LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

-LIGO-

CALIFORNIA INSTITUTE OF TECHNOLOGY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type	DCC Number	August 13, 2015
Test Procedure	**T1500440**-v1	
Hardware Watchdog PSoC Schematic and Code Changes		
B. Abbott, R. Abbott		

Distribution of this draft:
This is an internal working note of the LIGO Laboratory

California Institute of Technology
LIGO Project – MS 18-33
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

http://www.ligo.caltech.edu/

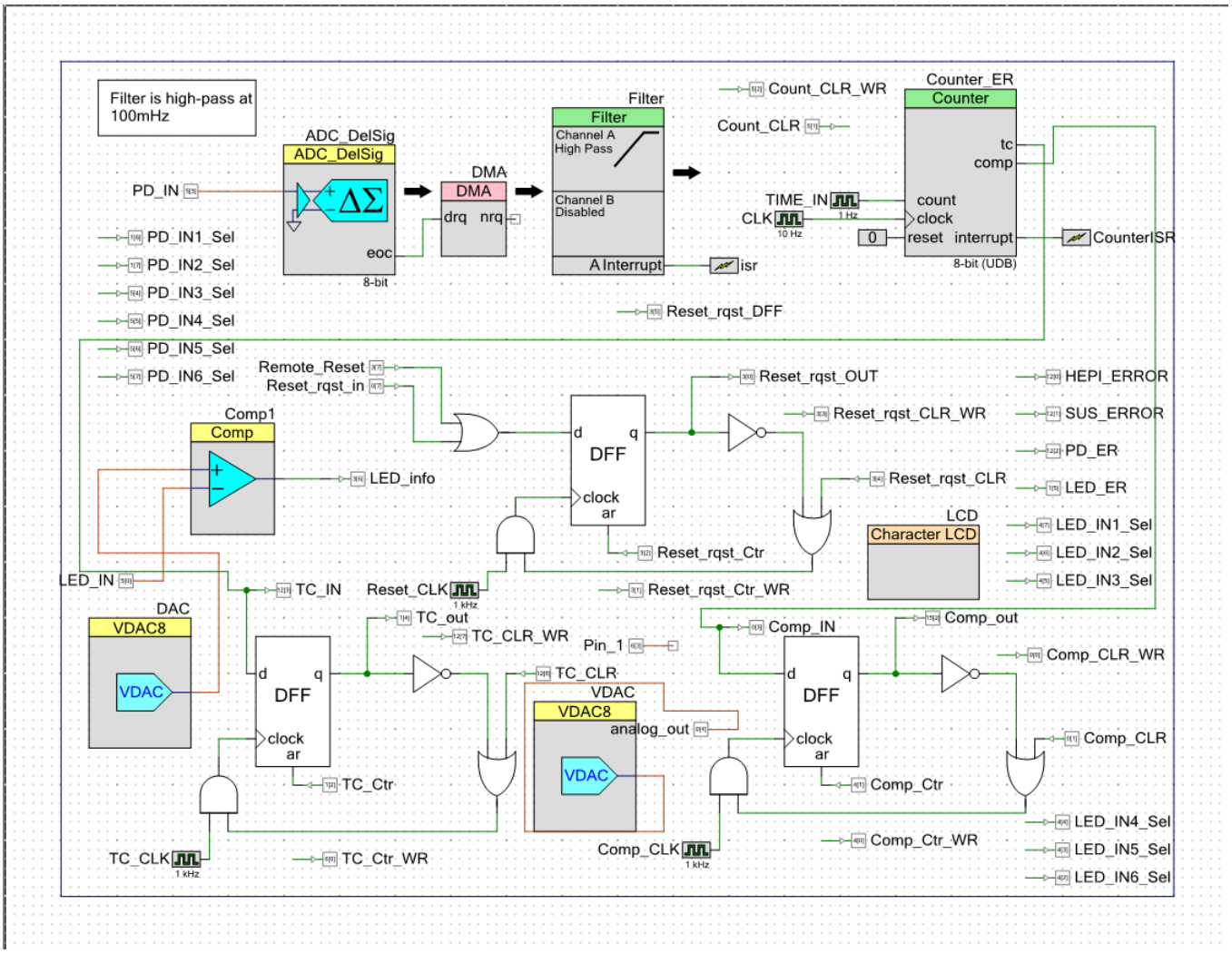
Overview:

The Cypress PSoC Creator 3.1 file for the 2014 (v1) iteration of the Hardware Watchdog Chassis (D1300642-v1) as found to be very confusing to understand and troubleshoot, which would make it unwieldy to maintain and upgrade, if needed. We set about to simplify the PSoC Top Design schematic, and comment and simplify the main.c code. This document shows the reasons for the changes, and the steps that were taken. After these changes were made, the new code was fully tested on a watchdog chassis to ensure that no functional changes occurred.

