Appendix B: Complete Software Listings

Contents

1. MPLAB Microcontroller Programming

```
/*
* File: AFE_SD_code.c
* Author: HOTM
*
* Created on May 3, 2017, 3:30 PM
*/

#include "FSIO.h"
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include "SDlib270.h"
#define INPUT 1
#define OUTPUT 0
#define ADC_READY LATBbits.LATB7

#ifdef __XC32

#pragma config FNOSC = PRIPLL // Oscillator selection
#pragma config POSCMOD = EC // Primary oscillator mode
#pragma config FPLLIDIV = DIV_1 // PLL input divider (4 -> 4)
#pragma config FPLLMUL = MUL_20 // PLL multiplier  ( 4x20 = 80)
#pragma config FPLLODIV = DIV_4 // PLL output divider
#pragma config FPBDIV = DIV_2 // Peripheral bus clock divider 10 mhz
#pragma config FSOSCEN = OFF // Secondary oscillator enable
/* Clock control settings
*/
#pragma config IESO = OFF // Internal/external clock switchover
#pragma config FCKSM = CSECME // Clock switching (CSx)/Clock monitor (CMx)
#pragma config OSCIOFNC = OFF // Clock output on OSCO pin enable
/* USB Settings
*/
#pragma config UPLLEN = OFF // USB PLL enable
#pragma config UPLLIDIV = DIV_2 // USB PLL input divider
#pragma config FVBUSONIO = OFF // VBUS pin control
#pragma config FUSBIDIO = OFF // USBID pin control
/* Other Peripheral Device settings
*/
#pragma config FWDTEN = OFF // Watchdog timer enable
#pragma config WDTPS = PS1024 // Watchdog timer post-scaler
//#pragma config FSRSSEL = PRIORITY_7 // SRS interrupt priority
#pragma config DEBUG = ON
#pragma config JTAGEN = OFF

#pragma config    ICESEL    = ICS_PGx2    // ICE pin selection
#else

#if defined (__PIC32MX__)
        #pragma config FPLLMUL  = MUL_20    // PLL Multiplier
```

```
        #pragma config FPLLIDIV = DIV_2     // PLL Input Divider
        #pragma config FPLLODIV = DIV_1     // PLL Output Divider
        #pragma config FPBDIV   = DIV_2     // Peripheral Clock divisor
        #pragma config FWDTEN   = OFF       // Watchdog Timer
        #pragma config WDTPS = PS1          // Watchdog Timer Postscale
        #pragma config FCKSM = CSDCMD       // Clock Switching & Fail Safe Clock Monitor
        #pragma config OSCIOFNC = OFF       // CLKO Enable
        #pragma config POSCMOD  = HS        // Primary Oscillator
        #pragma config IESO     = OFF       // Internal/External Switch-over
        #pragma config FSOSCEN  = OFF       // Secondary Oscillator Enable (KLO was off)
        #pragma config FNOSC= PRIPLL        // Oscillator Selection
        #pragma config CP       = OFF       // Code Protect
        #pragma config BWP      = OFF       // Boot Flash Write Protect
        #pragma config PWP      = OFF       // Program Flash Write Protect
        #pragma config ICESEL   = ICS_PGx2  // ICE/ICD Comm Channel Select
        #pragma config DEBUG = ON           // Background Debugger Enable
#endif
#endif

#define LARGE_NUMBER 100000000

// FILE1.TXT
const    char str1[] =
{'D',0x00,'A',0x00,'T',0x00,'A',0x00,'2',0x00,'9',0x00,'.',0x00,'T',0x00,'X',0x00,'T',0x00,'\0',0x00};


void config_pic_pins(void){
        TRISBbits.TRISB0 = OUTPUT;    //pin21
        TRISBbits.TRISB7 = INPUT;     //ADC_READY
        REFOTRIMbits.ROTRIM = 0;
        REFOCONbits.RODIV = 0;
        REFOCONbits.ROSEL = 0;        //refclk fed by sysclk; can be mapped to any pin, see table 11-2
        RPB9R = 0b0111;               //refclk0 mapped to pin 1

        ANSELA = 0;
        ANSELB = 0;
        ANSELC = 0;

        //pic2afe
        SDI1R = 0b0010;                 //SDI1 mapped to B1 (SOMI/MISO)
        RPC3R = 0b0011;                 //SDO1 mapped to C3 (SIMO/MOSI)
        //pic2SD
        SDI2R=0b0100;                   //SDI2 to B2
        RPA8R=0b0100;                   //SDO2 to A8
}

void config_spi1(void){
        SPI1CONbits.ON = 0;             //turn off SPI

        SPI1CONbits.CKE = 1;            //output data switches on idle to active transition
        SPI1CONbits.SMP = 0;            //input data sampled at middle of data output time
```

```
	SPI1CONbits.CKP = 0;			//idle state low, active state high

	SPI1CONbits.MSTEN = 1;				//pic_spi is master, sck1 coming from afe
	SPI1STATbits.SPIROV = 0;			//SPI2BUF has not overflowed

