APPENDIX

CodeWarrior Code

```
//*********************************************************************
//*                              PROJECT
//*                         McMaster University
//*             COMP ENG 2DP4 - Microprocessors Systems
//*             Kapithaan Pathmanathan pathmk2 001415745
//*********************************************************************
//*********************************************************************
//*                         Program Description
//*   The following program is used as an acquisition system to accept
//*   an analog signal, process the signal and trasmit the signal to
//*   MATLAB.
//*
//*                         Project Requirements
//*   1.  14 MHz e-clock (bus speed)
//*   2.  ADC Resolution: 12-bit resolution
//*   3.  ADC Channel: AN5
//*
//*********************************************************************
//*********************************************************************
//*                              References
//*   1.  MC9S12G Reference and Data Sheet Manual
//*   2.  HSC12/9S12 - An Introduction to Software and Hardware Interfacing
textbook by Han-Way Huang
//*   3.  Lecture / Tutorials / Lab Templates posted on Avenue
//*
//*********************************************************************

/*Include*/
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"  /* derivative information */
#include "SCI.h"

/*Prototypes*/
void setEClk14(void);              //  Set bus speed to 14 MHz
void verifyClk(void);              //  Function used to test initial E-Clock
Speed
void delay1ms(unsigned int x);     //  Input the number of milliseconds you
want to delay
void setADC(void);                 //  Configures ADC registers
void OutCRLF(void);                //  Output a CR,LF to SCI to move cursor to
a new line
void verifySerialCommunication(void);   // test and verify Serial
Communication between ESDX and PC
void BCDLED(unsigned int n);       //  Lights LED in BCD format
void newdelay1ms(unsigned int numTimes);

unsigned int n;
unsigned int flag;
unsigned int counter = 0;          //  initialize counter variable for
interrupt
unsigned int freq = 0;             //  initialize frequency variable
```

```
unsigned int val;                  //  variable to store ADC value from port
AN5
unsigned int BPM;

//*********************************************************************
//                              MAIN PROGRAM
//*********************************************************************

void main(void) {


    PTJ = 0x00;           //  LED is toggled off at the start of the program
    DDRJ = 0xFF;          //  Set on board LED (pin D13) to an output
    DDRM = 0xFF;          //  Set Port M to digital outputs
    DDRP = 0x0F;          //  Set lower 4-bits of Port P as outputs
    DDR1AD = 0x0F;        //  Set lower 4-bits of PortAD as outputs

    setEClk14();          //  Set E-Clock (bus speed) to 14 MHz
    //verifyClk();
    //verifySerialCommunication();

    SCI_Init(115200);
    setADC();


/*
 * The next six assignment statements configure the Timer Input Capture
 */
    TSCR1 = 0x90;     // Timer System Control Register 1
                      // 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 = 0x00;     // 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
                      // 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
    TCTL4 = 0x02;     // Configured for falling edge on TIC[0]

/*
 * The next one assignment statement configures the Timer Interrupt Enable
```

```
 */
    TIE = 0x01;       // Timer Interrupt Enable

    EnableInterrupts;


    for(;;){

      if(counter%2==1){            //button is pressed to communicate
         while(counter%2==1){
            val = ATDDR0;
            if(val<100){
                flag=0;
                freq=0;
                for(n=0;n<500;n++){
                    val = ATDDR0;
                    if(counter%2==0){
                      break;
                    }
                    if(val>=100){
                        if(flag ==0){
                            flag = 1;
                        }
                    }
                    else{
                        if(flag ==1){
                            flag = 0;
                            freq++;
                        }
                    }
                    SCI_OutString("Value = ");
                    SCI_OutUDec(val);
                    OutCRLF();
                    newdelay1ms(10);
                }

                if(counter%2==0){
                    break;
                }

                BPM = freq*12;
                SCI_OutString("Beats/Min = ");
                BCDLED(BPM);
                SCI_OutUDec(BPM);
                break;
            }
         }
      }
      else {                           //button is not pressed or data
aquisition stopped
          while(counter%2==0){
              PTJ = 0x00;
              PTP = 0x00;
              PT1AD = 0x00;
              PTM = 0x00;
          }
```

```
      }

    }// loops forever
}



//*********************************************************************
//                            INTERRUPTS
//*********************************************************************

//  the following interrupt is used to recognize when the button is pushed to
