Project 3: Digital Multimeter

Lab Partners: Ryan Myers, Mihir Deshmukh

Sections: 329-01 (Deshmukh), 329-05 (Myers)

Fall Quarter

11/6/2019

Professor: Paul Hummel

Behavior Description

Our device is a multimeter using a terminal for our display. Our multimeter is capable of making DC and AC measurements (Frequency, V_{RMS}, V_{PP} for AC). Our multimeter can measure voltage signals between 0- 3.3V and frequency signals between 1Hz to 1KH with a resolution of 10mV for voltage measurements and 1Hz for frequency measurements. We use the VT100 communication protocol to communicate with our terminal. All the above measurements are printed onto the terminal as well as a bar graph with a scale for V_{RMS} and V_{DC} values with a resolution of 100mV.

System Specification

System Components	TI SimpleLink LaunchPad COM-08653 Keypad DAC MCP4921
MCU	MSP432P401R
Power Supply Voltage	5V
Power Supply Type	USB
System Operating Frequency	12MHz
DAC Characteristics	12-Bit resolution output voltage (Minimum 8bits precision) 2.7V-5.5V operation 8 pins Serial input
Device Weight	.200 kg
Input Signals	Square, Sine, Sawtooth, DC voltages
Waveform Frequencies	1Hz - 1KHz
AC Voltage Ranges	0-3V
Min V_{PP}	0.5V
Max DC Offset	2.75V
Terminal Display and Protocol	Any serial terminal with VT100 Protocol