	SPI1CONbits.MCLKSEL = 0;			//PBCLK is used by the Baud Rate Generator

	SPI1BRG = 0;		//F_sck = F_pb/(2*(SPI_BRG+1))
				  //F_sck = 5MHz
				  //F_pb = 10MHz
				  //SPI_BRG = 0

	SPI1CONbits.ON = 1;		//turn on SPI
}

unsigned char spi1send(unsigned char data){
	IFS1bits.SPI1RXIF = 0;			//clear receive interrupt flag
	unsigned char ans;
	SPI1BUF = data;				//send data to buffer
	while(!IFS1bits.SPI1RXIF);		//wait while buffer is full
	ans = SPI1BUF;
	return ans;
}

void sf_write(unsigned char address, unsigned long data){
	LATBbits.LATB0 = 0;		//chip select low

	spi1send(address);		//data to send
	spi1send((data >> 16));		//msB of data
	spi1send((data >> 8));		//middle Byte of data
	spi1send((data));		//lsB of data

	LATBbits.LATB0 = 1;		//chip select high
}

void timer_reg_init (void){
 LATBbits.LATB0 = 0;			//chip select low
 sf_write(0x00, 0x000000);		//write enable
 sf_write(0x01, 0x0017C0);		//LED2STC = 6050
 sf_write(0x02, 0x001f3e);		//LED2ENDC = 7998
 sf_write(0x03, 0x001770);		//LED2LEDSTC = 6000
 sf_write(0x04, 0x001f3f);		//LED2LEDENDC = 7999
 sf_write(0x05, 0x000050);		//ALED2STC = 50
 sf_write(0x06, 0x0007ce);		//ALED2ENDC = 1998
 sf_write(0x07, 0x000820);		//LED1STC = 2050
 sf_write(0x08, 0x000f9e);		//LED1ENDC = 3998
 sf_write(0x09, 0x0007d0);		//LED1LEDSTC = 2000
 sf_write(0x0a, 0x000f9f);		//LED1LEDENDC = 3999
 sf_write(0x0b, 0x000ff0);		//ALED1STC = 4050
 sf_write(0x0c, 0x00176e);		//ALED1ENDC = 5998
 sf_write(0x0d, 0x000006);		//LED2CONVST = 4
```

```
 sf_write(0x0e, 0x0007cf);                  //LED2CONVEND = 1999
 sf_write(0x0f, 0x0007d6);                  //ALED2CONVST = 2004
 sf_write(0x10, 0x000f9f);                  //ALED2CONVEND = 3999
 sf_write(0x11, 0x000fa6);                  //LED1CONVST = 4004
 sf_write(0x12, 0x00176f);                  //LED1CONVEND = 5999
 sf_write(0x13, 0x001776);                  //ALED1CONVST = 6004
 sf_write(0x14, 0x001f3f);                  //ALED1CONVEND = 7999
 sf_write(0x15, 0x000000);                  //ADCRSTSTCT0 = 0
 sf_write(0x16, 0x000005);                  //ADCRSTENDCT0 = 3
 sf_write(0x17, 0x0007d0);                  //ADCRSTSTCT1 = 2000
 sf_write(0x18, 0x0007d5);                  //ADCRSTENDCT1 = 2003
 sf_write(0x19, 0x000fa0);                  //ADCRSTSTCT2 = 4000
 sf_write(0x1a, 0x000fa5);                  //ADCRSTENDCT2 = 4003
 sf_write(0x1b, 0x001770);                  //ADCRSTSTCT3 = 6000
 sf_write(0x1c, 0x001775);                  //ADCRSTENDCT3 = 6003
 sf_write(0x1d, 0x001f3f);                  //PRPCOUNT = 7999
 LATBbits.LATB0 = 1;                        //chip select high
}

void config_afe (void){
 LATBbits.LATB0 = 0;                        //chip select low
 sf_write(0x1e, 0x000101);                  //CONTROL1: 7: CLKALMPIN = 000; TIMEREN = 0;
NUMAV = 1;
 sf_write(0x20, 0x000000);                  //TIAGAIN: R_f, C_f, Stage 2 gain values are same for LED2,
LED1; STAGE2EN1: stage 2 bypassed; STG2GAIN1 = 0 (0dB); Cf_LED1 = 0 (5pF); Rf_LED1 = 0
(500kohms);
 sf_write(0x21, 0x000000);                  //TIA_AMB_GAIN: AMBDAC (ambient dac value) = 0 (0uA);
FLTRCNRSEL (filter corner select) = 0 (500Hz); STG2GAIN2 = 0 (0dB); Cf_LED2 = 0 (5pF);
Rf_LED2 = 0 (500kohms);
 sf_write(0x22, 0x010f0f);                  //LED_RANGE: 00: Imax = 100mA, Vhr = 1.1V; LED1[7:0];
LED2[7:0]; full scale current = 100mA
 sf_write(0x23, 0x020100);                  //TX_REF; RST_CLK_ON_PD_ALM; EN_ADC_BYP;
TXBRGMOD; DIGOUT_TRISTATE; XTALDIS; EN_SLOW_DIAG; PDN_TX; PDN_RX; PDN_AFE
 //rest are spare//
 LATBbits.LATB0 = 1;                        //chip select high
}

void put_in_read_mode(void){
        LATBbits.LATB0 = 0;                     //chip select low
        sf_write(0x00, 0x000001);               //set register to read mode
        LATBbits.LATB0 = 1;                     //chip select high
}

long sf_read(unsigned char address){
        long response;
        unsigned char most_sig_B, mid_sig_B, least_sig_B;
        /*SPI_READ = 0, set value of first timer register(address, data), write 1 to SPI_READ register,
read value of first timer register (send address, three bytes of 0s)