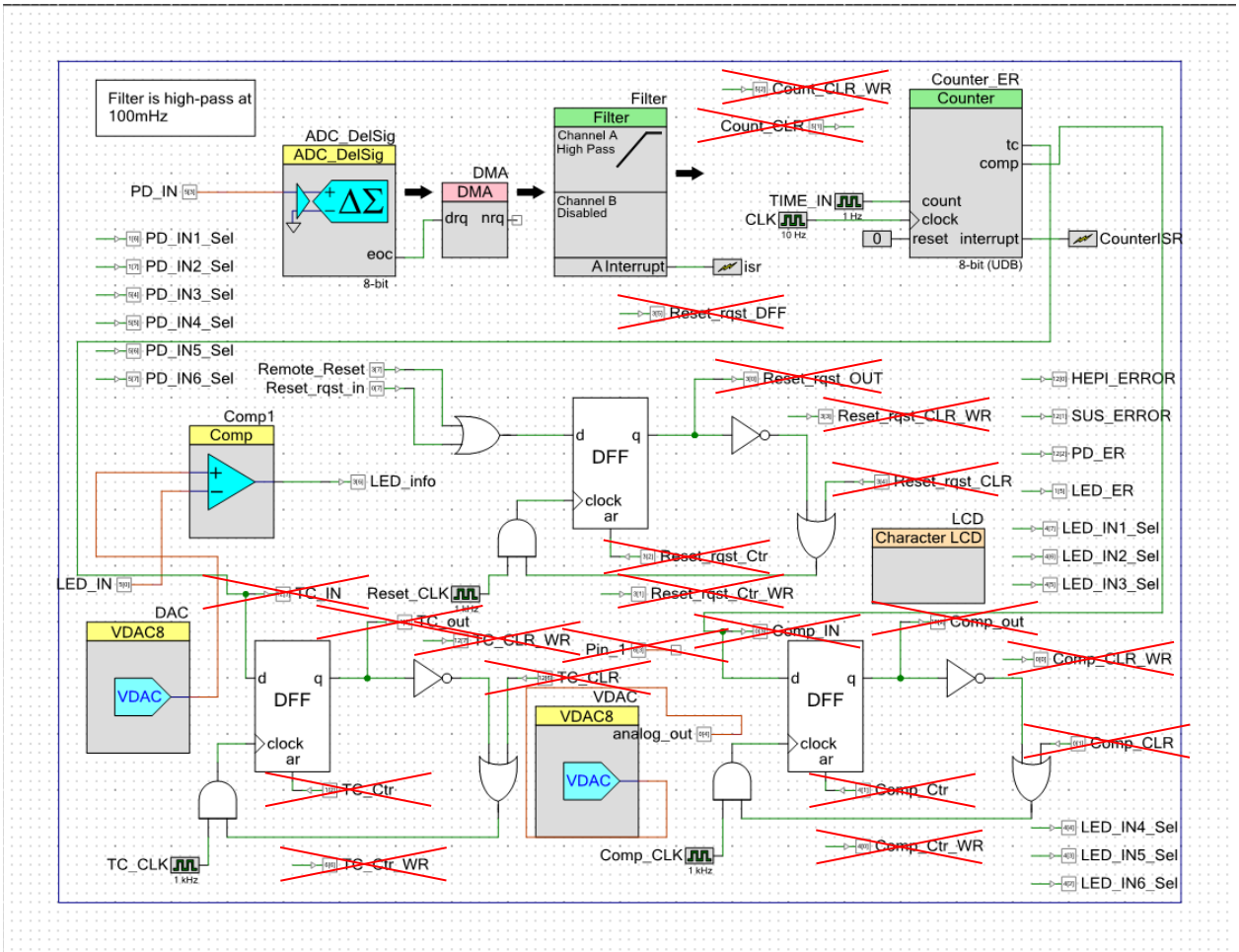
Process:

This is the schematic as it exists in the DCC as the "TopDesign.cysch" in a PSoC Creator 3.1 project file. We found it very difficult to follow and understand.

