_MEDLVLEN_bm;
    //enable global interrupts
    sei();
}

void tcc0_init(void)
{
    uint16_t period = 977;
    uint8_t offset = 15;
    //load tcc0 period register
    TCC0.PER = period + offset;
    TCC0.CNT = 0;
    //set tcc0 as event channel 0 source
    EVSYS.CH0MUX = EVSYS_CHMUX_TCC0_OVF_gc;
}

void usartd0_init(void)
{
    //initialize transmitter and reciever pins
    PORTD.OUTSET = PIN3_bm;
    PORTD.DIRSET = PIN3_bm;
    PORTD.DIRCLR = PIN2_bm;

    //set baud rate
    USARTD0.BAUDCTRLA = (uint8_t)bsel;
    USARTD0.BAUDCTRLB = (uint8_t)((bscale << 4)|(bsel >> 8));

    //set to 8 bit odd parity with 1 stop bit
    USARTD0.CTRLC = (USART_CMODE_ASYNCHRONOUS_gc | USART_PMODE_ODD_gc |
USART_CHSIZE_8BIT_gc)&(~USART_SBMODE_bm);

    //ENABLE TRANSMITTER AND RECIEVER
    USARTD0.CTRLB = USART_RXEN_bm | USART_TXEN_bm;
}
```

```
ISR(ADCA_CH0_vect)
{
    result = (ADCA.CH0.RESH<<8 | ADCA.CH0.RESL<<0);
    //set ADCA_READY flag
    adca_ready = 1;
}
```

SECTION 4

```
//*****
//Lab 7, Section 4
//Name: Steven Miller
//Class #: 11318
//PI Name: Anthony Stross
//Description: samples the photoresistor 137 times per second and outputs it to the computer
//*****
//includes
#include <avr/interrupt.h>
#include <avr/io.h>
//variables
volatile int16_t result;
volatile int8_t bsel = 5;
volatile int8_t bscale = -6;
volatile uint8_t adca_ready;
int main(void)
{
    int8_t upperbyte = 0;
    int8_t lowerbyte = 0;
    //initialize ADC
    adc_init();
    //initialize tcc0
    tcc0_init();
    //initialize interrupts
    inter_init();
    //initialize usartd0
    usartd0_init();
    //start timer counter
    TCC0.CTRLA = TC_CLKSEL_DIV64_gc;
    while (1)
    {
        if(adca_ready)
        {
            upperbyte = (ADCA.CH0.RESH<<0);
            lowerbyte = (ADCA.CH0.RESL<<0);
            //output adca result to computer
            while(!(USARTD0.STATUS & USART_DREIF_bm))
            {
                //do nothing
            }
            USARTD0.DATA = lowerbyte;
            while(!(USARTD0.STATUS & USART_DREIF_bm))
            {
                //do nothing
            }
            USARTD0.DATA = upperbyte;
            //reset adca
            adca_ready = 0;
        }
    }
}
```

```
void adc_init(void)
{
    //set port a pin 1 and 6 as inputs
    PORTA.DIRCLR = (PIN1_bm|PIN6_bm);
    //set adca as 12 bit signed right adjusted
    ADCA.CTRLB = (ADC_CONMODE_bm|ADC_RESOLUTION_12BIT_gc);
    //set adca reference voltage to +2.5V
    ADCA.REFCTRL = (0|ADC_REFSEL_AREFB_gc);
    //set adca channel 0 to differential with gain x1
    ADCA.CH0.CTRL = (ADC_CH_INPUTMODE_DIFFWGAIN_gc);
    //lower ADCA sampling
    ADCA.PRESCALER = ADC_PRESCALER_DIV512_gc;
    //set adca muxpos to port A pin 1 and pin 6
    ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN1_gc|ADC_CH_MUXNEG_PIN6_gc);
    //enable interrupts on ADCA ch0
    ADCA.CH0.INTCTRL = (ADC_CH_INTMODE_COMPLETE_gc|ADC_CH_INTLVL_MED_gc);
    //enable conversion trigger on channel event 0
    ADCA.EVCTRL = (ADC_EVSEL_0123_gc|ADC_EVACT_CH0_gc);
    //ENABLE ADC
    ADCA.CTRLA = (ADC_ENABLE_bm);
}

void inter_init(void)
{
    //enable medium level interrupts in pmic
    PMIC.CTRL = PMIC_MEDLVLEN_bm;
    //enable global interrupts
    sei();
}

void tcc0_init(void)
{
    uint16_t period = 115;
    uint8_t offset = 0;
    //load tcc0 period register
    TCC0.PER = period + offset;
    TCC0.CNT = 0;
    //set tcc0 as event channel 0 source
    EVSYS.CH0MUX = EVSYS_CHMUX_TCC0_OVF_gc;
}

void usartd0_init(void)
{
    //initialize transmitter and reciever pins
    PORTD.OUTSET = PIN3_bm;
    PORTD.DIRSET = PIN3_bm;
    PORTD.DIRCLR = PIN2_bm;

    //set baud rate
    USARTD0.BAUDCTRLA = (uint8_t)bsel;
    USARTD0.BAUDCTRLB = (uint8_t)((bscale << 4)|(bsel >> 4));

    //set to 8 bit odd parity with 1 stop bit
    USARTD0.CTRLC = (USART_CMODE_ASYNCHRONOUS_gc | USART_PMODE_ODD_gc |
USART_CHSIZE_8BIT_gc)&(~USART_SBMODE_bm);

    //ENABLE TRANSMITTER AND RECIEVER
    USARTD0.CTRLB = USART_RXEN_bm | USART_TXEN_bm;
}
```

```
ISR(ADCA_CH0_vect)
{
    result = (ADCA.CH0.RESH<<8 | ADCA.CH0.RESL<<0);
    //set ADCA_READY flag
    adca_ready = 1;
}
```

SECTION 5