Table 1: System Specification Listings for Digital Multimeter

System Architecture

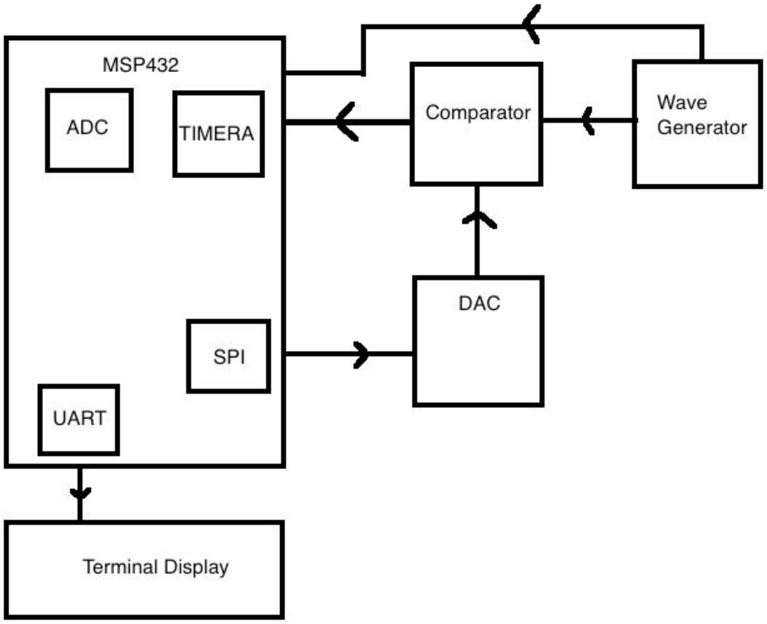

Figure 1: High level block diagram of DMM system

System Schematic

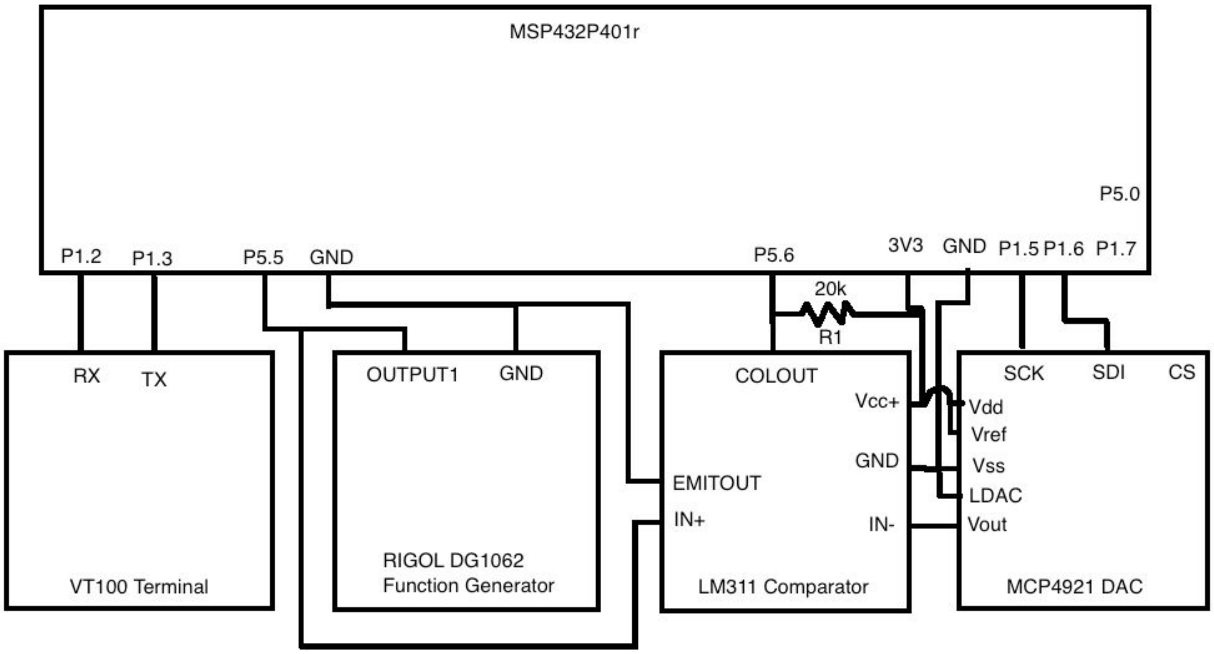

Figure 2: System schematic of DMM

Software Architecture

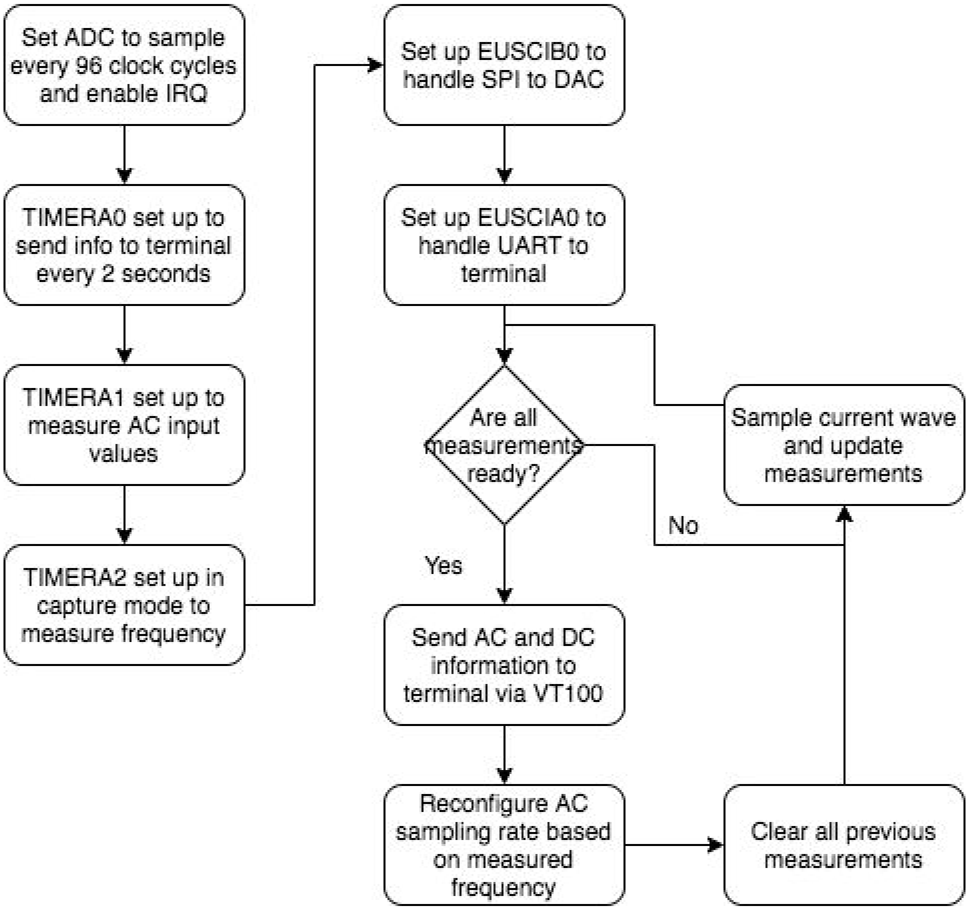
Set ADC to sample every 96 clock cycles and enable IRQ
TIMERA0 set up to send info to terminal every 2 seconds
TIMERA1 set up to measure AC input values
TIMERA2 set up in capture mode to measure frequency
Set up EUSCIB0 to handle SPI to DAC
Set up EUSCIA0 to handle UART to terminal
Are all measurements ready?
Yes
No
Sample current wave and update measurements
Send AC and DC information to terminal via VT100
Reconfigure AC sampling rate based on measured frequency
Clear all previous measurements