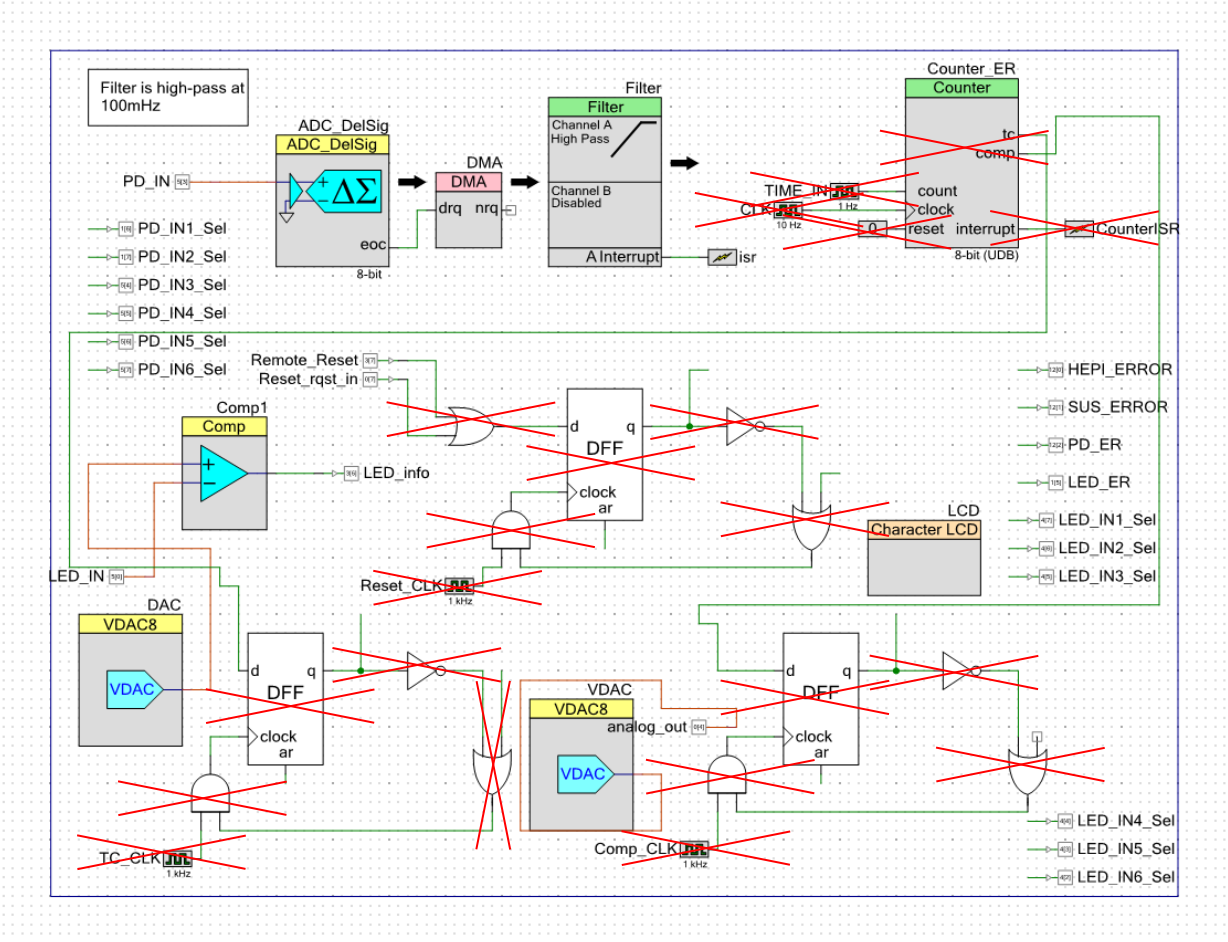
Before:



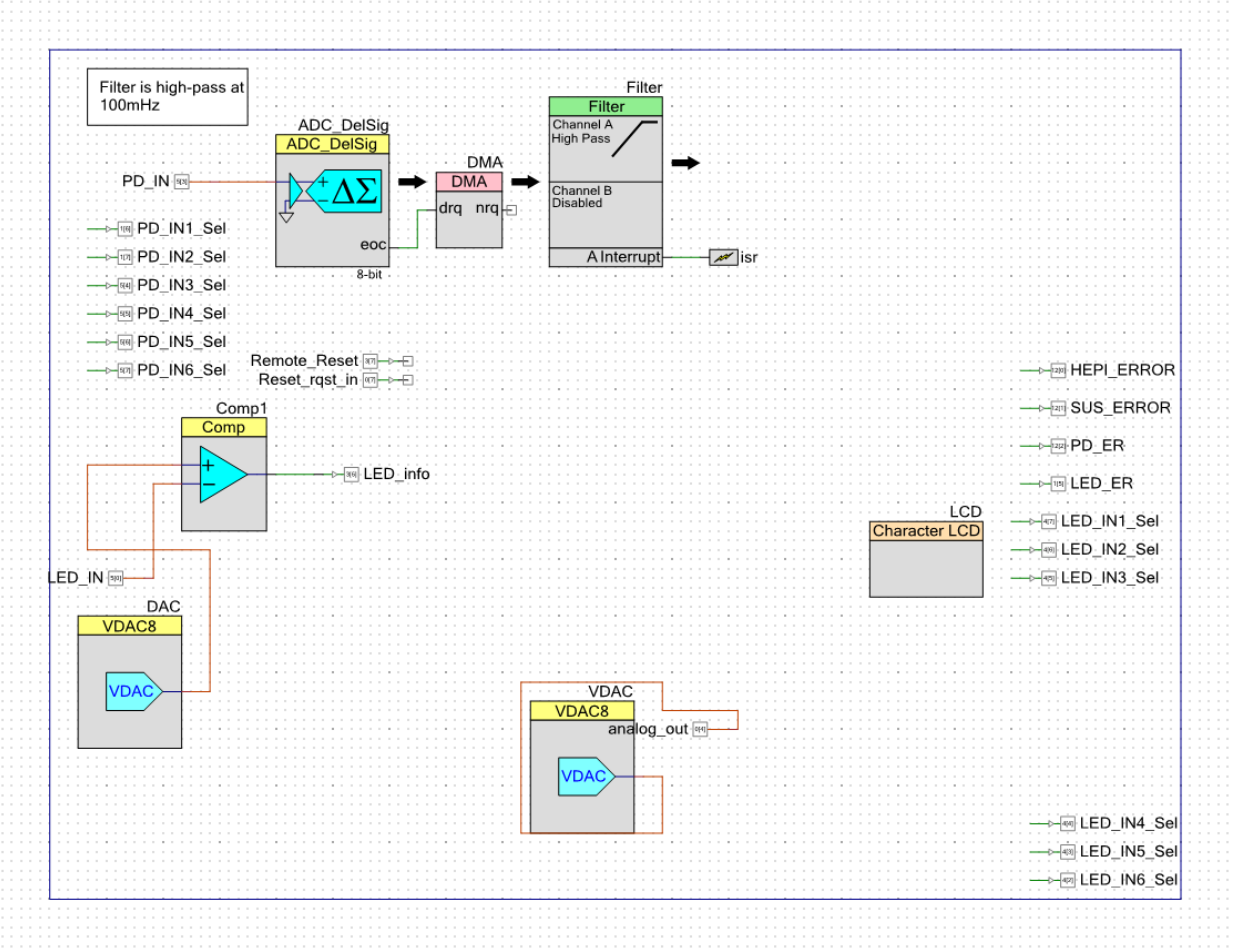
Pins were eliminated that were never used in the code, and were not wired on the hardware board at all. They were likely vestigal pins from older iterations of the design. The remaining pins were either MUX selector bits, Error output bits, the comparator output pin, the PD input signal or the Remote_Reset, and Reset_rqst_in which are both signals coming from the Hardware board, and are “OR”ed together in code, rendering the schematic “OR” symbol superfluous:



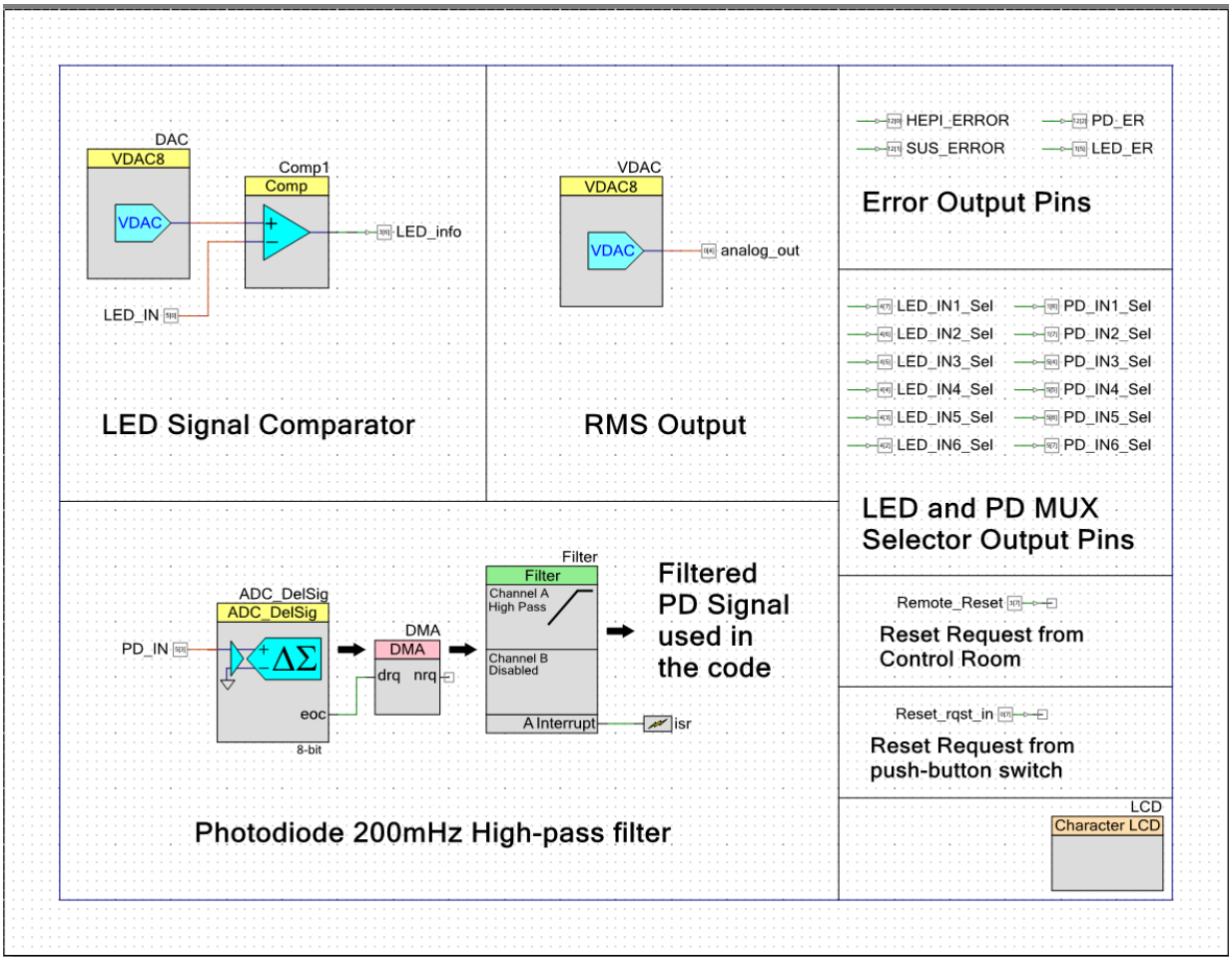
With those pins gone, it left logic gates, D-flip-flops, and a counter that all did nothing. I eliminated them, along with their individual clocks:



This left a pretty clean schematic that could be labeled and rearranged from this:



To this:


DAC
VDAC8
VDAC
Comp1
Comp
LED_info
LED_IN
LED Signal Comparator
VDAC
VDAC8
VDAC
analog_out
RMS Output
HEPI_ERROR
PD_ER
SUS_ERROR
LED_ER
Error Output Pins
LED_IN1_Sel
LED_IN2_Sel
LED_IN3_Sel
LED_IN4_Sel
LED_IN5_Sel
LED_IN6_Sel
PD_IN1_Sel
PD_IN2_Sel
PD_IN3_Sel
PD_IN4_Sel
PD_IN5_Sel
PD_IN6_Sel
LED and PD MUX
Selector Output Pins
ADC_DelSig
ADC_DelSig
PD_IN
eoc
8-bit
DMA
DMA
drq
nrq
Filter
Filter
Channel A
High Pass
Channel B
Disabled
A Interrupt
isr
Filtered
PD Signal
used in
the code
Photodiode 200mHz High-pass filter
Remote_Reset
Reset Request from
Control Room
Reset_rqst_in
Reset Request from
push-button switch
LCD
Character LCD

2. The code:

The original code was completely un-commented (except in the copy-and-pasted DMA_Config function). This made it extremely difficult to follow. There also was some vestigial code that turned on or off things that were otherwise not being used. I went through and commented the code, and removed useless code. Here is the original code:

```
/* File Name: main.c
* working code Hwardware Watchdog  2014
*
* Description:
* Sets up a complete hardware watchdog system
********************************************************************************  */

#include <device.h>
#include <project.h>
#include <math.h>
#define REQUEST_PER_BURST            1u
#define BYTES_PER_BURST                     1u
#define UPPER_SRC_ADDRESS            CYDEV_PERIPH_BASE
#define UPPER_DEST_ADDRESS           CYDEV_PERIPH_BASE
#define LED_TRUE   1
#define LED_FALSE  0
#define PD_TRUE    1
#define PD_FALSE   0
void DMA_Config(void);
void PD_RMS1(void);
void Pin_Init(void);
void Zero_Init(void);


void main()
{
    int     i, j,k,l,m, u, v, Vout, RMS_Scale;
    int     N=63, Nx, Ny, Nz;
    float   mean, max[6], min[6], s[6], rms_ratio;
    float   PD_IN[6][63];
    float   out[6][63], LED_IN[6], x[6], kf;
    float   PD_rms[6];
       int     rms_cal[6];
    int     timing_length, rms_length, minimum_length;
    int     timing_count, rms_count, which_count;
    float   timing_constant, rms_constant;
    int8    Vpd_rms_ref=30;
    float   Vled_ref=0.5;
    int16   output[6][63];
            timing_constant=1.0;
            rms_constant=1.0;
                      rms_ratio=0.55;
       X0:  Zero_Init();

            CyDelay(2000);
            u=0;
            v=0;
            l=0;
                      kf=0;
                      for (i=0; i<6; i++)
                     {
                                     rms_cal[i]=0;
                                     }
```

```
     PD_ER_Write(1);
     CyDelay(2000);
     PD_ER_Write(0);
     LED_ER_Write(1);
     CyDelay(2000);
     LED_ER_Write(0);


     PD_ER_Write(1);
     CyDelay(2000);
     PD_ER_Write(0);
     LED_ER_Write(1);
     CyDelay(2000);
     LED_ER_Write(0);


X1:              if(!Remote_Reset_Read())
                 goto xyz1;
                 else
                 goto xyz;
     xyz1:        Nx=0;
                  while(!Remote_Reset_Read())
                    {
                    Nx=Nx+1;
                    CyDelay(20);


               if(Nx>1000)

                     {
                      PD_ER_Write(0);
                      LED_ER_Write(0);
                      CyDelay(500);
                      PD_ER_Write(1);
                      LED_ER_Write(1);
                      CyDelay(500);
                      PD_ER_Write(0);
                      LED_ER_Write(0);
                      CyDelay(500);
                      PD_ER_Write(1);
                      LED_ER_Write(1);
                      CyDelay(500);
                      PD_ER_Write(0);
                      LED_ER_Write(0);
                      CyDelay(500);

                     goto xyz;
                     }
                     }
                     if((Nx>50)&(Nx<255))
                    goto read_setting;
                  if((Nx>300)&(Nx<525))
                     goto rms_writing;
                  if((Nx>575)&(Nx<775))
                     goto timing_writing;
                     goto xyz;


 read_setting:
                  rms_length=rms_constant*100;
                  timing_length=timing_constant*200;
                  PD_ER_Write(0);
                  LED_ER_Write(0);
                  CyDelay(1000);
                  if(rms_length>timing_length)
                    {
                     PD_ER_Write(1);
                     LED_ER_Write(1);
                    for(i=0; i<timing_length; i++)
                     {
```

```
                    CyDelay(20);
                    }
                    LED_ER_Write(0);
                    for(i=0; i<(rms_length-timing_length+1); i++)
                      {

                      CyDelay(20);
                      }
                    PD_ER_Write(0);
                    CyDelay(1000);
                    for (i=0; i<5; i++)
                  {
                     PD_ER_Write(1);
