Data Acquisition and Display: The Digital Inclination Angle Measurement System

Sakib Reza, rezas2, 400131994, L05

***Abstract*—Using the EsduinoXtreme HCS12GA240 microcontroller, a device capable of acquiring inclination angle measurement data between 0° and 90° is to be prototyped. From this device, a system must be created to process the data and transmit the data for display and recording. To do so, the AXDL337 breakout board is used. The process to do so was to quantify the analog signal, build the appropriate transducer, recognizing the precondition signal, data processing and controlling/communicating with the prototype. The creation and development of this device is highlighted below.**

I. INTRODUCTION & BACKGROUND

The creation of analog-to-digital conversion devices began as early as **1642** with **Blaise Pascal** inventing the mechanical calculator. The calculator worked in bases 6, 10, 12, and 20. Modern day analog-to-digital conversion is vital to everyday life, in a multitude of industries ranging from medicine to military.

To create a prototype capable of acquiring inclination angle measurement data, a process must be followed. An analog signal must first be found and processed by a transducer. The analog signal is received from the AXDL337 breakout board. From there the signal can optionally be placed in a signal-conditioning circuit, for preconditioning. From there analog-to-digital conversion occurs using one of many methods. With the ESDX, the conversion occurs with **successive approximation**. The prototype must take digital input/output in the form of a button push in addition to I/O to display the angle in two modes to LEDs. The data must be further processed, controlled and communicated via serial transmission to a PC application; in this case MATLAB.

From the first mechanical calculator there has been many significant jumps to ADC technology. Creating music today would be nothing without ADC technology. When sounds are recorded as analog signals, an analog-to-digital converter is needed to produce the pulse-code modulation data streams vital to create digital music files. Currently, this sampling rate used to record music can go up to 192 kHz, but 44.1 to 48 kHz is applied in radio/television applications.

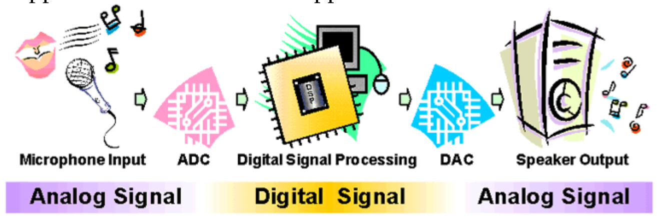


Figure 1: Graphic of the transition of signals in music

II. DESIGN METHODOLOGY

A. Final Pin Assignment Map

The prototype for the angle measurement device is created using 9 LED lights (7 red, 2 green), 9 150Ω resistors, 1 digital button, 1 AXDL337 breakout board, and 1 EsduinoXtreme HCS12GA240. On the ESDX, there are a total of 13 wires; 9 output wires, 2 input wires (one transmitting analog signals, one transmitting digital signals), 1 power source wire, set to 3.3V, and 1 ground wire.

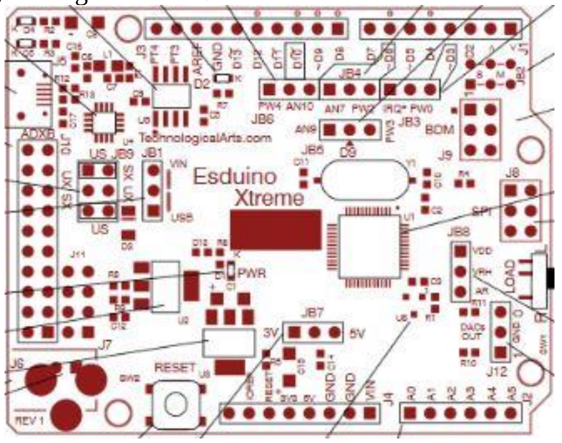


Figure 2: Schematic design for EsduinoXtreme

For the prototype, PAD pins 0-6, 8 & 11 were utilized as output to the LED lights, with AN7 being used to receive the transmission signal for the analog signal. PT0 was used to drive an interrupt based button toggle. The 3.3V source was used to power the accelerometer, and the ground was used to ground the switch, 9 LEDs, and the accelerometer.