Figure 3: Software architecture for DMM system

Bill of Materials

Item	Part #	Supplier Name	Quantity	Price Each
20k Resistor	N/A	N/A	1	0.10
DAC	MCP4921	Microchip	1	1.97
Dev Board	MSP432P401r Launchpad	TI	1	5.00
Comparator	LM311	TI	1	2.00
VT100 Serial Terminal	2016 Macbook Pro	Apple	1	1499.00

Table 2: Bill of materials for DMM system

Source Code

```
#include "msp.h"
#include "delay.h"
#include "adc_driver.h"
#include "uart_driver.h"
#include "spi_driver.h"

/**
 * main.c
 */
void main(void)
{
     WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;          // stop
watchdog timer
     float ac_pp;
     float dc;
     float ac_rms;
     uint32_t frequency;
     set_DCO(DCORSEL_12_MHz);
     Initialize_ADC();
     Initialize_UART();
     Initialize_SPI();
    __enable_irq();
     while(1) {
         if (Read_Measurement_Flag()) {
             Send_DAC_Voltage(Read_Center());
             Write_Desc_Values_To_VT100();
             __enable_irq();
         }
     }
}

/*
 * adc_driver.h
 *
 *  Created on: Oct 30, 2019
 *      Author: ryanmyers
 */

#ifndef ADC_DRIVER_H_
#define ADC_DRIVER_H_
```

```c
#define ADCPORT P5
#define ADCPIN BIT5
#define MEASUREMENT_READY 1
#define MEASUREMENT_UNAVAILABLE 0
#define ADC_RES 16383

void Initialize_ADC(void);
void ADC14_IRQHandler(void);
float Read_AC_PP(void);
float Read_AC_RMS(void);
float Read_DC(void);
void TA0_0_IRQHandler(void);
void TA1_0_IRQHandler(void);
void TA2_N_IRQHandler(void);
uint32_t Read_Freq(void);
uint8_t Read_Measurement_Flag(void);
float Read_Center(void);

#endif /* ADC_DRIVER_H_ */

/*
 * adc_driver.c
 *
 *  Created on: Oct 30, 2019
 *      Author: ryanmyers
 */

#include "msp.h"
#include "adc_driver.h"
#include <math.h>

static volatile uint32_t ms_cnt = 0;
static volatile uint8_t Measurement_Flag;
static volatile uint16_t ADC_Value;
static volatile uint16_t peak = 0;
static volatile uint16_t trough = 16384;
static volatile uint16_t max_per_cycle;
static volatile uint16_t min_per_cycle;
static volatile uint16_t dc_measurements[10];
static volatile uint16_t ac_measurements[20];
static volatile uint32_t freq = 1;
static volatile uint32_t sampling_rate = 1;
```

```
void Initialize_ADC(void) {
    CS->KEY = CS_KEY_VAL;   // Unlock clock registers

    CS->CTL1 &= ~(CS_CTL1_DIVHS_MASK | CS_CTL1_DIVS_MASK |
CS_CTL1_SELS_MASK | CS_CTL1_DIVA_MASK | CS_CTL1_SELA_MASK); // Clear
CS registers
    CS->CTL1 |= CS_CTL1_DIVHS_0 | CS_CTL1_DIVS_0 | CS_CTL1_SELS_3 |
CS_CTL1_DIVA_0 | CS_CTL1_SELA__REFOCLK; // Set DCO to drive HSCLK and
SMCLK
    CS->CLKEN |= CS_CLKEN_REFOFSEL;

    CS->KEY = 0; // Lock clock registers

    ADC14->CTL0 &= ~ADC14_CTL0_ENC;      //Disable conversion
    ADC14->CTL0 = ADC14_CTL0_SHP |       //Enable internal sample
timer
                  ADC14_CTL0_SSEL_4 |    //Select SMCLK
                  ADC14_CTL0_CONSEQ_2 |  //Single repeat channel
                  ADC14_CTL0_SHT0_2 |    //Sample 192 clocks
                  ADC14_CTL0_MSC |       //Auto repeat
                  ADC14_CTL0_ON;         //Turn on ADC
    ADC14->CTL1 = ADC14_CTL1_RES_3;      //14 Bit resolution and
mem[0]
    ADC14->MCTL[0] = ADC14_MCTLN_INCH_0; //Set all to 0 as not needed
    ADC14->IER0 = ADC14_IER0_IE0;        //Enable interrupts on
mem[0]
    ADCPORT->SEL0 |= ADCPIN;             //Initialize port to accept
input
    ADCPORT->SEL1 |= ADCPIN;
    ADC14->CTL0 |= ADC14_CTL0_ENC |       //Enable conversion again
                   ADC14_CTL0_SC;         //Enable sampling
    NVIC->ISER[0] = 1 << (ADC14_IRQn & 0x1F);

    Measurement_Flag = MEASUREMENT_UNAVAILABLE;

    TIMER_A0->CCR[0] = 65535;
    TIMER_A0->CCTL[0] = TIMER_A_CCTLN_CCIE;
    TIMER_A0->CTL = TIMER_A_CTL_TASSEL_2 | TIMER_A_CTL_MC_1;
    NVIC->ISER[0] = (1 << (TA0_0_IRQn & 0x1F));

    TIMER_A1->CCR[0] = 600;
    TIMER_A1->CCTL[0] = TIMER_A_CCTLN_CCIE;
    TIMER_A1->CTL = TIMER_A_CTL_TASSEL_2 | TIMER_A_CTL_MC_1;
    NVIC->ISER[0] = (1 << (TA1_0_IRQn & 0x1F));

//    TIMER_A2->CCR[0] = 6000;
//    TIMER_A2->CCTL[0] = TIMER_A_CCTLN_CCIE;
//    TIMER_A2->CTL = TIMER_A_CTL_TASSEL_2 | TIMER_A_CTL_MC_1;
//

    P5->DIR &= ~BIT6;
    P5->SEL0 |= BIT6;
    P5->SEL1 &= ~BIT6;

    TIMER_A2->CCTL[1] = TIMER_A_CCTLN_CCIE |
                        TIMER_A_CCTLN_CM_2 |
                        TIMER_A_CCTLN_CCIS_0 |
                        TIMER_A_CCTLN_SCS |
                        TIMER_A_CCTLN_CAP;
    TIMER_A2->CTL = TIMER_A_CTL_TASSEL_2 | TIMER_A_CTL_MC_2 |
TIMER_A_CTL_ID__8;
    TIMER_A2->EX0 = TIMER_A_EX0_IDEX__8;
    NVIC->ISER[0] = (1 << (TA2_N_IRQn & 0x1F));
}

void ADC14_IRQHandler(void) {
    static volatile uint32_t irq_cnt = 0;
    ADC14->CLRIFGR0 |= ADC14_CLRIFGR0_CLRIFG0;
    ADC_Value = ADC14->MEM[0];
    if (ADC_Value > peak) {
        peak = ADC_Value;
    }
    else if (ADC_Value < trough) {
        trough = ADC_Value;
    }
    dc_measurements[irq_cnt++ % 10] = ADC_Value;
}

void TA0_0_IRQHandler(void) {
    TIMER_A0->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
    static uint32_t irq_cnt = 0;
    if (irq_cnt++ == 200) {
        __disable_irq();
        max_per_cycle = peak;
        min_per_cycle = trough;
        freq = 187500/ms_cnt;
        if (freq == 0) {
            freq = 1;
        }
        sampling_rate = 1000/freq;
        if (sampling_rate == 0) {
