**Project 3:  
Multimeter Development**

**Cal Poly CPE 329-01**

**Spring 2017**

short line

Brendan Baronia

Melinda Ong

Professor Gerfen, Jeffrey B.

**May 19, 2017**

# Project Demonstration:

<https://youtu.be/b9EG54oi5EA>

# Purpose:

# The purpose of this system is to design and implement an MSP432 board to function as a digital multimeter in order to:

1. Understand how to develop UART and ADC systems.
2. Understand how to utilize the VT100 standard by creating a readable interface between the COM port and a suitable terminal on the computer.
3. Understand how to implement two timers to effectively and efficiently measure frequency.

# System Requirements:

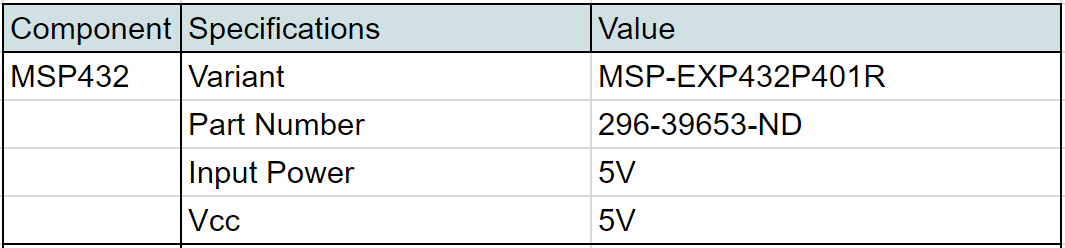
**Functional Requirements:**

1. Shall function as a DMM and measure voltage.   
   1. Voltage measurements shall be limited to 0 to 3.3 volts.
   2. Voltage measurements shall be limited to 0 to 1000 Hz.
   3. Voltage measurements shall be accurate to +/- 1 mv for AC and DC. 1.4 The DMM shall have a DC setting.
      1. DC measurements shall average over a 1 ms time period.
      2. DC measurements of a sinusoidal waveform should be equivalent to the DC offset of the sinusoid.
   4. The DMM shall have a AC setting.
      1. AC measurements shall be true-RMS.
      2. AC measurements shall display the various components.
         1. AC voltage measurements shall give the TrueRMS (includes DC offset).
         2. AC voltage measurements shall give the CalcRMS (TrueRMS – DC offset).
         3. AC voltage measurements shall give the peak-to-peak value.
      3. AC measurements shall work for various waveforms
         1. Sine waves shall be measurable
         2. Triangular waves shall be measurable
         3. Square waves shall be measurable
         4. Other periodic waveforms shall be measurable
      4. AC measurements shall work for waveforms of various amplitudes and offsets
         1. The maximum voltage that shall be measured is 3V
         2. The minimum voltage that shall be measured is 0V
         3. The minimum peak-to-peak voltage that shall be measured is 0.5V
         4. Offset values of up to 2.75V shall be measurable
2. Shall function as a DMM and measure frequency.   
   1. Frequency measurements shall be limited to 1 to 1000 HZ.
   2. Frequency measurements shall be accurate to within 1 Hz.
   3. Frequency measurements shall work for various waveforms.
      1. Sine waves shall be measurable.
      2. Triangular waves shall be measurable.
      3. Square waves shall be measurable.
      4. Other periodic waveforms shall be measurable.
3. Shall function as a DMM and have a terminal-based interface.   
   1. The terminal shall operate at a frequency greater than 9600 baud.
   2. The terminal shall utilize the VT100 protocol.
      1. The terminal shall display all fields in non-changing locations
   3. The terminal shall display AC voltages as described above.
   4. The terminal shall display DC voltages as described above.
   5. The terminal shall display frequency as described above.
   6. The terminal shall organize the presentation of information.
      1. AC, DC, and frequency shall be simple to read.
      2. The display may use horizontal and vertical lines (borders) to organize the presentation of information.
   7. The terminal shall use bar-graphs for voltages being measured.
      1. The terminal shall have a bar-graph for “Calc-RMS”.
      2. The terminal shall have a bar-graph for DC voltages.
      3. The bar graphs shall be shall have delineators, e.g. a scale, indicating the equivalent voltage being measured.
      4. The bar graphs shall be a single line of pixels, characters, etc.
      5. The bar graphs shall have length that is proportional to the voltage being measured.
      6. The bar graphs shall respond in real-time to changes in AC or DC voltage