        * send address,
        send byte of zeros (non-op) get first byte back, store as msB
        send byte of zeros (non-op) get second byte back, store as next msB
```

```
        send byte of zeros (non-op) get third byte back, store as lsB
        */

        //send
        LATBbits.LATB0 = 0;                 //chip select low
        spi1send(address);                  //send address to read

        //receive
        most_sig_B = spi1send(0x00);        //catch msB of response with non op
        mid_sig_B = spi1send(0x00);         //catch mid_sB of response with non op
        least_sig_B = spi1send(0x00);       //catch lsB of response with non op

        LATBbits.LATB0 = 1;                 //chip select high

        //concatenate bytes to single word
        response = least_sig_B|(mid_sig_B<<8)|(most_sig_B<<16)|(0x00<<24);
        return(response);                   //return response
}

double volt_convert (int volt_in_22_2s){
        //verify 2s compliment
        //if incoming msb (bit21) is 1, fill in all higher bits also equal to 1
        int volt_in_32_2s;
        int volt_in;
        double volt_out;
        if((volt_in_22_2s >> 21) == 1){
        volt_in_32_2s = (volt_in_22_2s||0xFFC00000);
        volt_in = (~volt_in_32_2s) + 1;
        volt_out = -((double)volt_in*1.2/2097151);
        }else{
        volt_in = volt_in_22_2s;
        volt_out = (double)volt_in*1.2/2097151;
        }

        return volt_out;
}

int main (void){
        int led2val, aled2val, led1val, aled1val, led2_diff_aled2val, led1_diff_aled1val;
        double led2volt, aled2volt, led1volt, aled1volt, led2_diff_aled2volt, led1_diff_aled1volt;
        config_pic_pins();

        config_spi1();

        timer_reg_init();

        config_afe();

        put_in_read_mode();

  FSFILE * pointer;
```

```
  #if defined(SUPPORT_LFN)
  char count = 80;
  #endif
  char * pointer2;
  SearchRec rec;
  unsigned char attributes;
  unsigned char size = 0, i;

          char disp = 0x0;

  while (!MDD_MediaDetect());
  // Initialize the library
  while (!FSInit());
#ifdef ALLOW_WRITES
  // Create a file
  pointer = FSfopen ("DATA29.TXT", "w");
  if (pointer == NULL)
          while(1);
          FSfprintf(pointer,"%c%c led2volt:   , aled2volt:   , led1volt:          , aled1volt:        ,
led2_diff_aled2volt:       , led1_diff_aled1volt:     ",0x0D, 0x0A);
          int a;
          for(a=0; a<1000;a++){
          led2val = sf_read(0x2a);
          aled2val = sf_read(0x2b);
          led1val = sf_read(0x2c);
          aled1val = sf_read(0x2d);
          led2_diff_aled2val = sf_read(0x2e);
          led1_diff_aled1val = sf_read(0x2f);


  // Write 21 1-byte objects from sendBuffer into the file
  //if (FSfwrite (sendBuffer, 1, 21, pointer) != 21)
          // while(1);
  // FSftell returns the file's current position
  //if (FSftell (pointer) != 21)
          // while(1);
//        double test_volt;
//        int test_val=0x3fffff;
//        test_volt=volt_convert(test_val);
          led2volt = volt_convert(led2val);
          aled2volt = volt_convert(aled2val);
          led1volt = volt_convert(led1val);
          aled1volt = volt_convert(aled1val);
          led2_diff_aled2volt = volt_convert(led2_diff_aled2val);
          led1_diff_aled1volt = volt_convert(led1_diff_aled1val);


//FSfprintf(pointer,"%c%c %d,  %d,      %d,      %d,      %d,      %d", 0x0D, 0x0A,
(long)(led2val*LARGE_NUMBER),(long)(aled2val*LARGE_NUMBER),
(long)(led1val*LARGE_NUMBER), (long)(aled1val*LARGE_NUMBER),
(long)(led2_diff_aled2val*LARGE_NUMBER), (long)(led1_diff_aled1val*LARGE_NUMBER));
```

```
        FSfprintf(pointer,"%c%c %d,    %d,      %d,      %d,      %d,      %d", 0x0D, 0x0A,
(led2volt),(aled2volt), (led1volt), (aled1volt), (led2_diff_aled2volt), (led1_diff_aled1volt));
// FSfseek sets the position one byte before the end
// It can also set the position of a file forward from the
// beginning or forward from the current position
  if (FSfseek(pointer, 1, SEEK_END))


        while(1);
        }
  // Close the file
  if (FSfclose (pointer))
        while(1);

#endif

  while(1);
}
```

