**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 volts**

**A graph with a red line

Description automatically generated  
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 VD = f(VA).**

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

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

**Therefore, VD = .012(VA)-1.06**

**PSEUDOCODE/FLOWCHARTS**

**SECTION 1**

**A diagram of a program

Description automatically generated  
Figure 2: Flowchart for “lab7\_1.C”**

**SECTION 2**

**A diagram of a flowchart

Description automatically generated**

**Figure 3: Flowchart for “lab7\_2.C”**

**SECTION 3**

**A diagram of a diagram

Description automatically generated**

**Figure 4: Flowchart for “lab7\_3.C”**

**SECTION 4**

**A diagram of a flowchart

Description automatically generated**

**Figure 5: Flowchart for “lab7\_4.C”**

**SECTION 5**

**A diagram of a flowchart

Description automatically generated**

**Figure 6: Flowchart for “lab7\_5.C”**

**A diagram of a computer program

Description automatically generated**

**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>

#include "usart.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 = (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

upperbyte = (ADCA.CH0.RESH<<8);

lowerbyte = (ADCA.CH0.RESL<<0);

data = (upperbyte|lowerbyte);

}

}

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|ADC\_CH\_GAIN1\_bm);

//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);

//load calibration register with factory values

ADCA.CALL = ADCA\_CALL;

ADCA.CALH = ADCA\_CALH;

//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 photoresistory 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|ADC\_CH\_GAIN1\_bm);

//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);

//load calibration register with factory values

ADCA.CALL = ADCA\_CALL;

ADCA.CALH = ADCA\_CALH;

//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>

//variables

volatile *int16\_t* result;

volatile *int8\_t* bsel = 5;

volatile *int8\_t* bscale = -6;

volatile *uint8\_t* adca\_ready;

int main(void)

{

float int1 = 0;

float int2 = 0;

float 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

/\*note that "result" isnt the actual voltage value

its the digital representation of the voltage value

I.E.: convert "result" into a voltage value then

transmit it\*/

//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)

{

USARTD0.DATA = '+';

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

}

//convert to voltage value

float result\_flt = result\*.0012;

//get first digit and transmit

int1 = (*uint8\_t*)(result\_flt);

USARTD0.DATA = (int1+48);

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//transmit decimal point

USARTD0.DATA = '.';

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//get second decimal digit and transmit

result2 =(10\*(result\_flt-int1));

int2 = (*uint8\_t*)(result2);

USARTD0.DATA = (int2+48);

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//get third decimal digit and transmit

result3 = (10\*(result2-int2));

int3 =(*uint8\_t*)(result3);

USARTD0.DATA = (int3+48);

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//output voltage symbol

USARTD0.DATA = 'V';

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//output carriage return

USARTD0.DATA = '\r';

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//output linefeed

USARTD0.DATA = 10;

while(!(USARTD0.STATUS & USART\_DREIF\_bm))

{

//do nothing

}

//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|ADC\_CH\_GAIN1\_bm);

//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);

//load calibration register with factory values

ADCA.CALL = ADCA\_CALL;

ADCA.CALH = ADCA\_CALH;

//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|ADC\_CH\_GAIN1\_bm);

//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);

//load calibration register with factory values

ADCA.CALL = ADCA\_CALL;

ADCA.CALH = ADCA\_CALH;

//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)|(00 >> 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|PIN5\_bm|PIN4\_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|ADC\_CH\_GAIN1\_bm);

//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);

//load calibration register with factory values

ADCA.CALL = ADCA\_CALL;

ADCA.CALH = ADCA\_CALH;

//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;

//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