**Development Constraints:**

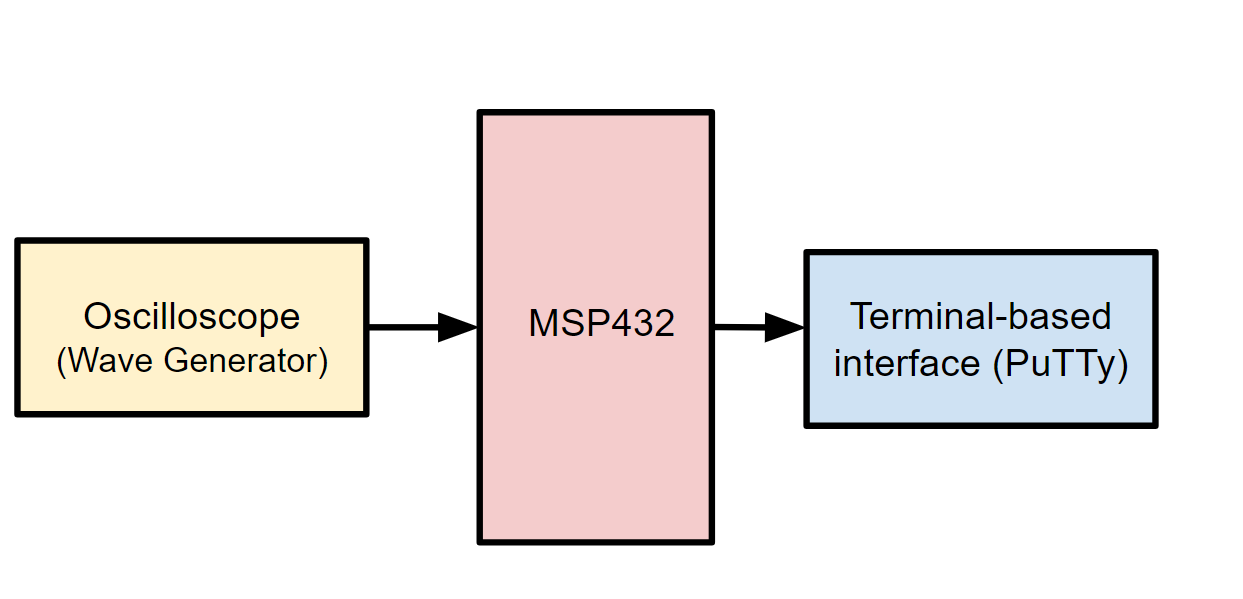
1. Shall function as a DMM and utilize #defines from msp432p401r.h

# System Specification:

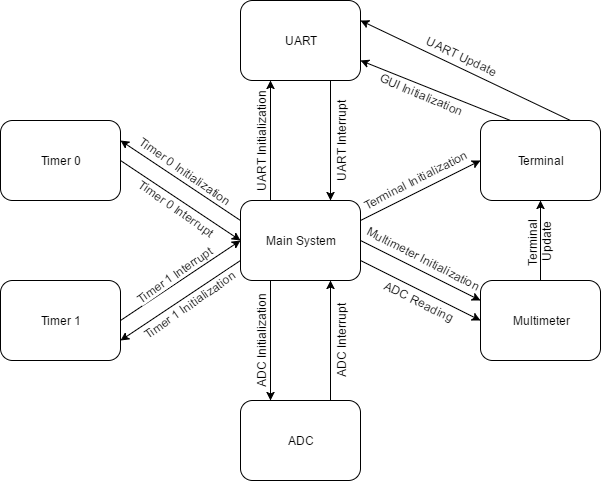


**Table 1: System Specifications**

System Architecture:

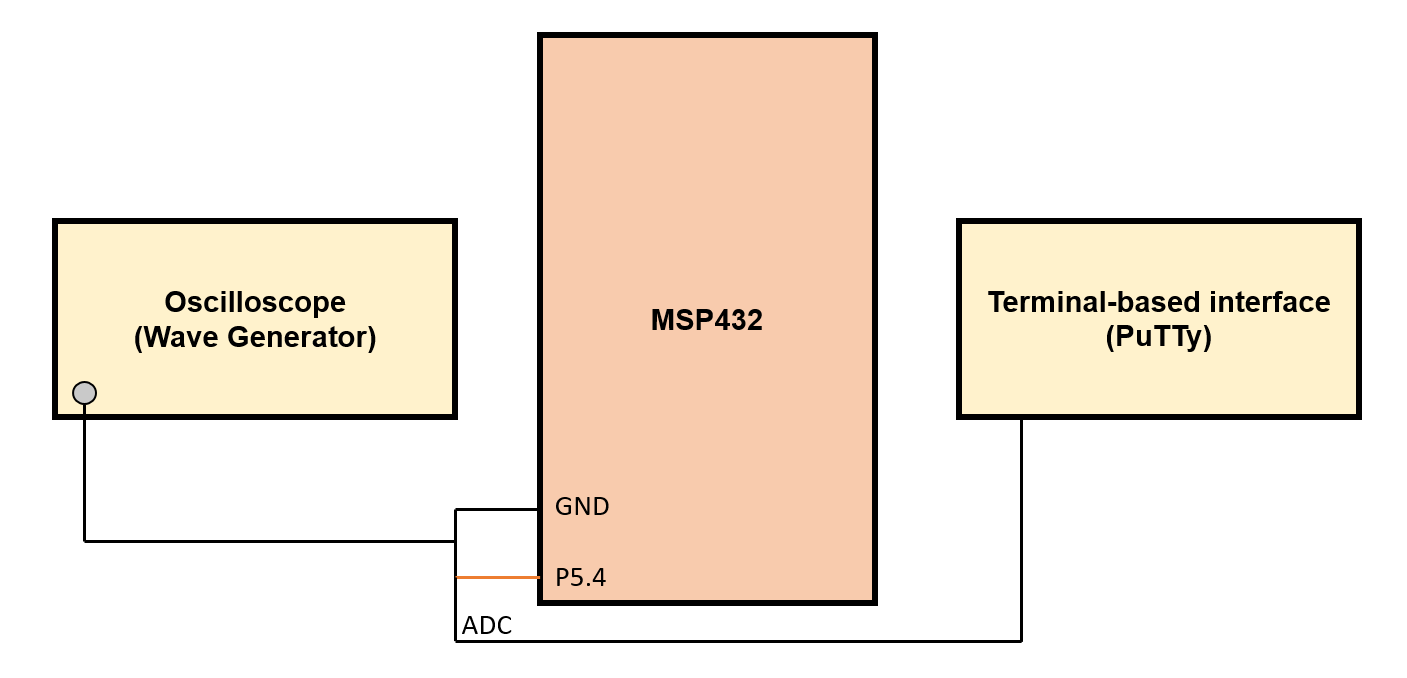


**Figure 1: Overall system block diagram**

****

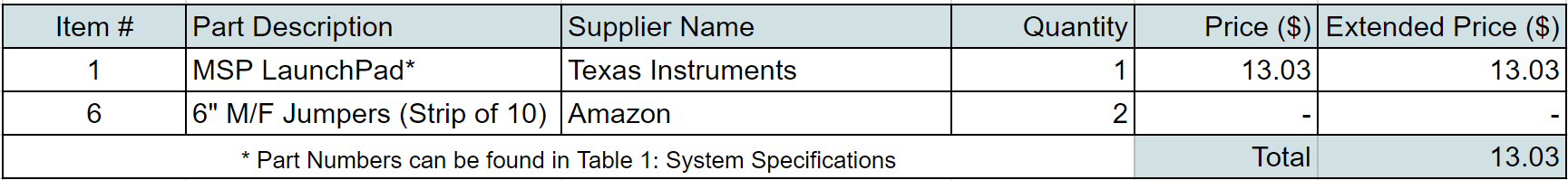
**Figure 2: Top-level Software flowchart**

# Component Design;



**Figure 3: Schematic Diagram of the MSP432, Oscilloscope, and Terminal Output**

# Bill of Materials:



**Table 2: Bill of Materials**

System Integration:

***a.) Development Process***

The development process followed a straightforward object oriented design. Each object of the system was developed then integrated through the main system, with proper testing as each phase was completed.