2. MATLAB PPG Analysis

2.1 Analysis_Custom.m

```
% Calculate AI, IPA, and PPT for custom board

function [IPAAverage,AIAverage,PPTAverage]=Analysis_Custom(concHbO2)
Frequency = 106.4;
resample_factor_conc = 1;

%% Find peaks of concentration waveform

concHbO2_smooth=smooth(concHbO2,0.04,'loess');
length_concHbO2=length(concHbO2_smooth);
[peaksconc,locsconc]=findpeaks(concHbO2_smooth,1:length_concHbO2,'MinPeakProm
inence',0.00000004);
findpeaks(concHbO2_smooth,1:length_concHbO2,'MinPeakProminence',0.00000004);
numpeaksconc=length(peaksconc);
% Find minimums

conc_neg=concHbO2_smooth*(-1);
[minconc,locsminconc]=findpeaks(conc_neg,1:length_concHbO2,'MinPeakProminence
',0.00000004);
findpeaks(conc_neg,1:length_concHbO2,'MinPeakProminence',0.00000004);
minconc=minconc*(-1);
numminsconc=length(minconc);

%% Isolate one waveform - make more robust by using concavity
RiseTimeAggregate=0;
PWDAggregate=0;
PPTAggregate=0;
DiastolicPhaseAggregate=0;
AIAggregate=0;
IPAAggregate=0;
count=0;
for n=1:(numminsconc-3)
    min1loc=locsminconc(n+1);
    min2loc=locsminconc(n+2);

    length_onewave= min2loc-min1loc;
    for i=1:length_onewave
        onewave(i)=concHbO2_smooth(i+min1loc);
    end
    %plot(onewave);

[onewave_peaks,onewave_peaks_locs]=findpeaks(onewave,1:length_onewave,'MinPea
kProminence',0.00000002);

    systolic_peak=onewave_peaks(1);
    systolic_loc=onewave_peaks_locs(1);

    if ((length(onewave_peaks))>1)
        diastolic_peak=onewave_peaks(2);
        diastolic_loc=onewave_peaks_locs(2);

        baseline= onewave(1);
```

```
        % Find dicrotic notch, systolic peak = Pulse wave amplitude (PWA)
        onewave_neg=onewave*(-1);
        %findpeaks(onewave_neg,1:length_onewave,'MinPeakProminence',0.00003)

[dicrotic_notch,locdicrotic1]=findpeaks(onewave_neg,1:length_onewave,'MinPeak
Prominence',0.000000001);
        dicrotic=dicrotic_notch*(-1);
        mag_dicrotic=dicrotic-baseline;
        mag_systolic=systolic_peak-baseline;
        mag_diastolic=diastolic_peak-baseline;
        locdicrotic=locdicrotic1(1);
        %% Timing

        % Rise time = Systolic Phase
        RiseTime=systolic_loc*resample_factor_conc*(1/Frequency);

        % Pulse Wave Duration = Length (onewave)
        PWD = length_onewave*resample_factor_conc*(1/Frequency);

        % Pulse Propogation Time
        PPT = (diastolic_loc-
systolic_loc)*resample_factor_conc*(1/Frequency);

        % Diastolic Phase
        DiastolicPhase = (length_onewave-
systolic_loc)*resample_factor_conc*(1/Frequency);

        % Augmentation Index
        AI=mag_diastolic/mag_systolic;

        %% Integrate under signal

        %A1
        for k=1:locdicrotic
            A1(k)=onewave(k);
        end

        %A2
        A2_length=length_onewave-locdicrotic;
        for s=1:A2_length
            A2(s)=onewave(s-1+locdicrotic);
        end

        % Inflection Point Area Ratio
        A1_trap=trapz(A1);
        A2_trap=trapz(A2);

        IPA=A2_trap/A1_trap;


        %% Calculate Averages
        RiseTimeAggregate=RiseTimeAggregate+RiseTime;
        PWDAggregate = PWDAggregate + PWD;
        PPTAggregate = PPTAggregate + PPT;
        DiastolicPhaseAggregate = DiastolicPhaseAggregate+ DiastolicPhase;
```

```
        AIAggregate = AIAggregate + AI;
        IPAAggregate = IPAAggregate + IPA;
        count=count+1;

    end
    clear onewave;
end
    RiseTimeAverage=RiseTimeAggregate/count;
    PWDAverage=PWDAggregate/count;
    PPTAverage=PPTAggregate/count;
    DiastolicPhaseAverage=DiastolicPhaseAggregate/count;
    AIAverage=AIAggregate/count;
    IPAAverage=IPAAggregate/count;
end
```

2.2 Analysis.m

```
% Calculate AI, IPA, and PPT
function [IPAAverage,AIAverage,PPTAverage]=Analysis(concHbO2)
Frequency = 500;
resample_factor= 1;
resample_factor_conc = 1;

%% Find peaks of concentration waveform

concHbO2_smooth=smooth(concHbO2,0.01,'loess');
length_concHbO2=length(concHbO2_smooth);
[peaksconc,locsconc]=findpeaks(concHbO2_smooth,1:length_concHbO2,'MinPeakProm
inence',0.00004);
findpeaks(concHbO2_smooth,1:length_concHbO2,'MinPeakProminence',0.00004);
numpeaksconc=length(peaksconc);
% Find minimums

conc_neg=concHbO2_smooth*(-1);
[minconc,locsminconc]=findpeaks(conc_neg,1:length_concHbO2,'MinPeakProminence
',0.0001);
findpeaks(conc_neg,1:length_concHbO2,'MinPeakProminence',0.0001);
minconc=minconc*(-1);
numminsconc=length(minconc);

%% Isolate one waveform - make more robust by using concavity
RiseTimeAggregate=0;
PWDAggregate=0;
PPTAggregate=0;
DiastolicPhaseAggregate=0;
AIAggregate=0;
IPAAggregate=0;
count=0;
for n=1:(numminsconc-3)
    min1loc=locsminconc(n+1);
    min2loc=locsminconc(n+2);

    length_onewave= min2loc-min1loc;
    for i=1:length_onewave
        onewave(i)=concHbO2_smooth(i+min1loc);
    end
```

```
    %plot(onewave);

[onewave_peaks,onewave_peaks_locs]=findpeaks(onewave,1:length_onewave,'MinPea
kProminence',0.00002);

    systolic_peak=onewave_peaks(1);
    systolic_loc=onewave_peaks_locs(1);

    if ((length(onewave_peaks))>1)
        diastolic_peak=onewave_peaks(2);
        diastolic_loc=onewave_peaks_locs(2);

        baseline= onewave(1);

        % Find dicrotic notch, systolic peak = Pulse wave amplitude (PWA)
        onewave_neg=onewave*(-1);
        %findpeaks(onewave_neg,1:length_onewave,'MinPeakProminence',0.00003)

[dicrotic_notch,locdicrotic]=findpeaks(onewave_neg,1:length_onewave,'MinPeakP
rominence',0.000003);
        dicrotic=dicrotic_notch*(-1);
        mag_dicrotic=dicrotic-baseline;
        mag_systolic=systolic_peak-baseline;
        mag_diastolic=diastolic_peak-baseline;

        %% Timing

        % Rise time = Systolic Phase
        RiseTime=systolic_loc*resample_factor_conc*(1/Frequency);

        % Pulse Wave Duration = Length (onewave)
        PWD = length_onewave*resample_factor_conc*(1/Frequency);

        % Pulse Propogation Time
        PPT = (diastolic_loc-
systolic_loc)*resample_factor_conc*(1/Frequency);

        % Diastolic Phase
        DiastolicPhase = (length_onewave-
systolic_loc)*resample_factor_conc*(1/Frequency);

        % Augmentation Index
        AI=mag_diastolic/mag_systolic;

        %% Integrate under signal

        %A1
        for k=1:locdicrotic
            A1(k)=onewave(k);
        end

        %A2
        A2_length=length_onewave-locdicrotic;
        for s=1:A2_length
            A2(s)=onewave(s-1+locdicrotic);
        end
```

```
        % Inflection Point Area Ratio
        A1_trap=trapz(A1);
        A2_trap=trapz(A2);

        IPA=A2_trap/A1_trap;


        %% Calculate Averages
        RiseTimeAggregate=RiseTimeAggregate+RiseTime;
        PWDAggregate = PWDAggregate + PWD;
        PPTAggregate = PPTAggregate + PPT;
        DiastolicPhaseAggregate = DiastolicPhaseAggregate+ DiastolicPhase;
        AIAggregate = AIAggregate + AI;
        IPAAggregate = IPAAggregate + IPA;
        count=count+1;

    end
    clear onewave;
end
    RiseTimeAverage=RiseTimeAggregate/count;
    PWDAverage=PWDAggregate/count;
    PPTAverage=PPTAggregate/count;
    DiastolicPhaseAverage=DiastolicPhaseAggregate/count;
    AIAverage=AIAggregate/count;
    IPAAverage=IPAAggregate/count;
end
```