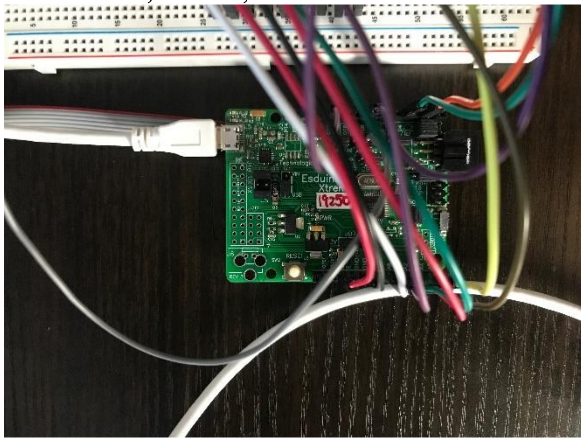


Figure 3: Photo of wires connected to ESDX

B. *Quantify Signal Properties*

The quantification of the signal properties was a tedious process; if the accelerometer adjusted on the breadboard between uses, the signal will be different based on

C. *Transducer*

The transducer used in the prototype is the ADXL337 Breakout Board from SparkFun. This piece of hardware is an accelerometer, used to measure the static acceleration of gravity for tilt-sensing applications. This device is a 3V, sensitive device with a 0.01 uF capacitor, with the maximum data collection rate of 500 Hz. This device is used in applications like gaming, image stabilization, disk drive protection, etc.

The accelerometer was used as the transducer in the system to receive an analog voltage representing the angle data.

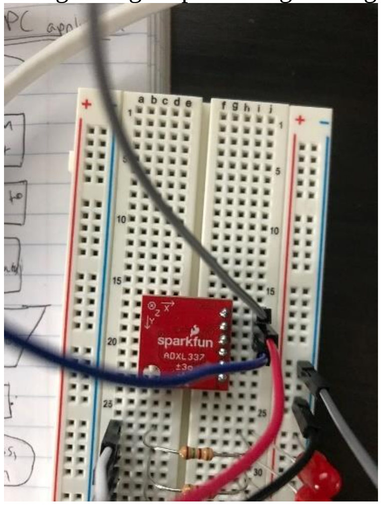


Figure 4: Picture of Transducer and Wiring

D. *Precondition/Amplification/Buffer*

For this system, there was no precondition added to the prototype. A precondition could be added but is unnecessary in this situation due to the analog signal being simple and easy to process.

E. *ADC*

The analog-to-digital conversion occurs after the transducer produces meaningful output in the form of an analog signal. The ADC onboard the ESDX creates digital representation from the analog signal thorough a process known as Successive Approximation, which employs a binary search algorithm to convert analog signals into digital representation.

The conversion occurs at AN7, due to the LSD of the student number being 4. The resolution used is 10-bit resolution, right justified.

F. *LED Display MODE 0 (BCD) & MODE 1 (Bar)*

The 9 LED's display the angle data from the ESDX in two forms; binary-coded decimal, and bar approximation. The outputs were driven from PAD 0-6, 8, and 11. These ports were chosen as GPIO because they were simple and easy to implement.

The switch caused an interrupt that changed the mode the prototype was operating in, flashing the onboard LED from PTJ on the ESDX indicating the mode has changed. The interrupt was implemented using PT0.

For BCD mode, the angle from the data was split into two separate digits. From there, if statements were used create the binary representation outputted said representation to the ESDX. Only 8 of the LEDs are used in this process.

For the bar representation, a simple if/else statement was used to display the angle, in groups of 10 from 5 to 85.

G. *Data Processing*

The processing of data from the ADC was a tedious process; if the accelerometer adjusted on the breadboard between uses, the signal will be different based on the adjustments. Thus, inclination angles and their corresponding DC voltages were recorded using Microsoft Excel and a level app on a phone. Once the data was plotted, measuring in increments of 5°, a line of best fit or multiple lines of best fit were found and calculated. The most consistent numbers were then applied into CodeWarrior into the C code.