The first systems developed were the UART and ADC systems. Since much of the project relied on these, they were quickly developed using software delays to read and write. Since items were only written out via UART, the system used a short delay when dropping each character on the wire to ensure transmission was complete. Similarly, items were only read in from the ADC system so the system used a short delay before reading another value.

From here, the GUI system was developed. By using the VT100 standard, a simple GUI was added over a COM port to a suitable terminal on the computer. This system was dependent on a functional UART system, so once the GUI was designed, it was integrated into the UART system to create a cohesive unit.

The next step was to read data from the ADC. DC values require few calculations, so they were first implemented and integrated into the GUI. Once the DC values were steady, AC values were calculated from RMS calculations after collecting multiple data points.

With both the DC and AC components functioning, the next step involved reading frequencies. Unfortunately, due to the software delays previously implemented, large sections of code were rewritten to allow to accurate frequency measurement. Using multiple interrupts, frequency could be accurately measured while other parts of the system worked to produce other results.

***b.) Bugs and Resolutions***

The primary bug encountered while designing this system was discovered when frequency was implemented. Because of the software delays previously implemented for data transmission, frequency was unstable and inaccurate. Using a combination of UART and ADC interrupts, some of the inaccuracy was removed, but the system was still unstable. By implementing a timer to process DC and AC measurements, the inaccuracy was further reduced, but the complex calculations required caused major fluctuations in frequency measurements. Finally, a second timer was implemented with highest priority to measure frequency alone, removing all instability and creating a very accurate frequency measurement.

# Conclusion:

The purpose of this project was to to design and implement an MSP432 board to function as a digital multimeter. The multimeter utilized a terminal-based interface and displayed both DC and AC field requirements in presentable and non-changing locations. The DC measurements included a voltage average over a 1ms time period. The AC measurements included the values for true-RMS (including DC offset), calc-RMS (not including DC offset), peak-to-peak voltage, and frequency. The system was required to accurately work to +/- 1mV for various periodic waveforms including, but not limited to, sinusoidal, triangular, and square.

This system could be improved with more precision over shorter frequencies. Our system measurements were not accurate with waves less than 4 Hz. This could be improved by taking samples to account for longer periods, but in turn, would increase the measurement generation time making the system less efficient.

Finally, packaging could be improved to mobilize the system. An LCD could be integrated for portability.

# Appendices:

***a.) C Code***

***i.) main.c***

#include "msp.h"  
#include "uart.h"  
#include "terminal.h"  
#include "adc.h"  
#include "multimeter.h"  
  
#define CONVERSION 13 /\*Conversion factor for ADC measurements\*/  
#define CYCLETIMER 48000 /\*Timer for voltage readings\*/  
#define CYCLETIMER0 9600 //Adjust this for frequency  
  
  
/\*keeps track of system time in ms & last read voltage\*/  
static short val;  
static int time;  
  
int main() {  
 /\*Stop watchdog timer\*/  
 WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;  
  
 P1->DIR |= BIT0;  
 P1->OUT &= ~BIT0;  
 /\*Disable interrupts\*/  
 \_\_disable\_irq();  
 /\* Transition to VCORE Level 1: AM0\_LDO --> AM1\_LDO \*/  
 while ((PCM->CTL1 & PCM\_CTL1\_PMR\_BUSY));  
 PCM->CTL0 = PCM\_CTL0\_KEY\_VAL | PCM\_CTL0\_AMR\_1;  
 while ((PCM->CTL1 & PCM\_CTL1\_PMR\_BUSY));  
  