                     LED_ER_Write(1);
                     CyDelay(500);
                     PD_ER_Write(0);
                     LED_ER_Write(0);
                     CyDelay(500);
                  }
                     goto xyz;
                   }
                 else
                  {
                    PD_ER_Write(1);
                    LED_ER_Write(1);
                    for(i=0; i<rms_length; i++)
                    {

                    CyDelay(20);
                    }
                    PD_ER_Write(0);
                    for(i=0; i<(timing_length-rms_length+1); i++)
                      {

                       CyDelay(20);
                       }
                       LED_ER_Write(0);
                       CyDelay(1000);
                     for (i=0; i<5; i++)
                  {
                     PD_ER_Write(1);
                     LED_ER_Write(1);
                     CyDelay(500);
                     PD_ER_Write(0);
                     LED_ER_Write(0);
                     CyDelay(500);
                  }
                     goto xyz;
                 }

rms_writing:      PD_ER_Write(0);
                  LED_ER_Write(0);
                  CyDelay(1010);
                  Ny=0;
                  while(!Remote_Reset_Read())
                    {
                    Ny=Ny+1;
                    CyDelay(20);
                    if(Ny>350)
                       goto xyz;
                    }
                     if((Ny<30))
                       goto xyz;
                     else
                     {
                     rms_constant=0.01*Ny;
                     for (i=0; i<10; i++)
                  {
                     PD_ER_Write(1);
                     LED_ER_Write(1);
```

```
                        CyDelay(500);
                        PD_ER_Write(0);
                        LED_ER_Write(0);
                        CyDelay(500);
                    }
                        goto xyz;
                        }

    timing_writing:    PD_ER_Write(0);
                       LED_ER_Write(0);
                       CyDelay(1010);
                       Nz=0;
                       while(!Remote_Reset_Read())
                       {
                       Nz=Nz+1;
                       CyDelay(20);
                       if(Nz>350)
                       goto xyz;
                       }
                        if(Nz<18)
                          goto xyz;
                        else
                        {
                        timing_constant=0.005*Nz;
                        for (i=0; i<15; i++)
                    {
                        PD_ER_Write(1);
                        LED_ER_Write(1);
                        CyDelay(500);
                        PD_ER_Write(0);
                        LED_ER_Write(0);
                        CyDelay(500);
                    }
                        goto xyz;
                        }

xyz:      for (j=0; j<N; j++)
          {
           for (i=0; i<6; i++)
             {
               CyDelay(1);
               LED_IN1_Sel_Write(0);
               LED_IN2_Sel_Write(0);
               LED_IN3_Sel_Write(0);
               LED_IN4_Sel_Write(0);
               LED_IN5_Sel_Write(0);
               LED_IN6_Sel_Write(0);
               PD_IN1_Sel_Write(0);
               PD_IN2_Sel_Write(0);
               PD_IN3_Sel_Write(0);
               PD_IN4_Sel_Write(0);
               PD_IN5_Sel_Write(0);
               PD_IN6_Sel_Write(0);
               CyDelay(1);
                 if(i==0)
                 PD_IN1_Sel_Write(1);
                 if(i==1)
                 PD_IN2_Sel_Write(1);
                 if(i==2)
                 PD_IN3_Sel_Write(1);
                 if(i==3)
                 PD_IN4_Sel_Write(1);
                 if(i==4)
                 PD_IN5_Sel_Write(1);
                 if(i==5)
                 PD_IN6_Sel_Write(1);
                 CyDelay(1);
                     ADC_DelSig_IRQ_Start();
                     ADC_DelSig_Start();
                     ADC_DelSig_StartConvert();
                 Filter_Start();
```

```
                if(ADC_DelSig_IsEndConversion(ADC_DelSig_WAIT_FOR_RESULT))
                  {
                   out[i][j]=ADC_DelSig_GetResult16();
                  }
                  CyDelay(1);
               }
        }
              for (i=0; i<6; i++)
                {
                 PD_rms[i]=0.0;
                 mean=0.0;
           for(j=1; j<N; j++)
                  {
                   mean=mean+out[i][j];
                  }
                  mean=mean/(N-1);
                 for(j=1; j<N; j++)
                  {
                   PD_rms[i]=PD_rms[i]+(out[i][j]-mean)*(out[i][j]-mean);
                  }
                  PD_rms[i]=PD_rms[i]/(N-1);
                  PD_rms[i]=sqrt(PD_rms[i]);
                                  rms_cal[i]=PD_rms[i];
                  PD_rms[i]=PD_rms[i]/rms_ratio;