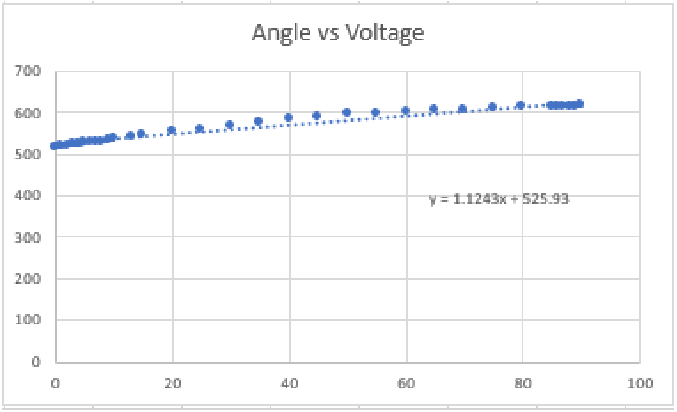


Figure 5: Example of sample data testing

H. *Control/Communicate (Algorithm Flowchart(s))*

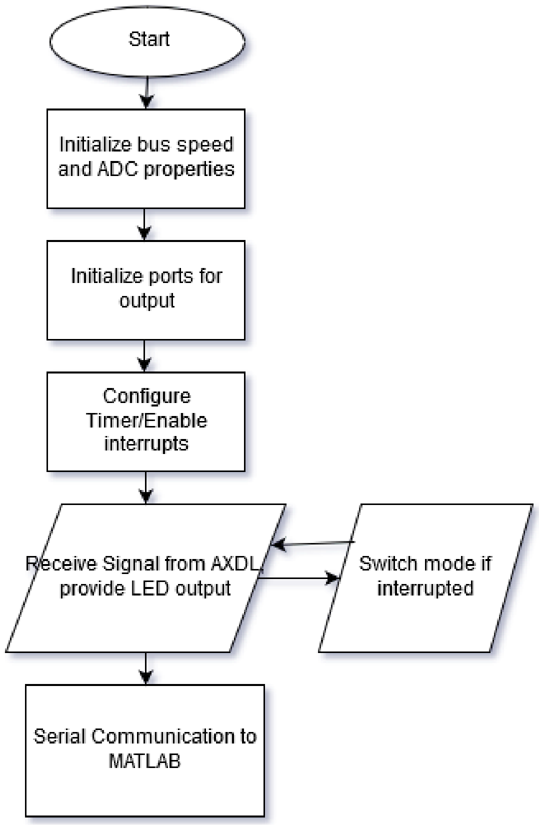

Start
Initialize bus speed and ADC properties
Initialize ports for output
Configure Timer/Enable interrupts
Receive Signal from AXDL, provide LED output
Switch mode if interrupted
Serial Communication to MATLAB

Figure 6: Flowchart for uC Code

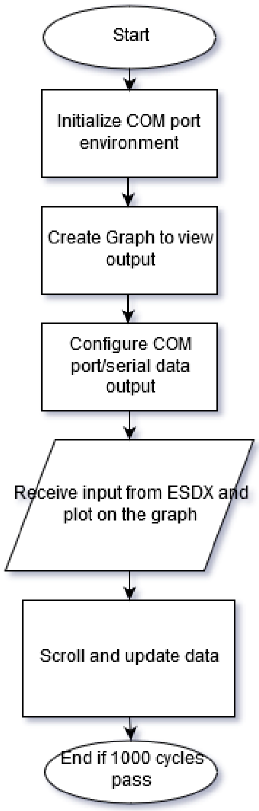

Start
Initialize COM port environment
Create Graph to view output
Configure COM port/serial data output
Receive input from ESDX and plot on the graph
Scroll and update data
End if 1000 cycles pass

Figure 7: Flowchart for PC application (MATLAB)