 /\* Configure Flash wait-state to 1 for both banks 0 & 1 \*/  
 FLCTL->BANK0\_RDCTL = (FLCTL->BANK0\_RDCTL &  
 ~(FLCTL\_BANK0\_RDCTL\_WAIT\_MASK)) | FLCTL\_BANK0\_RDCTL\_WAIT\_1;  
 FLCTL->BANK1\_RDCTL = (FLCTL->BANK0\_RDCTL &  
 ~(FLCTL\_BANK1\_RDCTL\_WAIT\_MASK)) | FLCTL\_BANK1\_RDCTL\_WAIT\_1;  
  
 /\*Set CLK to 48 MHz\*/  
 CS->KEY = CS\_KEY\_VAL;  
 CS->CTL0 = 0;  
 CS->CTL0 = CS\_CTL0\_DCORSEL\_5;  
 CS->CTL1 = CS->CTL1 & ~(CS\_CTL1\_SELM\_MASK | CS\_CTL1\_DIVM\_MASK) |  
 CS\_CTL1\_SELM\_3;  
 CS->KEY = 0;  
  
 UART0\_init(); /\*start up uart transmission\*/  
 terminal\_init(); /\*set up terminal frame\*/  
 ADC\_init(); /\*start up ADC\*/  
 multimeter\_init(); /\*start up multimeter system\*/  
  
 TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled  
  
 TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_CCIE; // TACCR1 interrupt enabled  
 TIMER\_A0->CCR[0] = CYCLETIMER0; //Set CCR0 for on cycle  
 TIMER\_A0->CCR[1] = CYCLETIMER; //Set CCR0 for on cycle  
  
  
 TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode  
 TIMER\_A\_CTL\_MC\_\_CONTINUOUS;  
  
 time = 0;  
  
 NVIC\_EnableIRQ(EUSCIA0\_IRQn); /\*enable uart tx interrupt\*/  
 NVIC\_SetPriority(EUSCIA0\_IRQn, 6); /\*set last priority for uart\*/  
  
 NVIC\_SetPriority(ADC14\_IRQn, 5); /\*higher priority for adc\*/  
 NVIC\_EnableIRQ(ADC14\_IRQn); /\*enable ADC interrupt\*/  
  
 NVIC\_EnableIRQ(TA0\_0\_IRQn);  
 NVIC\_SetPriority(TA0\_0\_IRQn, 1);  
 NVIC\_EnableIRQ(TA0\_N\_IRQn);  
 NVIC\_SetPriority(TA0\_N\_IRQn, 7);  
  
  
 ADC14->CTL0 |= ADC14\_CTL0\_ENC | ADC14\_CTL0\_SC; /\*start reading adc\*/  
  
 /\*enable interrupts\*/  
 \_\_enable\_irq();  
  
 /\*endless loop\*/  
 while(1) {  
 }  
}  
  
void TA0\_0\_IRQHandler() {  
 TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;  
 TIMER\_A0 ->CCR[0] += CYCLETIMER0;  
 time += 1;  
 readFrequency(val, time);  
}  
  
/\*Timer interrupt handler\*/  
void TA0\_N\_IRQHandler() {  
 if (TIMER\_A0->CCTL[1] & TIMER\_A\_CCTLN\_CCIFG) {  
 readVol(val);  
 TIMER\_A0->CCR[1] += CYCLETIMER;  
 TIMER\_A0->CCR[1] &= ~TIMER\_A\_CCTLN\_CCIFG;  
 }  
}  
  
/\*Uart interrupt handler\*/  
void EUSCIA0\_IRQHandler() {  
 /\*push next value over UART\*/  
 printVals();  
}  
  
/\*ADC interrupt handler\*/  
void ADC14\_IRQHandler() {  
 /\*read value from ADC\*/  
 val = ADC14->MEM[0] / CONVERSION;  
  
 /\*start next ADC processing\*/  
 ADC14->CTL0 |= ADC14\_CTL0\_ENC | ADC14\_CTL0\_SC;  
}

***ii.) multimeter.c***

#include "multimeter.h"  
#include "terminal.h"  
#include "msp.h"  
#include <math.h>  
  