            sampling_rate = 1;
        }
        peak = 0;
        trough = 16384;
        Measurement_Flag = MEASUREMENT_READY;
        irq_cnt = 0;
    }
}

void TA1_0_IRQHandler(void) {
    static uint32_t sample_num = 0;
    static uint32_t irq_cnt = 0;
    TIMER_A1->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
    if (++sample_num == sampling_rate) {
        ac_measurements[irq_cnt % 20] = ADC_Value;
        irq_cnt++;
        sample_num = 0;
    }
}

void TA2_N_IRQHandler(void) {
    static volatile uint32_t last_cap = 0;
    static volatile uint32_t current_cap = 0;
    static volatile uint32_t overflow_cnt = 0;
    TIMER_A2->CCTL[1] &= ~TIMER_A_CCTLN_CCIFG;
    if (TIMER_A2->CCTL[1] & TIMER_A_CCTLN_COV) {
        TIMER_A2->CCTL[1] &= ~TIMER_A_CCTLN_COV;
        overflow_cnt++;
    }
    else {
        current_cap = TIMER_A2->CCR[1];
        ms_cnt = ((current_cap + (65525 * overflow_cnt)) - last_cap);
        last_cap = current_cap;
        overflow_cnt = 0;
        TIMER_A2->CCTL[1] &= ~TIMER_A_CCTLN_COV;
    }
}
```

```
float Read_AC_PP(void) {
    float pp_val = .000198 * (max_per_cycle - min_per_cycle) -
.00477;
    return pp_val;
}

float Read_AC_RMS(void) {
    float rms_val = 0;
    uint64_t rms_total = 0;
    int i;
    for (i = 0; i < 20; i ++) {
        rms_total += (ac_measurements[i] * ac_measurements[i]);
    }
    rms_total /= 20;
    rms_total = sqrt(rms_total);
    rms_val = .000198 * rms_total - .00477;
    return rms_val;
}

uint32_t Read_Freq(void) {
    return freq;
}

float Read_DC(void) {
    uint32_t dc_total = 0;
    int i;
    for (i = 0; i < 10; i++) {
        dc_total += dc_measurements[i];
    }
    dc_total /= 10;
    return .000198 * dc_total - .00477;
}

uint8_t Read_Measurement_Flag(void) {
    if (Measurement_Flag) {
        Measurement_Flag = MEASUREMENT_UNAVAILABLE;
        return MEASUREMENT_READY;
    }
    else {
        return MEASUREMENT_UNAVAILABLE;
    }
}
```

```
float Read_Center(void) {
    return .000198 * ((max_per_cycle + min_per_cycle) / 2) - .00477;
}

/*
 * uart_driver.h
 *
 *  Created on: Oct 24, 2019
 *      Author: ryanmyers
 */

#ifndef UART_DRIVER_H_
#define UART_DRIVER_H_

#define UART_PORT P1
#define UART_RXD BIT2
#define UART_TXD BIT3

#define FBRCLK_12MHz_UCBRW 6
#define FBRCLK_12MHz_UCBRF 8
#define FBRCLK_12MHz_UCBRS 0x20
#define FBRCLK_12MHz_OS16 1

#define INPUT_READY 1
#define INPUT_UNAVAILABLE 0

#define ZERO 48
#define ONE 49
#define TWO 50
#define THREE 51
#define FOUR 52
#define FIVE 53
#define SIX 54
#define SEVEN 55
#define EIGHT 56
#define NINE 57
#define NEW_LINE 10
#define RETURN 13

void Write_Desc_Values_To_VT100(void);
void Init_Desc_Values_To_VT100(void);
void EUSCIA0_IRQHandler(void);
void Initialize_UART(void);
```

```c
void Send_Serial_Char(unsigned char c);
uint16_t GetInputValue(void);
int CheckInputFlag(void);

#endif /* UART_DRIVER_H_ */

/*
 * uart_driver.c
 *
 *  Created on: Oct 24, 2019
 *      Author: ryanmyers
 */

#include "msp.h"
#include "uart_driver.h"
#include "adc_driver.h"

static volatile int input_flag = INPUT_UNAVAILABLE;
static volatile uint8_t inValue[4];

/*void EUSCIA0_IRQHandler(void) {
    static volatile int char_index = 0;
    EUSCI_A0->IFG &= ~EUSCI_A_IFG_RXIFG;
    unsigned char rx_char = EUSCI_A0->RXBUF;
    EUSCI_A0->TXBUF = rx_char;
    if (char_index < 4 && input_flag == INPUT_UNAVAILABLE) {
        switch(rx_char) {
            case ZERO:
                inValue[char_index] = 0;
                char_index++;
                break;

            case ONE:
                inValue[char_index] = 1;
                char_index++;
                break;

            case TWO:
                inValue[char_index] = 2;
                char_index++;
                break;

            case THREE:
```

```
    inValue[char_index] = 3;
    char_index++;
    break;

case FOUR:
    inValue[char_index] = 4;
    char_index++;
    break;

case FIVE:
    inValue[char_index] = 5;
    char_index++;
    break;

case SIX:
    inValue[char_index] = 6;
    char_index++;
    break;

case SEVEN:
    inValue[char_index] = 7;
    char_index++;
    break;

case EIGHT:
    inValue[char_index] = 8;
    char_index++;
    break;

case NINE:
    inValue[char_index] = 9;
    char_index++;
    break;

case NEW_LINE:
    input_flag = INPUT_READY;
    char_index = 0;
    break;

case RETURN:
    input_flag = INPUT_READY;
    char_index = 0;
    break;
```

```
            default:
                break;

        }
    }

}

int CheckInputFlag(void) {
    return input_flag;
}

uint16_t GetInputValue(void) {
    uint16_t input = 0;
    int i;
    int scaler = 1;
    for (i = 3; i >= 0; i--) {
        input += inValue[i] * scaler;
        scaler *= 10;
    }
    input_flag = INPUT_UNAVAILABLE;
    return input;
}*/

void Initialize_UART(void)
{

    CS->KEY = CS_KEY_VAL;   // Unlock clock registers

    CS->CTL1 &= ~(CS_CTL1_DIVHS_MASK | CS_CTL1_DIVS_MASK |
CS_CTL1_SELS_MASK); // Clear CS registers
    CS->CTL1 |= CS_CTL1_DIVHS_0 | CS_CTL1_DIVS_0 | CS_CTL1_SELS_3; //
Set DCO to drive HSCLK and SMCLK

    CS->KEY = 0; // Lock clock registers

    EUSCI_A0->CTLW0 = EUSCI_A_CTLW0_SWRST;

    EUSCI_A0->CTLW0 = EUSCI_A_CTLW0_SWRST
                        | EUSCI_A_CTLW0_UCSSEL_2;