I. *Full System Block Diagram*

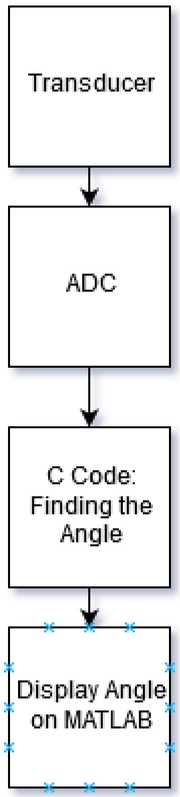


Figure 8: Full System Block Diagram

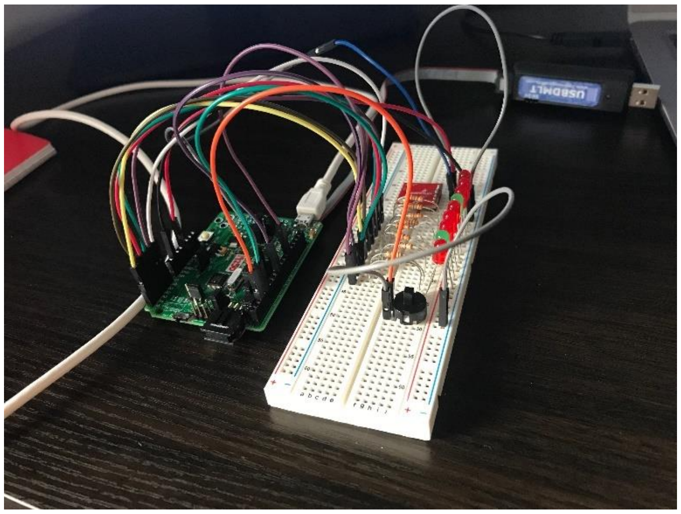


Figure 9: Full System Hardware Overview

J. *Full System Circuit Schematic*

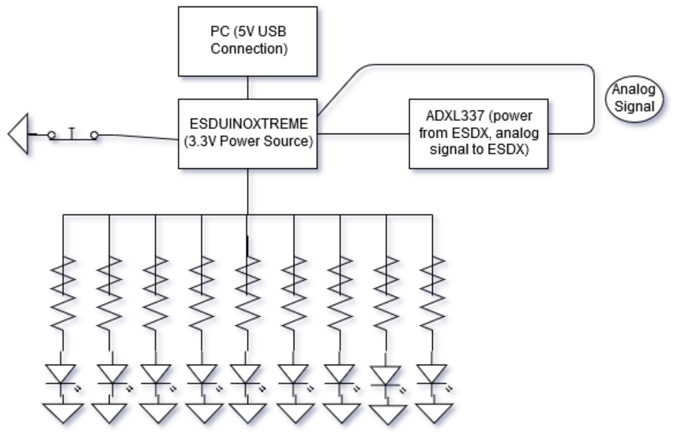

PC (5V USB Connection)
ESDUINOXTREME (3.3V Power Source)
ADXL337 (power from ESDX, analog signal to ESDX)
Analog Signal

Figure 10: Full System Circuit Schematic

III. RESULTS

After all the calculations, code, and setup, a functioning prototype was created, following the schematic designs in the previous section. An inclination angle measuring device working from 0° to 90°.