#define MAXMEASUREMENTS 30000 /\*adjust number of measurements taken\*/  
#define FREQUENCYMULTIPLIER 5000 /\*adjust for frequency error\*/  
#define MAXFREQUENCIES 10  
  
/\*Private functions\*/  
void readPeak();  
void readVoltage();  
  
/\*Private variables\*/  
static short values[MAXMEASUREMENTS];  
static int location;  
static short myMid;  
static short lastReading;  
static int lastTime;  
static short myMin;  
static short myMax;  
  
/\*Multimeter setup\*/  
void multimeter\_init() {  
 int counter;  
  
 lastTime = 0;  
 lastReading = 4000;  
 location = 0;  
 for (counter = 0; counter < MAXMEASUREMENTS; counter++)  
 values[counter] = 0;  
}  
  
/\*Reads a value and time for multimeter\*/  
void readVol(short val) {  
 /\*Add value to array of past values\*/  
 values[location++] = val;  
  
 /\*Avoid array overflow\*/  
 if (location >= MAXMEASUREMENTS) {  
 location = 0;  
  
 /\*Perform calculations on values\*/  
 readVoltage();  
 readPeak();  
  
 }  
}  
  
  
/\*Finds the peak to peak of the array\*/  
void readPeak() {  
 updatePeak(myMax - myMin);  
}  
  
/\*Finds the RMS values of the array\*/  
void readVoltage() {  
 unsigned long long sum = 0;  
 int counter;  
 myMin = 4000;  
 myMax = 0;  
 unsigned long long DCsum = 0;  
 /\*Sum square of all values in array\*/  
 for (counter = 0; counter < MAXMEASUREMENTS; counter++)  
 {  
 sum = sum + (values[counter]\*values[counter]);  
 DCsum += values[counter];  
 /\*Check for minimum and maximum values\*/  
 if (values[counter] < myMin)  
 myMin = values[counter];  
 if (values[counter] > myMax)  
 myMax = values[counter];  
 }  
  
 myMid = (myMin + myMax) / 2;  
 /\*calculate rms\*/  
 sum = sum / counter;  
 unsigned long long temp = sqrt(sum);  
  
 /\*update values in terminal\*/  
 updateDC(DCsum / counter);  
  
 updateOffset(DCsum / counter);  
 updateTrueRMS(temp);  
 updateCalcRMS(temp - (DCsum / counter));  
}  
  
/\*Finds the frequency of the waveform\*/  
void readFrequency(short val, int time) {  
 if (lastReading < myMid && val > myMid) {  
 int newTime = time - lastTime;  
 if (newTime < 0)  
 newTime += 65384;  
 updateFrequency(FREQUENCYMULTIPLIER / newTime);  
 lastTime = time;  
 }  
 lastReading = val;  
}

***iii.) terminal.c***

#include "uart.h"  
#include "terminal.h"  
#include "format.h"  
  
/\*Private variable\*/  
static int location;  
  
/\*print terminal frame\*/  
void terminal\_init() {  
 location = 0;  
  
 /\*print until end of frame\*/  
 while (frame[location] != '\0') {  
 uartPrint(frame[location++]);  
 }  
 location = 0;  
}  
  
/\*Update offset for next terminal push\*/  
void updateOffset(int offset) {  
 /\*go to offset position in update\*/  
 int counter = OFFSET;  
  
 /\*set values in update\*/  
 update[counter] = offset / 100 + '0';  
 counter += 2;  
 update[counter++] = (offset / 10) % 10 + '0';  
 update[counter] = (offset / 1) % 10 + '0';  
}  
  
/\*Update peak to peak for next terminal push\*/  
void updatePeak(int peakToPeak) {  
 /\*go to peak to peak position in update\*/  
 int counter = PEAK;  
  
 /\*set values in update\*/  
 update[counter] = peakToPeak / 100 + '0';  
 counter += 2;  
 update[counter++] = (peakToPeak / 10) % 10 + '0';  
 update[counter] = (peakToPeak / 1) % 10 + '0';  
}  
  
/\*update True RMS for next terminal push\*/  
void updateTrueRMS(int trueRMS) {  
 /\*go to True RMS location in update\*/  
 int counter = TRUERMS;  
  