    EUSCI_A0->BRW = FBRCLK_12MHz_UCBRW;
    EUSCI_A0->MCTLW = (FBRCLK_12MHz_UCBRS << 8)
                            | (FBRCLK_12MHz_UCBRF << 4)
                            | (EUSCI_A_MCTLW_OS16);

    //UART_PORT->DIR |= UART_TXD;
    //UART_PORT->DIR &= ~UART_RXD;
    UART_PORT->SEL0 |= UART_TXD | UART_RXD;
    UART_PORT->SEL1 &= ~(UART_TXD | UART_RXD);
    //UART_PORT->REN &= ~(UART_TXD | UART_RXD);

    EUSCI_A0->CTLW0 &= ~EUSCI_A_CTLW0_SWRST;

    //EUSCI_A0->IE |= EUSCI_A_IE_RXIE;
    //NVIC->ISER[0] = (1 << (EUSCIA0_IRQn & 0x1F));

    Init_Desc_Values_To_VT100();
}

void Write_Desc_Values_To_VT100(void)
{
    float val = 1.0;
    uint32_t i;
    uint32_t temp = 0;
    uint32_t Vpp = 0;
    uint32_t ones = 0;
    uint32_t tens = 0;
    uint32_t hundreds = 0;

    // Return to Top left corner
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '3';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '9';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'A';                        //Up
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Reset cursor to AC RMS
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '7';                        //7 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';                        //right
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Write Value Here in the format x.xx
    val = Read_AC_RMS();
    temp = (int)(val * 100);
    hundreds = ((temp - (temp % 100)) / 100 ) + ZERO;
    EUSCI_A0->TXBUF = hundreds;                       //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '.';                            //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    tens = (val * 100 - (hundreds-ZERO) * 100)/10 + ZERO;
    EUSCI_A0->TXBUF = tens;                            //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    ones = ((val * 100) - (hundreds - ZERO)*100 - (tens - ZERO)*10) +
ZERO;
    EUSCI_A0->TXBUF = ones;                           //4 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Reset cursor to AC RMS Bar
    EUSCI_A0->TXBUF = 0x0A;                       //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '4';                        //4 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                         //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Write number of bars as RMS
    for(i = 0; i < 33*(val/3.3); i++)
    {
        EUSCI_A0->TXBUF = 178;                         //New-line
        while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    }
    for(i = 0; i < 33-(33*(val/3.3)); i++)
    {
        EUSCI_A0->TXBUF = ' ';                         //New-line
        while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    }

    //Reset cursor to AC Vpp
    EUSCI_A0->TXBUF = 0x0A;                        //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x0A;                        //New-line (down
two lines to vpp)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                        //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                         //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '5';                         //Reset cursor to
very left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                         //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                        //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                         //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '7';                         //Reset cursor to
start of Vpp (8 right)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';                         //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Write Value Here in the format x.xx
    val = Read_AC_PP();
    temp = (int)(val * 100);
    hundreds = ((temp - (temp % 100)) / 100 ) + ZERO;
    EUSCI_A0->TXBUF = hundreds;                       //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '.';                            //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    tens = (val * 100 - (hundreds-ZERO) * 100)/10 + ZERO;
    EUSCI_A0->TXBUF = tens;                            //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    ones = ((val * 100) - (hundreds - ZERO)*100 - (tens - ZERO)*10) +
ZERO;
    EUSCI_A0->TXBUF = ones;                           //4 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Go down to AC Frq
    EUSCI_A0->TXBUF = 0x0A;                       //New-line (down
one line to AC Frq)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '3';                        //Reset cursor to
very left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '7';                        //Reset cursor to
start of Vpp (8 right)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Write Value Here in the format x.xx
    temp = Read_Freq();
    if(temp == 1000)
```

```
  {
    EUSCI_A0->TXBUF = '1';     //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';       //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';      //4 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
  }
  else if(temp < 1000 && temp >= 100)
  {
    hundreds = temp/100;
    EUSCI_A0->TXBUF = hundreds + ZERO;
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    tens = (temp - hundreds*100)/10;
    EUSCI_A0->TXBUF = tens + ZERO;