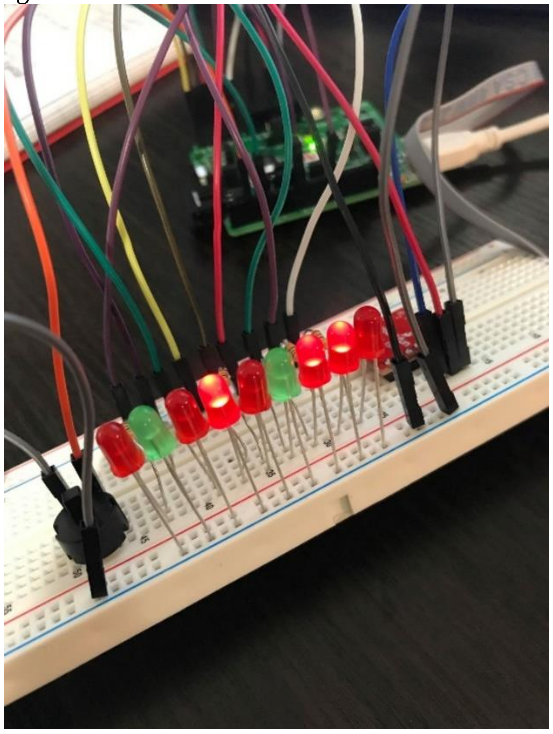


Figure 11: BCD mode

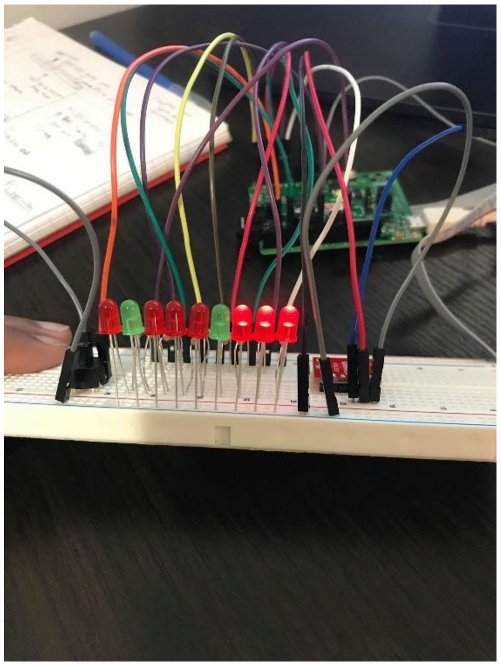
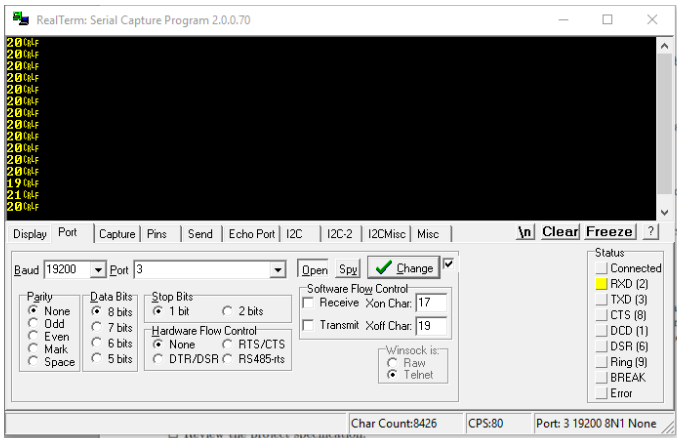


Figure 12: Bar Mode



RealTerm: Serial Capture Program 2.0.0.70

```
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
20CRLF
19CRLF
21CRLF
20CRLF
```

Baud 19200 Port 3

Parity: [✓ None] Odd Even Mark Space

Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits

Stop Bits: [✓ 1 bit] 2 bits

Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts

Software Flow Control: Receive Xon Char: 17 Transmit Xoff Char: 19

Winsock is: Raw [✓ Telnet]

Status: Connected, RXD (2), TXD (3), CTS (8), DCD (1), DSR (6), Ring (9), BREAK, Error

Char Count:8426 CPS:80 Port: 3 19200 8N1 None

Figure 13: Realterm Angle measurement numbers

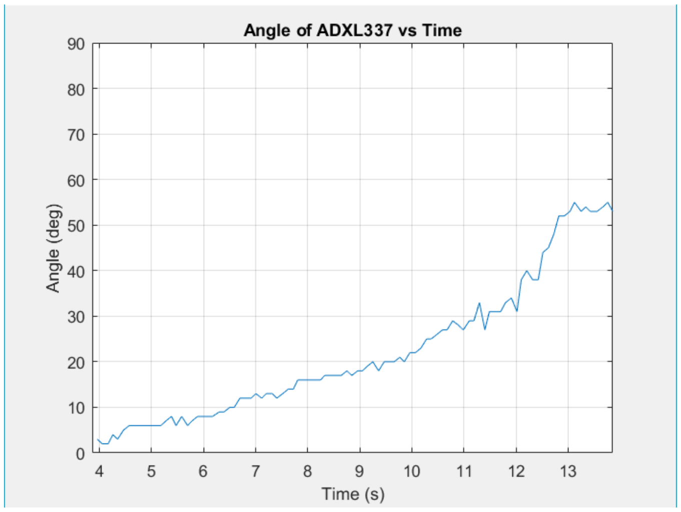


Figure 14: MATLAB graph for inclination angle measurements

IV. Discussion

There were many inconveniences and problems found during the development of the prototype. One problem was the ESDX's lack of floating point or trigonometric function capabilities. The lack of the floating point functionality was worked around by rounding to full numbers and minimizing the use of floating point numbers. To work around the lack of access to trigonometric functions, calculations had to be modified with experimental approximation.

The maximum quantization error is calculated with the following formula:

$$\textit{Max Quantized Error} = (V\mathit{max}-V\mathit{min})/2^{n+1}$$

With our maximum voltage being 3.3 V and our minimum voltage being 1.8 V, and our resolution being 10-bits, the maximum error is approximately 0.0007324.

The maximum serial communication rate can be calculated with the following formula:

$$\textit{Baud Divisor} = \textit{frequency}/(16*\textit{MaxBaudRate})$$

With the baud divisor being set to 1 and the frequency being 4 MHz, the maximum baud rate we achieve is 400000. The closest standard baud rate below this threshold is 115200. Due to limitations in MATLAB, and errors due to high baud rates, the rate used for the prototype is 19200.