                  PD_rms[i]=PD_rms[i]/rms_constant;
                 }


                                  VDAC_Start();
                                  if (PD_rms[0]>(0.2*Vpd_rms_ref))
                 {
                                  rms_cal[0]=0.69*rms_cal[0]+0.57*kf;
                                  kf=rms_cal[0];
                  VDAC_SetValue(rms_cal[0]);
                                  CyDelay(10);
                                 }
                   else
                 {
                                  VDAC_SetValue(3);
                                  CyDelay(10);
                 }


         m=0;
      for (i=0; i<6; i++)
         {

          if (PD_rms[i]>Vpd_rms_ref)
            {
            m=1;

            }
         else
             m=m;
      }
            if(m==1)
            {
              PD_ER_Write(1);

              CyDelay(1);
            }
            else
            {
              PD_ER_Write(0);

              CyDelay(1);
            }

X2:     for (i=0; i<6; i++)
```

```
            {
             Pin_Init();

             DAC_Start();
             Comp1_Start();
             CyDelay(1);
             if(i==0)
             LED_IN1_Sel_Write(1);
             if(i==1)
             LED_IN2_Sel_Write(1);
             if(i==2)
             LED_IN3_Sel_Write(1);
             if(i==3)
             LED_IN4_Sel_Write(1);
             if(i==4)
             LED_IN5_Sel_Write(1);
             if(i==5)
             LED_IN6_Sel_Write(1);
             CyDelay(1);


          if (Comp1_GetCompare())
             {
                k=1;
             }
             else
             {
                 k=k;
             }
            }
              if (k==1)
               {
                if(l==0)
                {
                 LED_ER_Write(0);
                 }
                 else
               {  LED_ER_Write(1); }

                }
              else
                {
                 LED_ER_Write(0);

                }

X3:           if((m==1) || (k==1))
                 {
                   l=l+1;
                   m=0;
                   k=0;
                 }
              else
                {
                   l=0;
                   Count_CLR_WR_Write(0);
                   Counter_ER_Stop();
                }


X4:           if(l>(687*timing_constant))
                 {
                   HEPI_ERROR_Write(1);

                   goto X5;
                 }
              else
                 {
                   HEPI_ERROR_Write(0);
                   goto X1;
                 }
```

```
X5:           if (l>(1330*timing_constant))
                 { SUS_ERROR_Write(1);
                   l=3000;

                  }
              else
                 {
                  SUS_ERROR_Write(0);

                 }

X6:       for (j=0; j<N; j++)
          {
           for (i=0; i<6; i++)
             {

              CyDelay(1);
              LED_IN1_Sel_Write(0);
              LED_IN2_Sel_Write(0);
              LED_IN3_Sel_Write(0);
              LED_IN4_Sel_Write(0);
              LED_IN5_Sel_Write(0);
              LED_IN6_Sel_Write(0);
              PD_IN1_Sel_Write(0);
              PD_IN2_Sel_Write(0);
              PD_IN3_Sel_Write(0);
              PD_IN4_Sel_Write(0);
              PD_IN5_Sel_Write(0);
              PD_IN6_Sel_Write(0);
              CyDelay(1);
                if(i==0)
                PD_IN1_Sel_Write(1);
                if(i==1)
                PD_IN2_Sel_Write(1);
                if(i==2)
                PD_IN3_Sel_Write(1);
                if(i==3)
                PD_IN4_Sel_Write(1);
                if(i==4)
                PD_IN5_Sel_Write(1);
                if(i==5)
                PD_IN6_Sel_Write(1);
                CyDelay(1);

                ADC_DelSig_IRQ_Start();
                    ADC_DelSig_Start();
                    ADC_DelSig_StartConvert();
                Filter_Start();
                DMA_Config();
                CYGlobalIntEnable;
                if(ADC_DelSig_IsEndConversion(ADC_DelSig_WAIT_FOR_RESULT))
                  {
                   out[i][j]=ADC_DelSig_GetResult16();
                  }

                  CyDelay(1);

              }
          }
               for (i=0; i<6; i++)
                 {
                  PD_rms[i]=0.0;
                  mean=0.0;
                  for(j=1; j<N; j++)
                   {
                    mean=mean+out[i][j];
                   }
                   mean=mean/(N-1);
                  for(j=1; j<N; j++)
                   {
```

```
                PD_rms[i]=PD_rms[i]+(out[i][j]-mean)*(out[i][j]-mean);
               }
               PD_rms[i]=PD_rms[i]/(N-1);
               PD_rms[i]=sqrt(PD_rms[i]);
                                rms_cal[i]=PD_rms[i];
                                PD_rms[i]=PD_rms[i]/rms_ratio;
                                PD_rms[i]=PD_rms[i]/rms_constant;
             }