 /\*set values in update\*/  
 update[counter] = trueRMS / 100 + '0';  
 counter += 2;  
 update[counter++] = (trueRMS / 10) % 10 + '0';  
 update[counter] = (trueRMS / 1) % 10 + '0';  
}  
  
/\*Update Calc RMS for next terminal push\*/  
void updateCalcRMS(int calcRMS) {  
 /\*go to Calc RMS location in update\*/  
 int counter = CALCRMS;  
  
 /\*set values in update\*/  
 update[counter] = calcRMS / 100 + '0';  
 counter += 2;  
 update[counter++] = (calcRMS / 10) % 10 + '0';  
 update[counter] = (calcRMS / 1) % 10 + '0';  
  
 /\*print bar graph in update\*/  
 int numPoints = calcRMS / 8.3;  
 for (counter = RMSBAR; counter < RMSBAR + numPoints; counter++) {  
 update[counter] = '\*';  
 }  
 /\*clear any stars from last update\*/  
 for ( ; counter < RMSBAR + 53; counter++) {  
 update[counter] = ' ';  
 }  
}  
  
/\*Update DC for next terminal push\*/  
void updateDC(int dc) {  
 /\*go to dc position in update\*/  
 int counter = DC;  
  
 /\*set values in update\*/  
 update[counter] = dc / 100 + '0';  
 counter += 2;  
 update[counter++] = (dc / 10) % 10 + '0';  
 update[counter] = (dc / 1) % 10 + '0';  
  
 /\*print bar graph in update\*/  
 int numPoints = dc / 8.3;  
 for (counter = DCBAR; counter < DCBAR + numPoints; counter++) {  
 update[counter] = '\*';  
 }  
 /\*clear any stars from last update\*/  
 for ( ; counter < DCBAR + 53; counter++) {  
 update[counter] = ' ';  
 }  
}  
  
/\*update frequency for next terminal push\*/  
void updateFrequency(int frequency) {  
 /\*go to frequency position in update\*/  
 int counter = FREQUENCY;  
  
 /\*set values in update\*/  
 update[counter++] = (frequency / 1000) % 10 + '0';  
 update[counter++] = (frequency / 100) % 10 + '0';  
 update[counter++] = (frequency / 10) % 10 + '0';  
 update[counter] = frequency % 10 + '0';  
}  
  
/\*push update to terminal\*/  
void printVals() {  
 /\*go to beginning of update if end reached\*/  
 if (update[location] == '\0')  
 location = 0;  
  
 /\*send next char in update to UART\*/  
 uartPrint(update[location++]);  
}

***iv.) uart.c***

#include "uart.h"  
#include "msp.h"  
  
/\*UART setup\*/  
void UART0\_init() {  
 /\*Configure UART pins\*/  
 P1->SEL0 |= BIT2 | BIT3;  
 P1->SEL1 &= ~(BIT2 | BIT3);  
  
 /\* Configure UART \*/  
 EUSCI\_A0->CTLW0 |= EUSCI\_A\_CTLW0\_SWRST;  
 EUSCI\_A0->CTLW0 = EUSCI\_A\_CTLW0\_SWRST | EUSCI\_B\_CTLW0\_SSEL\_\_SMCLK;  
 /\* Baud Rate calculation \*/  
 /\* 49000000/118200 = 416 \*/  
 EUSCI\_A0->BRW = 416;  
 EUSCI\_A0->MCTLW = 0;  
  
 EUSCI\_A0->CTLW0 &= ~EUSCI\_A\_CTLW0\_SWRST;  
 EUSCI\_A0->IE |= EUSCI\_A\_IE\_TXIE;  
}  
  
/\*pushes next char to UART\*/  
void uartPrint(char c) {  
 while (!(EUSCI\_A0->IFG &0x02)); /\*make sure txbuf empty\*/  
 EUSCI\_A0->TXBUF = c; /\*push c\*/  
}

***v.) adc.c***

#include "msp.h"  
#include "adc.h"  
  