The biggest limitation for speed in this system would be the Nyquist rate, because if information gets lost, the system will fail. For this system, the Nyquist rate is 500 Hz/2 which is 250 Hz. 500 Hz is the maximum sampling frequency, therefore the threshold is 250 Hz. If this threshold is exceeded loss of information known as aliasing occurs.

In general, analog input signals with sharp transitions are able to be accurately reproduced with limitations; the number of discrete data points offers an easy way to model the waves.

V. Conclusion

The development of this inclination angle prototype offers a very thorough and knowledgeable understanding of a product development with microcontrollers. From the calculations to output, there are many places to improve upon, but an accurate angle measurement device was created. This process can be easily translated to other analog signals that need to be digitally measured, recorded, and analyzed. An example of this perhaps a laser distance measuring device. Or perhaps a pressure sensing device, to be used in prosthesis applications. The possibilities are truly endless.

References

[1] Marguin, Jean (1994). Histoire des instruments et machines à calculer, trois siècles de mécanique pensante 1642-1942 (in French). Hermann. ISBN 978-2-7056-6166-3.

[2] U. Beis, "An Introduction to Delta Sigma Converters," An Introduction to Delta Sigma Converters, 19-Feb-2016. [Online]. Available: https://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html. [Accessed: 08-Apr-2019].

[3] ADC and DAC. [Online]. Available: http://macao.communications.museum/eng/Exhibition/secondfloor/moreinfo/ADConverter.html. [Accessed: 08-Apr-2019].