```
//*****
//Lab 7, Section 5
//Name: Steven Miller
//Class #: 11318
//PI Name: Anthony Stross
//Description: allows you to switch between the photoresistor and the analog input j3
//*****
//includes
#include <avr/interrupt.h>
#include <avr/io.h>
//variables
volatile int16_t result;
volatile int8_t bsel = 5;
volatile int8_t bscale = -6;
volatile uint8_t adca_ready;
volatile uint8_t light_true;
volatile uint8_t function_true;
int main(void)
{
    int8_t upperbyte = 0;
    int8_t lowerbyte = 0;
    //initialize ADC
    adc_init();
    //initialize tcc0
    tcc0_init();
    //initialize interrupts
    inter_init();
    //initialize usartd0
    usartd0_init();
    //start timer counter
    TCC0.CTRLA = TC_CLKSEL_DIV64_gc;
    while (1)
    {
        if(adca_ready)
        {
            upperbyte = (ADCA.CH0.RESH<<0);
            lowerbyte = (ADCA.CH0.RESL<<0);
            //output adca result to computer
            USARTD0.DATA = lowerbyte;
            while(!(USARTD0.STATUS & USART_DREIF_bm))
            {
                //do nothing
            }
            USARTD0.DATA = upperbyte;
            while(!(USARTD0.STATUS & USART_DREIF_bm))
            {
                //do nothing
            }
            //reset adca
            adca_ready = 0;
        }
    }
}
```

```
//NOTE THAT PHOTORESISTOR IS SET AS DEFAULT UPON BOOTUP
void adc_init(void)
{
    //set port a pin 1 and 6 as inputs
    PORTA.DIRCLR = (PIN1_bm|PIN6_bm);
    //set adca as 12 bit signed right adjusted
    ADCA.CTRLB = (ADC_CONMODE_bm|ADC_RESOLUTION_12BIT_gc);
    //set adca reference voltage to +2.5V
    ADCA.REFCTRL = (0|ADC_REFSEL_AREFB_gc);
    //set adca channel 0 to differential with gain x1
    ADCA.CH0.CTRL = (ADC_CH_INPUTMODE_DIFFWGAIN_gc);
    //lower ADCA sampling
    ADCA.PRESCALER = ADC_PRESCALER_DIV512_gc;
    //set adca muxpos to port A pin 1 and pin 6
    ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN1_gc|ADC_CH_MUXNEG_PIN6_gc);
    //enable interrupts on ADCA ch0
    ADCA.CH0.INTCTRL = (ADC_CH_INTMODE_COMPLETE_gc|ADC_CH_INTLVL_MED_gc);
    //enable conversion trigger on channel event 0
    ADCA.EVCTRL = (ADC_EVSEL_0123_gc|ADC_EVACT_CH0_gc);
    //ENABLE ADC
    ADCA.CTRLA = (ADC_ENABLE_bm);
}

void inter_init(void)
{
    //enable medium level interrupts in pmic
    PMIC.CTRL = PMIC_MEDLVLEN_bm;
    //enable global interrupts
    sei();
}

void tcc0_init(void)
{
    uint16_t period = 115;
    uint8_t offset = 0;
    //load tcc0 period register
    TCC0.PER = period + offset;
    TCC0.CNT = 0;
    //set tcc0 as event channel 0 source
    EVSYS.CH0MUX = EVSYS_CHMUX_TCC0_OVF_gc;
}
```



```
void usartd0_init(void)
{
    //initialize transmitter and reciever pins
    PORTD.OUTSET = PIN3_bm;
    PORTD.DIRSET = PIN3_bm;
    PORTD.DIRCLR = PIN2_bm;

    //set baud rate
    USARTD0.BAUDCTRLA = (uint8_t)bsel;
    USARTD0.BAUDCTRLB = (uint8_t)((bscale << 4)|(bsel >> 4));

    //set to 8 bit odd parity with 1 stop bit
    USARTD0.CTRLA = (USART_CMODE_ASYNCHRONOUS_gc | USART_PMODE_ODD_gc |
    USART_CHSIZE_8BIT_gc)&(~USART_SBMODE_bm);

    //ENABLE TRANSMITTER AND RECIEVER
    USARTD0.CTRLB = USART_RXEN_bm | USART_TXEN_bm;

    //enable interrupts
    USARTD0.CTRLA = USART_RXCINTLVL_MED_gc;
}

ISR(ADCA_CH0_vect)
{
    result = (ADCA.CH0.RESH<<8 | ADCA.CH0.RESL<<0);
    //set ADCA_READY flag
    adca_ready = 1;
}

ISR (USARTD0_RXC_vect)
{
    char C = USARTD0.DATA;
    //SWITCH TO FUNCTION GENERATOR
    if(C == 'F')
    {
        //switch ADCA inputs to function generator
        ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN5_gc|ADC_CH_MUXNEG_PIN4_gc);
        //since ADCA could be in the middle of a conversion, we need to flush the ADC channel
        ADCA.CTRLA |= (ADC_FLUSH_bm);
    }
    //SWITCH TO PHOTORESISTOR
    else if(C == 'L')
    {
        //switch ADCA inputs to function generator
        ADCA.CH0.MUXCTRL = (ADC_CH_MUXPOS_PIN1_gc|ADC_CH_MUXNEG_PIN6_gc);
        //since ADCA could be in the middle of a conversion, we need to flush the ADC channel
        ADCA.CTRLA |= (ADC_FLUSH_bm);
    }
}
```

APPENDIX

N/A