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = temp - hundreds*100 - tens*10 + ZERO;
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
  }
  else if(temp < 100 && temp >= 10)
  {
    EUSCI_A0->TXBUF = ZERO + temp/10;
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ZERO + (temp - (temp/10)*10);
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
  }
  else
  {
    EUSCI_A0->TXBUF = ZERO + temp;
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
  }

  //Go down to DC Val
  EUSCI_A0->TXBUF = 0x0A;       //New-line (down
one line to DCV)
  while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
  EUSCI_A0->TXBUF = 0x1B;       //Esc
  while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
  EUSCI_A0->TXBUF = '[';        //[
  while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
```

```
    EUSCI_A0->TXBUF = '3';                        //Reset cursor to
very left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '7';                        //Reset cursor to
start of Vpp (8 right)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Write Value Here in the format x.xx
    val = Read_DC();
    temp = (int)(val * 100);
    hundreds = ((temp - (temp % 100)) / 100 ) + ZERO;
    EUSCI_A0->TXBUF = hundreds;                       //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '.';                            //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    tens = (val * 100 - (hundreds-ZERO) * 100)/10 + ZERO;
    EUSCI_A0->TXBUF = tens;                            //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    ones = ((val * 100) - (hundreds - ZERO)*100 - (tens - ZERO)*10) +
ZERO;
    EUSCI_A0->TXBUF = ones;                           //4 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Go down to DC Bar
    EUSCI_A0->TXBUF = 0x0A;                       //New-line (down
one line to DC Bar)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '3';                        //Reset cursor to
very left
```

```
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '0';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '7';                        //Reset cursor to
start of Vpp (8 right)
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //Write number of bars as RMS
    for(i = 0; i < 33*(val/3.3); i++)
    {
        EUSCI_A0->TXBUF = 178;                        //New-line
        while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    }
    for(i = 0; i < 33-(33*(val/3.3)); i++)
    {
        EUSCI_A0->TXBUF = ' ';                        //New-line
        while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    }

    // Return to Top left corner
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '3';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '9';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'A';                        //Up
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
}

/*
 * Initiates the displaying of the following values
 * AC RMS (True RMS including DC offset)
 * AC RMS bar graph
 * AC RMS bar graph scale
 * AC VPP
 * AC Frq
 * DC Vol
 * DC Vol bar graph
 * DC Vol bar graph scale
 */
void Init_Desc_Values_To_VT100(void)
{
    //AC RMS
    EUSCI_A0->TXBUF = 'A';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ' ';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'R';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'M';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'S';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ':';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ' ';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x0A;                       //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //AC Bar graph
    EUSCI_A0->TXBUF = 'A';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ' ';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'B';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'A';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'R';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ':';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ' ';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x0A;                       //New-line
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //6 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    //AC Bar scale
    EUSCI_A0->TXBUF = 'A';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'C';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = ' ';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'S';
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
```

```
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'L';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ':';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '0';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '1';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
```

```
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '2';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '.';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x0A;                       //New-line
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x1B;                       //Esc
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '[';                        //[
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '4';                        //43 position