2.3 c_HB.m

```
%Calculation of delta C_Hb, see Kosics paper

function  conc = c_HB(I1,I2)

I1_base=0.1;
I2_base=0.09;

alpha1_Hb_990 = 283.22 * 0.303;
alpha1_HbO2_990 = 1080 * 0.303;
alpha2_Hb_660 = 3226.56 * 0.303;
alpha2_HbO2_660 = 319.6 * 0.303;

delta1_A_990 = log(I1_base/I1);
delta2_A_660 = log(I2_base/I2);

L1_990= 0.5;
L2_660= 0.5;

num = (alpha1_HbO2_990)*(delta2_A_660)/(L2_660)-
(alpha2_HbO2_660)*(delta1_A_990)/(L1_990);
den = (alpha1_HbO2_990)*(alpha2_Hb_660) - (alpha2_HbO2_660)*(alpha1_Hb_990);

conc = num/den;

end
```

2.4 c_HBO2

```
% Calculation of delta C HbO2, see Kosics paper

function  conc = c_HBO2(I1,I2)

I1_base=0.1;
I2_base=0.09;

alpha1_Hb_990 = 283.22 * 0.303;
alpha1_HbO2_990 = 1080 * 0.303;
alpha2_Hb_660 = 3226.56 * 0.303;
alpha2_HbO2_660 = 319.6 * 0.303;

delta1_A_990= log(I1_base/I1);
delta2_A_660= log(I2_base/I2);

L1_990= 0.5;
L2_660= 0.5;

num = (alpha1_Hb_990)*(delta2_A_660)/(L2_660)-
(alpha2_Hb_660)*(delta1_A_990)/(L1_990);
den = (alpha1_Hb_990)*(alpha2_HbO2_660) - (alpha2_Hb_660)*(alpha1_HbO2_990);

conc = num/den;

end
```

2.5 derivatives.m

```
% Calculate derivatives
function [firstder,secondder]= derivatives(conc,filter)
conc_orig=conc;
conc_smooth=smooth(conc,0.01,'loess');

diffconc=diff(conc_smooth)*100;
diffconc=smooth(diffconc,filter,'loess');
firstder=diffconc;

second=diff(diffconc)*100;
second=smooth(second,0.01,'loess');
secondder=second;

end
```

2.6 feet_to_meters.m

```
function [meters]=feet_to_meters(feet,inches)
%% Convert feet to meters
%1 foot = 0.3048 meters, 1 inch = 0.0254 meters

fmeters=feet*0.3048;
imeters=inches*0.0254;

meters=fmeters+imeters;
```

2.7 heartrate_section0_custom.m

```
% Calculate heart rate and HRV when file is first loaded for custom board

function [HR,HRV] = heartrate_section0_custom(waveform)

resample_factor=1;
Frequency=104.6;

%Smooth
waveform_smooth=smooth(waveform,0.01,'loess');

length_waveform=length(waveform_smooth);
[peaks,locs]=findpeaks(waveform_smooth,1:length_waveform,'MinPeakProminence',
5000);
numpeaks=length(peaks);

%Locs accounting for resampling
locs_real=locs*resample_factor;
%Locs in terms of seconds
locs_s=locs_real*(1/Frequency);
%Locs in terms of ms
locs_ms=locs_s*1000;

for i=1:numpeaks-1
    HR_NN(i)=locs_s(i+1)-locs_s(i);
end

    num_NN=length(HR_NN);
    sum_HR_NN=sum(HR_NN);
    sec_p_beat=sum_HR_NN/num_NN;
    beats_p_sec=1/sec_p_beat;
    beats_p_min=beats_p_sec*60;

HR= beats_p_min;

%% Heart Rate Variability
for i=1:numpeaks-1
    time(i)=locs_ms(i+1)-locs_ms(i);
end

HRV=std(time);

end
```

2.8 heartrate_section0.m

```
% Calculate heart rate and HRV when file is first loaded
function [HR,HRV] = heartrate_section0(waveform)

resample_factor=1;
Frequency=500;

%Smooth
waveform_smooth=smooth(waveform,0.01,'loess');
```

```
length_waveform=length(waveform_smooth);
[peaks,locs]=findpeaks(waveform_smooth,1:length_waveform,'MinPeakProminence',
0.01);
% If you want to see plot:
% findpeaks(waveform_smooth,1:length_waveform,'MinPeakProminence',0.01);
numpeaks=length(peaks);

%Locs accounting for resampling
locs_real=locs*resample_factor;
%Locs in terms of seconds
locs_s=locs_real*(1/Frequency);
%Locs in terms of ms
locs_ms=locs_s*1000;

for i=1:numpeaks-1
    HR_NN(i)=locs_s(i+1)-locs_s(i);
end

    num_NN=length(HR_NN);
    sum_HR_NN=sum(HR_NN);
    sec_p_beat=sum_HR_NN/num_NN;
    beats_p_sec=1/sec_p_beat;
    beats_p_min=beats_p_sec*60;

HR= beats_p_min;

%% Heart Rate Variability
for i=1:numpeaks-1
    time(i)=locs_ms(i+1)-locs_ms(i);
end

HRV=std(time);

end
```