enable/disable serial communication
interrupt VectorNumber_Vtimch0 void ISR_Vtimch0(void){
    unsigned int temp;
    PTJ^= 0x01;
    counter++;

    temp = TC0;       //Refer back to TFFCA, we enabled FastFlagClear, thus
by reading the Timer Capture input we automatically clear the flag, allowing
another TIC interrupt

}


//*********************************************************************
//                            FUNCTIONS
//*********************************************************************

//-------------------setEClk14---------------------
// Setup target E-Clock (Bus Clock) to 14 Mhz
void setEClk14(void){
   CPMUCLKS = 0x80;          //  PLLSEL = 1
   CPMUOSC = 0x00;           //  OSCE = 0, fREF = 1MHz
   CPMUSYNR = 0x0D;          //  VCOFRQ = 0, SYNDIV = 13
   CPMUPOSTDIV = 0x00;       //  POSTDIV = 0
   CPMUREFDIV = 0x00;        //  REFFRQ = 00
   // fVCO = 2 * fREF * (SYNDIV+1) = 28
   // PLLCLK = fVCO / (1+POSTDIV) = fVCO / 1 = 28
   // fBUS = PLLCLK / 2 = 14 MHz
}

//-------------------verifyClk-----------------------
//  To Test and Verify E-Clock
void verifyClk(void){
   PTJ=0x00;
   while(1){
      PTJ = 0x00;
      delay1ms(1000);         // delay of 1 second
      PTJ = 0x01;             // turn on LED on ESDX
      delay1ms(1000);
   }
}

//------------------delay1ms----------------------
//  Implements a 1ms delay function
```

```
//  Derived from Lec W8
void delay1ms(unsigned int k){
  unsigned int ix;
  TSCR1 = 0x90;         // enable timer and fast timer flag clear
  TSCR2 = 0x00;         // disable timer interrupt, set prescaler to 1
  TIOS|=0x01;           // enable OC0 (not necessary)
  TC0 = TCNT + 14000;
  for(ix=0;ix<k;ix++) {
    while (!(TFLG1_C0F));
    TC0 += 14000;
  }
  TIOS &=~0x01;         // disable OC0 (not necessary)
}

//---------------------setADC----------------------
// Configure ADC Registers
// Setup and enable ADC channel 5
// Refer to Chapter 14 in S12G Reference Manual for ADC subsystem details
void setADC(void){

    ATDCTL1 = 0x4F;     // set for 12-bit resolution
    ATDCTL2 = 0x00;
      ATDCTL3 = 0x88;       // right justified, one sample per sequence
      ATDCTL4 = 0x06;       // prescaler = 6; ATD clock = 14MHz / (2 * (6 +
1)) == 1 MHz
      ATDCTL5 = 0x25;       // continuous conversion on channel 5, port AN5
}

//---------------------OutCRLF---------------------
// Output a CR,LF to SCI to move cursor to a new line
// Input: none
// Output: none
// Toggle LED each time through the loop

void OutCRLF(void){
  SCI_OutChar(CR);
  SCI_OutChar(LF);
  PTJ ^= 0x20;          // toggle LED D2
}

//------------------verifySerialCommunication--------
// Used to test and verify serial communication between ESDX and PC
void verifySerialCommunication(void){
  SCI_Init(115200);
  SCI_OutString("Test - ESDX communication with MatLab");
  SCI_OutChar(CR);
  SCI_OutString("1,2,3 ... It works! ");
  SCI_OutChar(CR);
}

//------------------BCDLED-------------------------
// Function to light LED in Binary Coded Decimal Format
void BCDLED(unsigned int n){
   unsigned char BCDvalues[10] = {
    0x00,0x01,0x02,0x03,0x04,0x05, 0x06, 0x07, 0x08, 0x09
   };
   unsigned int hundreds;
```

```
    unsigned int tens;
    unsigned int ones;
    unsigned int remainder;

    hundreds = n/100;
    remainder = n%100;
    tens = remainder/10;
    ones = remainder%10;

    PTM = BCDvalues[hundreds];
    PTP = BCDvalues[tens];
    PT1AD = BCDvalues[ones];

}

//-----------------newdelay1ms---------------------
// Sampling delay
void newdelay1ms(unsigned int numTimes){
    unsigned int i,j;
    for(i=0;i<numTimes;i++){
        for(j=0;j<2333;j++){    // 14000 / 6 clock cycles
        }
    }
}
```

MATLAB Code

```
%   2DP4 - Microproccesors Systems - Final Project
%   Serial Communication between ESDX and MATLAB
%   Kapithaan Pathmanathan - pathmk2 - 001415745

clc;
clear;

delete(instrfindall);       % delete all and close any ports
s = serial('COM2');         % select COM Port 2
s.BaudRate = 115200;        % set Baud Rate based on 14 Mhz bus speed: 115200
s.Terminator = 'CR';        % set CR as terminator
fopen(s);                   % open file

%Initilization - Variables
VFS = 5;                    % full scale voltage
bits = 12;                  % number of bits
voltage = 0;                % Voltage Low
res = (VFS/((2^bits)-1));   % resolution (step size)


% Plot Parameters
% set x limits (time)
xmin = 0;
xmax = 500;
% set y limits (voltage)
ymin = 0;
ymax = 5;

%Plot Titles
title('Voltage.vs.Time: Kapithaan Pathmanathan 001415745')
xlabel('Time (seconds)')
ylabel('Voltage (V)')

line = animatedline;
startT = datetime('now');

while 1>0
    Output = fgetl(s)               % returns the next line (value)
    if isempty(Output)              % do nothing if no output value is
present
        val = 0;
    else
        val = str2double(Output)*res      % normalize ADC value
    end
    t = datetime('now') - startT;        % calculate time
    addpoints(line,datenum(t),val)

    [x,y] = getpoints(line);
    xlim(datenum([t-seconds(10) t]));

    drawnow          % draw continuous waveform using values serially
communicated by ESDX
```

```
end

% make sure to close COM port and delete on going
fclose(s);
delete(s);
clear (s);
```