APPENDIX

```
//*********************************************************************
//*                       McMaster University                         *
//*               COMP ENG 2DP4: Microprocessor Systems               *
//*                          Lab Section 05                           *
//*                    Sakib Reza 400131994 rezas2                    *
//*********************************************************************
//*********************************************************************
//*                          Final Project                            *
//*                   DATA ACQUISITION  AND  DISPLAY:                 *
//*        THE  DIGITAL  INCLINATION  ANGLE  MEASUREMENT  SYSTEM      *
//*********************************************************************

#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include "SCI.h"
unsigned short val;
float degree;
int button = 0, fDig, sDig, high, low, extra;

// Prototypes
void setCLK(void); // set bus speed
void setADC(void); // set ADC = AN4
void delay1ms(unsigned int); // to test bus speed, 1ms delay
void OutCRLF(void); // for serial communication


void main(void) {

  setCLK();   // set bus speed to 4 MHz
  setADC();   // set ADC at AN7 with 10-bit resolution


  //Set Ports
  DDRJ = 0xFF;
  DDR1AD = 0b01111111;
  DDR0AD = 0b11111111;


  TSCR1 = 0x90;    // Timer System Control Register 1, 0x90 = 1001 0000
                    // TSCR1[7] = TEN:  Timer Enable (0-disable, 1-enable)
                    // TSCR1[6] = TSWAI:  Timer runs during WAI (0-enable, 1-disable)
                    // TSCR1[5] = TSFRZ:  Timer runs during WAI (0-enable, 1-disable)
                    // TSCR1[4] = TFFCA:  Timer Fast Flag Clear All (0-normal 1-
read/write clears interrupt flags)
                    // TSCR1[3] = PRT:  Precision Timer (0-legacy, 1-precision)
                    // TSCR1[2:0] not used

  TSCR2 = 0x01;    // Timer System Control Register 2
                    // TSCR2[7] = TOI: Timer Overflow Interrupt Enable (0-inhibited,
1-hardware irq when TOF=1)
                    // TSCR2[6:3] not used
                    // TSCR2[2:0] = Timer Prescaler Select: See Table22-12 of MC9S12G
Family Reference Manual r1.25 (set for bus/1)


  TIOS = 0xFE;     // Timer Input Capture or Output capture      , 0xFE = 1111 1110 so
pin 0 is input-capture, all others are output-compare
                   // set TIC[0] and input (similar to DDR)
  PERT = 0x01;     // Enable Pull-Up resistor on TIC[0]

  TCTL3 = 0x00;    // TCTL3 & TCTL4 configure which edge(s) to capture   (TCTL3 is for
pins 7-4, neflect)
  TCTL4 = 0x02;    // Configured for falling edge on TIC[0]              (TCTL4 pins
3-0, pin 2 is set 1 0 = falling edges captured!)


  TIE = 0x01;      //Timer Interrupt Enable

  // initialize serial communication
  SCI_Init(19200);
  //SCI_OutString("Sakib Reza; 400131994");
  //OutCRLF();

  // enable interrupts
    EnableInterrupts;


  for(;;) {

    while (button == 0){
      val = ATDDR0;

      //refer to excel spreadsheet for these functions
      if (val <= 507.73){
        degree = 0;

      } else if (val <= 535){
        degree = (val-507.73)/1.8981;


      }else if (val <= 561){
        degree = (val-507.2)/1.7795;


      } else if (val <= 581){
        degree = (val-522.67)/1.3;


      } else if (val <= 600) {
        degree = (val-528)/1.2;


      } else if (val <= 611) {
        degree = (val-559)/0.7;


      } else if (val <= 618) {
        degree = (val - 554.81)/0.696;


      } else {
        degree = 90;

      }


      fDig = degree / 10;
      sDig = degree - (fDig * 10);
```

```
//first set of LEDs
if(fDig == 0){
  high = 0b00000000;
  extra = 0b00000000;

}

if(fDig == 1){
  high = 0b00010000;
  extra = 0b00000000;
}

if(fDig == 2){
  high = 0b00100000;
  extra = 0b00000000;

}

if(fDig == 3){
  high = 0b00110000;
  extra = 0b00000000;

}

if(fDig == 4){
  high = 0b01000000;
  extra = 0b00000000;

}

if(fDig == 5){
  high = 0b01010000;
  extra = 0b00000000;

}

if(fDig == 6){
  high = 0b01100000;
  extra = 0b00000000;

}

if(fDig == 7){
  high = 0b01110000;
  extra = 0b00000000;

}

if(fDig == 8){
  high = 0b00000000;
  extra = 0b00000001;

}

if(fDig == 9){
  high = 0b00010000;
  extra = 0b00000001;
}
//second set of LEDS
if(sDig == 0){
  low = 0b00000000;
```

```
      }

      if(sDig == 1){
        low = 0b00000001;
      }

      if(sDig == 2){
        low = 0b00000010;

      }

      if(sDig == 3){
        low = 0b00000011;

      }

      if(sDig == 4){
        low = 0b00000100;

      }

      if(sDig == 5){
        low = 0b00000101;

      }

      if(sDig == 6){
        low = 0b00000110;

      }

      if(sDig == 7){
        low = 0b00000111;

      }

      if(sDig == 8){
        low = 0b00001000;

      }

      if(sDig == 9){