2.9 heartrate_section1_custom.m

```
% Calculate heart rate and HRV over a specified time interval for custom
% board

function [HR,HRV] = heartrate_section1_custom(waveform,starttime,endtime)

resample_factor=1;
Frequency=105

% Convert start time to position in array
starttime_pos=starttime*Frequency+1;
endtime_pos=endtime*Frequency+1;

waveform=waveform(starttime_pos:endtime_pos);

waveform_smooth=smooth(waveform,0.07,'loess');
length_waveform=length(waveform_smooth);
[peaks,locs]=findpeaks(waveform_smooth,1:length_waveform,'MinPeakProminence',
15000);
numpeaks=length(peaks);
```

```
%Locs accounting for resampling
locs_real=locs*resample_factor;
%Locs in terms of seconds
locs_s=locs_real*(1/Frequency);
%Locs in terms of ms
locs_ms=locs_s*1000;

for i=1:numpeaks-1
    HR_NN(i)=locs_s(i+1)-locs_s(i);
end

    num_NN=length(HR_NN);
    sum_HR_NN=sum(HR_NN);
    sec_p_beat=sum_HR_NN/num_NN;
    beats_p_sec=1/sec_p_beat;
    beats_p_min=beats_p_sec*60;

HR= beats_p_min;

%% Heart Rate Variability
for i=1:numpeaks-1
    time(i)=locs_ms(i+1)-locs_ms(i);
end

HRV=std(time);

end
```

2.10 heartrate_section1.m

```
% Calculate heart rate and HRV over a specified time interval
function [HR,HRV] = heartrate_section1(waveform,starttime,endtime)

resample_factor=1;
Frequency=500;

% Convert start time to position in array
starttime_pos=starttime*Frequency+1;
endtime_pos=endtime*Frequency+1;

waveform=waveform(starttime_pos:endtime_pos);

waveform_smooth=smooth(waveform,0.01,'loess');
length_waveform=length(waveform_smooth);
[peaks,locs]=findpeaks(waveform_smooth,1:length_waveform,'MinPeakProminence',
0.01);
%findpeaks(waveform_smooth,1:length_waveform,'MinPeakProminence',0.01);
numpeaks=length(peaks);

%Locs accounting for resampling
locs_real=locs*resample_factor;
%Locs in terms of seconds
locs_s=locs_real*(1/Frequency);
%Locs in terms of ms
locs_ms=locs_s*1000;
```

```
for i=1:numpeaks-1
    HR_NN(i)=locs_s(i+1)-locs_s(i);
end

    num_NN=length(HR_NN);
    sum_HR_NN=sum(HR_NN);
    sec_p_beat=sum_HR_NN/num_NN;
    beats_p_sec=1/sec_p_beat;
    beats_p_min=beats_p_sec*60;

HR= beats_p_min;

%% Heart Rate Variability
for i=1:numpeaks-1
    time(i)=locs_ms(i+1)-locs_ms(i);
end

HRV=std(time);

end
```

2.11 Readandcalculate.m

```
%% Heart of the Matter
% Senior Design Project
% Frequency is 500 ksps
function [IR, RED, concHbO2,concHb, seconds] = Readandcalculate(filename);

%% Read in Data
data = xlsread(filename);
seconds = data(:,1);
IR= data(:,3);
RED= data(:,2);

%% Concentration calculation
for i=1:length(IR);
    IRi=IR(i);
    REDi=RED(i);
    concHbO2(i)=c_HBO2(IRi,REDi);
    concHb(i)=c_HB(IRi,REDi);
end
```

2.12 stiffnessindex.m

```
%Calculate stiffness index
function [SI] = stiffnessindex(meters,PPTaverage)
SI=meters/PPTaverage;
```