/\*initialize ADC\*/  
void ADC\_init() {  
 /\*ADC pin 5.4\*/  
 P5->SEL1 |= BIT4;  
 P5->SEL0 |= BIT4;  
  
  
 /\*configure ADC\*/  
 ADC14->CTL0 = ADC14\_CTL0\_SHT0\_2 | ADC14\_CTL0\_SHP | ADC14\_CTL0\_ON;  
 ADC14->CTL1 = ADC14\_CTL1\_RES\_2;  
  
 ADC14->MCTL[0] |= ADC14\_MCTLN\_INCH\_1;  
 ADC14->IER0 |= ADC14\_IER0\_IE0;  
  
 SCB->SCR &= ~SCB\_SCR\_SLEEPONEXIT\_Msk;  
}

***vi.) format.h***

#ifndef FORMAT\_H\_  
#define FORMAT\_H\_  
  
/\*locations for items when updating\*/  
#define DCBAR 7  
#define RMSBAR 67  
#define OFFSET 164  
#define PEAK 152  
#define TRUERMS 128  
#define CALCRMS 176  
#define DC 187  
#define FREQUENCY 140  
  
/\*initial frame setup\*/  
char frame[] = {0x1B, '[', '?', '2', '5', 'l', 0x1B, '[', 'H', '|','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','|','\r', '\n','|',' ','V','o','l','t',  
 'a','g','e',' ','(','V',')',' ','|','0','.','0',' ',' ',' ',  
 '0','.','5',' ',' ',' ','1','.','0',' ',' ',' ','1','.','5',  
 ' ',' ',' ','2','.','0',' ',' ',' ','2','.','5',' ',' ',' ',  
 '3','.','0',' ',' ',' ','3','.','5',' ',' ',' ','4','.','0',  
 ' ',' ','|','\r', '\n','|','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','|','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','|',  
 '\r', '\n','|',' ','D','C',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ','|',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','|','\r', '\n',  
 '|',' ','C','a','l','c','R','M','S',' ',' ',' ',' ',' ','|',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ','|','\r', '\n','|','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','|','\r', '\n','|',' ','D','C',' ','V',  
 'a','l','u','e',' ',' ',':',' ','0','.','0','0',' ','V',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ','|','\r', '\n','|','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','|',  
 '\r', '\n','|',' ','A','C',' ','V','a','l','u','e','s',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','|','\r', '\n',  
 '|','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','|','\r', '\n','|',' ','T',  
 'r','u','e',' ','R','M','S',' ',' ',':',' ','0','.','0','0',  
 ' ','V',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ','|','\r', '\n','|',' ','F','r','e','q',  
 'u','e','n','c','y',' ',':',' ','0','0','0','0',' ','H','z',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ','|','\r', '\n','|',' ','P','e','a','k','-','P','e',  
 'a','k',' ',':',' ','0','.','0','0',' ','V',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','|',  
 '\r', '\n','|',' ','D','C',' ','O','f','f','s','e','t',' ',  
 ':',' ','0','.','0','0',' ','V',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','|','\r', '\n',  
 '|',' ','C','a','l','c','R','M','S',' ',' ',' ',':',' ','0',  
 '.','0','0',' ','V',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ','|','\r', '\n','|','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','-','-','-','-','-','-','-','-','-','-',  
 '-','-','-','-','-','|', '\0'};  
  
/\*update setup\*/  
char update[] = {0x1B,'[','4',';','1','6','H',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',0x1B,'[','5',';','1','6','H',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',  
 ' ',' ',0x1B,'[','1','1',';','1','5','H','0','.','0','0',  
 0x1B,'[','1','2',';','1','5','H','0','0','0','0',0x1B,'[',  
 '1','3',';','1','5','H','0','.','0','0',0x1B,'[','1','4',  
 ';','1','5','H','0','.','0','0',0x1B,'[','1','5',';','1',  
 '5','H','0','.','0','0',0x1B,'[','7',';','1','5','H','0',  
 '.','0','0', '\0'};  
#endif /\* FORMAT\_H\_ \*/

***b.) References***

Bill of Materials: Provided by Professor Gerfen via PolyLearn shared through Cal Poly IEEE