        low = 0b00001001;
      }

      PT1AD = high|low;
      PT0AD = extra;

      //SCI_OutUDec(val);
      //OutCRLF();

      SCI_OutUDec(degree);
      OutCRLF();
      delay1ms(100);
    }


    while (button == 1){
      val = ATDDR0;
```

```
// refer to excel spreadsheet for these functions
if (val <= 506){
  degree = 0;

} else if (val <= 535){
  degree = (val-507.73)/1.8981;


}else if (val <= 561){
  degree = (val-507.2)/1.7795;


} else if (val <= 581){
  degree = (val-522.67)/1.3;


} else if (val <= 600) {
  degree = (val-528)/1.2;


} else if (val <= 611) {
  degree = (val-559)/0.7;


} else if (val <= 620) {
  degree = (val - 556.81)/0.696;


} else {
  degree = 90;

}


if (degree <= 5){
    PT1AD = 0b00000000;
    PT0AD = 0b00000000;

} else if (degree <= 15){
    PT1AD = 0b00000001;
    PT0AD = 0b00000000;

} else if (degree <= 25){
    PT1AD = 0b00000011;
    PT0AD = 0b00000000;

} else if (degree <= 35){
    PT1AD = 0b00000111;
    PT0AD = 0b00000000;

} else if (degree <= 45){
    PT1AD = 0b00001111;
    PT0AD = 0b00000000;

} else if(degree <= 55){
    PT1AD = 0b00011111;
    PT0AD = 0b00000000;

} else if(degree <= 65){
    PT1AD = 0b00111111;
    PT0AD = 0b00000000;
```

```
      } else if(degree <= 75){
          PT1AD = 0b01111111;
          PT0AD = 0b00000000;

      } else if(degree <= 85){
          PT1AD = 0b01111111;
          PT0AD = 0b00000001;

      } else if(degree <= 90){
          PT1AD = 0b01111111;
          PT0AD = 0b00001001  ;
      }

      //SCI_OutUDec(val);
      //OutCRLF();

      SCI_OutUDec(degree);
      OutCRLF();
      delay1ms(100);
    }

  }
}


interrupt  VectorNumber_Vtimch0 void ISR_Vtimch0(void) {    // interrupt for PT0 built
into CodeWarrior
  unsigned int temp; // used to enable FastFlagClearing

  if (button == 0){ // changing between modes & flashing onboard LED to know it worked
    button = 1;
    PTJ = 0x01;
  } else {
    button = 0;
    PTJ = 0x00;
  }

    //SCI_OutString("The new mode is:");
    //SCI_OutUDec (button);
    //OutCRLF();

  temp = TC0;       //Allow another TIC interrupt
}

// functions

void setCLK(void) {  // Bus Speed = 4 MHz
  CPMUPROT = 0x26; // protection of clock config is enabled
  CPMUCLKS = 0x80; // choose PLL = 1 MHz
  CPMUOSC = 0x00;
  CPMUSYNR = 0x0F; // VCOFRQ = 00, SYNDIN = 15 => VCLOCK = 2*1MHz*(1+SYNDIV) = 32MHz;
  CPMUFLG = 0x00; // LOCK = 0: PLLCLK = 32MHz/4 = 8MHz & BUSCLK = PLLCLK/2 = 4MHz

}

void setADC(void) {     // ADC at AN7 with 10-BIT resolution
    ATDCTL1 = 0x2F; // setting to 10-bit resolution
    ATDCTL3 = 0x88; // right justified, one sample per sequence
    ATDCTL4 = 0x01; // prescaler = 1; ATD clock = 4MHz / (2 * (2 + 1)) == 1MHz
    ATDCTL5 = 0x27; // conversion occurs at AN7
```

```
}

void delay1ms(unsigned int k) {   // function modified from Lab Code to set a k ms
delay at 4 Mhz bus speed
  int ix;
  TSCR1 = 0x90; // enable timer and fast timer flag clear
  TSCR2 = 0x00; // disabling timer interrupt
  TIOS |= 0x02;
  TC1 = TCNT + 4000; // (1/4MHz)*4000 = 1ms

  for(ix=0;ix<k;ix++) {
    while(!(TFLG1_C1F));
     TC1 += 4000;
  }
  TIOS &= ~0x01;
}

void OutCRLF(void){   // newline function for serial communication
  SCI_OutChar(CR);
  SCI_OutChar(LF);
  PTJ ^= 0x20;
}
```

```
%COMPENG 2DP4 Final Project: Serial Communication
%Sakib Reza
%rezas2
%400131994

 if ~isempty(instrfind) %making sure MATLAB can find the right port
     fclose(instrfind);
      delete(instrfind);
 end

plotTitle = 'Angle of ADXL337 vs Time';  %creating the graph to display the data
xLabel = 'Time (s)';
yLabel = 'Angle (deg)';
plotGrid = 'on';
min = 0;
max = 90;
timeAxis = 10;
delay = 0.01;

time = 0; %dynamic graphing variables, c1 & c2 used as time intervals
angle = 0;
c1 = 0;
c2 = 0;

plotGraph = plot(time, angle); %time to run it
title(plotTitle);
xlabel(xLabel);
ylabel(yLabel);
axis([0 10 min max]);
grid(plotGrid);


s = serial('COM3'); %setting up the serial communication, with the right port/baud
rate
set(s,'BaudRate',19200);
fopen(s);
```

```
tic

while ishandle(plotGraph)   %actual function taking the serial communcation and 
outputting data to graph
    c1 = c1 + 1;
    dat = fscanf(s);
    c2 = c2 + 1;
    time(c2) = toc;
    angle(c2) = (str2double(dat(1:2)));

        if(timeAxis > 0)
        set(plotGraph,'XData',time(time > time(c2)-timeAxis),'YData',angle(time > 
time(c2)-timeAxis));
        axis([time(c2)-timeAxis time(c2) min max]);
        else
        set(plotGraph,'XData',time,'YData',angle);
        axis([0 time(c2) min max]);
        end

        pause(delay);

    if(c1 == 1000)
        clear angle;
        c1 = 0;
    end
end

fclose(s); %Turning off serial communcation and clearing variables
clear c2 dat delay max min plotGraph plotGrid plotTitle s timeAxis serialPort xLabel 
yLabel;
delete(s);
clear s
```