2.13 SD_GUI.m

```
%% Heart of the Matter - GUI
% Electrical Engineering Senior Design Project
% Brendan Galloway, Ana Guzman, Allison Walker, Nicole Wardeberg
% Last updated May 8, 2017
```

```
% Note: any function designed for our custom board will be followed with
% _custom in the name

function varargout = SD_GUI(varargin)
% SD_GUI MATLAB code for SD_GUI.fig
%      SD_GUI, by itself, creates a new SD_GUI or raises the existing
%      singleton*.
%
%      H = SD_GUI returns the handle to a new SD_GUI or the handle to
%      the existing singleton*.
%
%      SD_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SD_GUI.M with the given input arguments.
%
%      SD_GUI('Property','Value',...) creates a new SD_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SD_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SD_GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SD_GUI

% Last Modified by GUIDE v2.5 05-May-2017 02:03:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SD_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @SD_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before SD_GUI is made visible.
function SD_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SD_GUI (see VARARGIN)
```

```
% Choose default command line output for SD_GUI
handles.output = hObject;

% Default filter value
default=0.01;
set(handles.filter_textbox,'String',num2str(default));

% Update handles structure
guidata(hObject, handles);

% Display Notre Dame logo
axes(handles.axes4);
mark=imread('ND.png');
imshow(mark);

% UIWAIT makes SD_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SD_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%% Select File
% Calculates concentrations

function filename_Callback(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of filename as text
%        str2double(get(hObject,'String')) returns contents of filename as a
double
filename_in=get(handles.filename,'String');
[IR, RED, concHbO2,concHb, seconds]=Readandcalculate(filename_in);
handles.concHbO2=concHbO2;
handles.IR=IR;
handles.RED=RED;
handles.concHb=concHb;
handles.seconds=seconds;
handles.feet=0;
handles.inches=0;
handles.frequency=500;
axes(handles.axes1);

if get(handles.checkbox1,'Value') == 1
    [HR,HRV]=heartrate_section0(IR);
    [IPAAverage,AIAverage,PPTAverage]=Analysis(concHbO2);
else
    [HR,HRV]=heartrate_section0_custom(IR);
```

```
    [IPAAverage,AIAverage,PPTAverage]=Analysis_Custom(concHbO2);
end
set(handles.HR_textbox,'String',HR);
set(handles.HRV_textbox,'String',HRV);


handles.IPA=IPAAverage;
handles.AI=AIAverage;
handles.PPT=PPTAverage;

set(handles.IPA_text,'String',IPAAverage);
set(handles.AI_text,'String',AIAverage);


guidata(hObject, handles);

function filename_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%% Buttons

function HbO2_button_Callback(hObject, eventdata, handles)
% hObject    handle to HbO2_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of HbO2_button

% Plot whichever concentration is selected
if get(handles.HbO2_button,'Value') == 1;
    set(handles.Hb_button,'Value',0);
    concHbO2=handles.concHbO2;
    seconds=handles.seconds;
    axes(handles.axes1);

    %Filter percentage
    loess_percent=str2double(get(handles.filter2_text,'String'));
    concHbO2_smooth=smooth(concHbO2,loess_percent,'loess');

    %Axis bounds, color properties
    plot(seconds,concHbO2_smooth);
    xlabel('Time (seconds)');
    ylabel('Concentration');
    set(gca, 'YTick', []);
    set(gca, 'XColorMode','manual');
    set(gca, 'XColor','w');
    set(gca, 'YColorMode','manual');
```

```
    set(gca, 'YColor','w');
    start_time=str2double(get(handles.start_textbox,'String'));
    end_time=str2double(get(handles.end_textbox,'String'));
    axis([start_time end_time -Inf Inf]);
end

function Hb_button_Callback(hObject, eventdata, handles)
% hObject    handle to Hb_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Hb_button

% Plot the concentration that is selected
if get(handles.Hb_button,'Value') == 1;
    set(handles.HbO2_button,'Value',0);
    concHb=handles.concHb;
    seconds=handles.seconds;
    axes(handles.axes1);
    loess_percent=str2double(get(handles.filter2_text,'String'));
    concHb_smooth=smooth(concHb,loess_percent,'loess');

    %Axis bounds, color properties
    plot(seconds,concHb_smooth);
    xlabel('Time (seconds)');
    ylabel('Concentration');
    set(gca, 'YTick', []);
    set(gca, 'XColorMode','manual');
    set(gca, 'XColor','w');
    set(gca, 'YColorMode','manual');
    set(gca, 'YColor','w');
    start_time=str2double(get(handles.start_textbox,'String'));
    end_time=str2double(get(handles.end_textbox,'String'));
    axis([start_time end_time -Inf Inf]);
    guidata(hObject, handles);
end

function time_button_Callback(hObject, eventdata, handles)
% hObject    handle to time_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Update the time (in seconds) displayed
starttime=str2double(get(handles.start_textbox,'String'));
endtime=str2double(get(handles.end_textbox,'String'));

% Update stored concentration
concHbO2=handles.concHbO2;

% Plot whichever concentration is selected
HbButton_status= get(handles.Hb_button,'Value');
if HbButton_status == 1;
   Hb_button_Callback(hObject, eventdata,handles);
else
   HbO2_button_Callback(hObject, eventdata, handles);
end
```

```
% Update stored IR wave
IR= handles.IR;

% Decide if you are calculating from the Eval board or custom board
if get(handles.checkbox1,'Value') == 1
    [HR,HRV]=heartrate_section1(IR,starttime,endtime);
else
    [HR,HRV]=heartrate_section1_custom(IR,starttime,endtime);
end

set(handles.HR_textbox,'String',HR);
set(handles.HRV_textbox,'String',HRV);

% Select which derivative to plot
FirstderButton_status= get(handles.firstder_button,'Value');
if FirstderButton_status == 1;
   firstder_button_Callback(hObject, eventdata,handles);
else
   secondder_button_Callback(hObject, eventdata, handles);
end
guidata(hObject, handles);

function firstder_button_Callback(hObject, eventdata, handles)
% hObject    handle to firstder_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of firstder_button

% Plot first derivative
loess_percent=str2double(get(handles.filter_textbox,'String'));
if get(handles.firstder_button,'Value') == 1;
    set(handles.secondder_button,'Value',0);
    conc=handles.concHbO2;
    [firstder,secondder] = derivatives(conc,loess_percent);
    axes(handles.axes2);
    seconds=handles.seconds;
    seconds=seconds(1:(length(seconds)-1));

    %Axis bounds
    plot(seconds,firstder);
    xlabel('Time (seconds)');
    ylabel('First Derivative of HbO_2');
    set(gca, 'YTick', []);
    set(gca, 'XColorMode','manual');
    set(gca, 'XColor','w');
    set(gca, 'YColorMode','manual');
    set(gca, 'YColor','w');
    start_time=str2double(get(handles.start_textbox,'String'));
    end_time=str2double(get(handles.end_textbox,'String'));
    axis([start_time end_time -Inf Inf]);

end

function secondder_button_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to secondder_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of secondder_button

% Plot second derivative
loess_percent=str2double(get(handles.filter_textbox,'String'));
if get(handles.secondder_button,'Value') == 1;
    set(handles.firstder_button,'Value',0);
    conc=handles.concHbO2;
    seconds=handles.seconds;
    seconds=seconds(1:(length(seconds)-2));
    [firstder,secondder] = derivatives(conc,loess_percent);
    axes(handles.axes2);

    plot(seconds,secondder);
    xlabel('Time (seconds)');