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
```

```
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'D';                        //left
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

//AC Vpp
EUSCI_A0->TXBUF = 'A';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'V';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'p';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'p';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ':';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x0A;                       //New-line
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x1B;                       //Esc
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '[';                        //[
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '8';                        //6 position
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'D';                        //left
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

//AC Frequency
EUSCI_A0->TXBUF = 'A';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'F';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'R';
```

```
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'Q';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ':';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x0A;                       //New-line
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x1B;                       //Esc
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '[';                        //[
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '8';                        //6 position
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'D';                        //left
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

//DC Voltage
EUSCI_A0->TXBUF = 'D';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'V';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'O';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'L';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ':';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x0A;                       //New-line
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x1B;                       //Esc
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '[';                        //[
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '8';                        //6 position
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
```

```c
EUSCI_A0->TXBUF = 'D';                        //left
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

//DC Bar
EUSCI_A0->TXBUF = 'D';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'B';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'A';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'R';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ':';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x0A;                       //New-line
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x1B;                       //Esc
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '[';                        //[
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '8';                        //6 position
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'D';                        //left
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

//DC Scale
EUSCI_A0->TXBUF = 'D';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ' ';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'S';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'C';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'L';
```

```
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = ':';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '0';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '1';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
```

```
EUSCI_A0->TXBUF = '2';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '|';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '.';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x0A;                        //New-line
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 0x1B;                        //Esc
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '[';                         //[
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '4';                         //43 position
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = '3';
while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
EUSCI_A0->TXBUF = 'D';                         //left
```

```
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    // Return to Top left corner
    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '3';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '9';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'A';                        //Up
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

    EUSCI_A0->TXBUF = 0x1B;                       //Esc
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '[';                        //[
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = '8';                        //8 position
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
    EUSCI_A0->TXBUF = 'D';                        //left
    while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));
}
```