    ylabel('Second Derivative of HbO_2');
    set(gca, 'YTick', []);
    set(gca, 'XColorMode','manual');
    set(gca, 'XColor','w');
    set(gca, 'YColorMode','manual');
    set(gca, 'YColor','w');
    start_time=str2double(get(handles.start_textbox,'String'));
    end_time=str2double(get(handles.end_textbox,'String'));
    axis([start_time end_time -Inf Inf]);
end

%% Enter times, height, and age

function start_textbox_Callback(hObject, eventdata, handles)
% hObject    handle to start_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of start_textbox as text
%        str2double(get(hObject,'String')) returns contents of start_textbox
as a double

function start_textbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to start_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function end_textbox_Callback(hObject, eventdata, handles)
% hObject    handle to end_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of end_textbox as text
%        str2double(get(hObject,'String')) returns contents of end_textbox as
a double


% --- Executes during object creation, after setting all properties.
function end_textbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to end_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function age_text_Callback(hObject, eventdata, handles)
% hObject    handle to age_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of age_text as text
%        str2double(get(hObject,'String')) returns contents of age_text as a
double
age=str2double(get(handles.age_text,'String'));
handles.age=age;

% --- Executes during object creation, after setting all properties.
function age_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to age_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function feet_text_Callback(hObject, eventdata, handles)
% hObject    handle to feet_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of feet_text as text
%        str2double(get(hObject,'String')) returns contents of feet_text as a
double

%Convert to metric and update SI
feet=str2double(get(handles.feet_text,'String'));
handles.feet=feet;
```

```
inches=str2double(get(handles.inches_text,'String'));
meters=feet_to_meters(feet,inches);
handles.meters=meters;
PPTAverage=handles.PPT;
SI=stiffnessindex(meters,PPTAverage);
set(handles.SI_text,'String',SI);

% --- Executes during object creation, after setting all properties.
function feet_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to feet_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function inches_text_Callback(hObject, eventdata, handles)
% hObject    handle to inches_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inches_text as text
%        str2double(get(hObject,'String')) returns contents of inches_text as
a double

% Convert to metric and update stiffness index
feet=str2double(get(handles.feet_text,'String'));
handles.feet=feet;
inches=str2double(get(handles.inches_text,'String'));
meters=feet_to_meters(feet,inches);
handles.meters=meters;
PPTAverage=handles.PPT;
SI=stiffnessindex(meters,PPTAverage);
set(handles.SI_text,'String',SI);

% --- Executes during object creation, after setting all properties.
function inches_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inches_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% Filters

function filter_textbox_Callback(hObject, eventdata, handles)
% hObject    handle to filter_textbox (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of filter_textbox as text
%        str2double(get(hObject,'String')) returns contents of filter_textbox
as a double
if get(handles.firstder_button,'Value') == 1
    firstder_button_Callback(hObject, eventdata, handles);
else %get(handles.secondder_button_Callback(hObject,eventdata,handles) == 1
    secondder_button_Callback(hObject,eventdata,handles);
    end

% --- Executes during object creation, after setting all properties.
function filter_textbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filter_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function filter2_text_Callback(hObject, eventdata, handles)
% hObject    handle to filter2_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of filter2_text as text
%        str2double(get(hObject,'String')) returns contents of filter2_text
as a double
if get(handles.HbO2_button,'Value') == 1;
    HbO2_button_Callback(hObject,eventdata,handles);
else Hb_button_Callback(hObject, eventdata,handles);
end

% --- Executes during object creation, after setting all properties.
function filter2_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filter2_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% Textboxes that only display information

function HR_textbox_Callback(hObject, eventdata, handles)
% hObject    handle to HR_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of HR_textbox as text
%        str2double(get(hObject,'String')) returns contents of HR_textbox as
a double

% --- Executes during object creation, after setting all properties.
function HR_textbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to HR_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function HRV_textbox_Callback(hObject, eventdata, handles)
% hObject    handle to HRV_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function HRV_textbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to HRV_textbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function IPA_text_Callback(hObject, eventdata, handles)
% hObject    handle to IPA_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of IPA_text as text
%        str2double(get(hObject,'String')) returns contents of IPA_text as a
double

% --- Executes during object creation, after setting all properties.
function IPA_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to IPA_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

function AI_text_Callback(hObject, eventdata, handles)
% hObject    handle to AI_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of AI_text as text
%        str2double(get(hObject,'String')) returns contents of AI_text as a
double

% --- Executes during object creation, after setting all properties.
function AI_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AI_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function SI_text_Callback(hObject, eventdata, handles)
% hObject    handle to SI_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SI_text as text
%        str2double(get(hObject,'String')) returns contents of SI_text as a
double

% --- Executes during object creation, after setting all properties.
function SI_text_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SI_text (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% Popup Menu
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
```

```
%        contents{get(hObject,'Value')} returns selected item from popupmenu1

% Determine the selected data set.
str = get(handles.popupmenu1, 'String');
val = get(handles.popupmenu1,'Value');
% Set current data to the selected data set.
switch str{val};
case 'PPG Waveform' % User selects PPG Waveform
   axes(handles.axes3);
   ppgwave=imread('ppgwave.png');
   imshow(ppgwave);
case 'Second Derivative' % User selects Second Derivative.
   axes(handles.axes3);
   secondd=imread('2d.png');
   imshow(secondd);
case 'Inflection Point Area'
    axes(handles.axes3);
    IPApic=imread('IPA.png');
    imshow(IPApic);
end
% Save the handles structure.
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% Checkbox for info type
% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1
```