                                VDAC_Start();
                                if (PD_rms[0]>(0.2*Vpd_rms_ref))
             {

                                rms_cal[0]=0.69*rms_cal[0]+0.57*kf;
                                kf=rms_cal[0];
              VDAC_SetValue(rms_cal[0]);
                                CyDelay(10);
                               }
              else
             {
                                VDAC_SetValue(3);
                                CyDelay(10);
             }



m=0;
for (i=0; i<6; i++)
   {
     if (PD_rms[i]>Vpd_rms_ref)
                      {
                      m=1;
                     }
                else
                      m=m;
             }

          for (i=0; i<6; i++)
           {
             CyDelay(1);
             Pin_Init();
             CyDelay(5);
             if(i==0)
             LED_IN1_Sel_Write(1);
             if(i==1)
             LED_IN2_Sel_Write(1);
             if(i==2)
             LED_IN3_Sel_Write(1);
             if(i==3)
             LED_IN4_Sel_Write(1);
             if(i==4)
             LED_IN5_Sel_Write(1);
             if(i==5)
             LED_IN6_Sel_Write(1);
                DAC_Start();
                Comp1_Start();
                CyDelay(1);
       if (LED_info_Read())
     if (Comp1_GetCompare())
                   k=1;
                else
                   k=k;
            }
                if((m==1) || (k==1))
            {
              l=l+1;

            }
               else
            {
                l=l;
```

```
                    }
                       if (m==0)
                           PD_ER_Write(0);
                       if (m==1)
                           PD_ER_Write(1);
                       if (k==0)
                           LED_ER_Write(0);
                       if (k==1)
                           LED_ER_Write(1);


                       if(((!Remote_Reset_Read()) || Reset_rqst_in_Read()) & (m==0) & (k==0))

                      {

                       CyDelay(2000);
                            l=0;
                            goto X0;
                      }

                       else
                       {
                       m=0;
                       k=0;
                       Reset_rqst_CLR_WR_Write(1);
                       Reset_rqst_CLR_WR_Write(0);
                       goto X5;
                      }


 }

                  void Pin_Init(void)
                  {
                   CyDelay(10);
                   LED_IN1_Sel_Write(0);
                   LED_IN2_Sel_Write(0);
                   LED_IN3_Sel_Write(0);
                   LED_IN4_Sel_Write(0);
                   LED_IN5_Sel_Write(0);
                   LED_IN6_Sel_Write(0);
                   PD_IN1_Sel_Write(0);
                   PD_IN2_Sel_Write(0);
                   PD_IN3_Sel_Write(0);
                   PD_IN4_Sel_Write(0);
                   PD_IN5_Sel_Write(0);
                   PD_IN6_Sel_Write(0);
                   CyDelay(1);
                 }

/*******************************************************************************
* Function Name: DMA_Config
********************************************************************************
* Summary:
*  Initializes and sets up DMA for use
*
*******************************************************************************/
void DMA_Config(void)
{
        /* Declare variable to hold the handle for DMA channel */
        uint8 channelHandle;
        /* Declare DMA Transaction Descriptor for memory transfer into
         * High Pass Filter Channel.
         */
        uint8 tdChanA;
        /* Configure the DMA to Transfer the data in 1 burst with individual trigger
         * for each burst.
         */
        channelHandle = DMA_DmaInitialize(BYTES_PER_BURST, REQUEST_PER_BURST,
                                          HI16(UPPER_SRC_ADDRESS), HI16(UPPER_DEST_ADDRESS));
```

```
        tdChanA = CyDmaTdAllocate();

        CyDmaTdSetConfiguration(tdChanA, 1, DMA_INVALID_TD, 0);
        /* Set the source address as ADC_DelSig and the destination as
         * Filter Channel A.   */
        CyDmaTdSetAddress(tdChanA, LO16((uint32)ADC_DelSig_DEC_SAMP_PTR),
LO16((uint32)Filter_STAGEAH_PTR));
        /* Set tdChanA to be the initial TD associated with channelHandle */
        CyDmaChSetInitialTd(channelHandle, tdChanA);

        /* Enable the DMA channel represented by channelHandle and preserve the TD */
        CyDmaChEnable(channelHandle, 1);
}

   void Zero_Init(void)
    {
       HEPI_ERROR_Write(0);
       SUS_ERROR_Write(0);
       PD_ER_Write(0);
       LED_ER_Write(0);
       Comp_CLK_Start();
       CLK_Start();
       Reset_CLK_Start();
       TC_CLK_Start();
       TIME_IN_Start();
       Count_CLR_WR_Write(0);
       TC_Ctr_WR_Write(1);
       Comp_Ctr_WR_Write(1);
       Reset_rqst_Ctr_WR_Write(1);
       TC_Ctr_WR_Write(0);
       Comp_Ctr_WR_Write(0);
       Reset_rqst_Ctr_WR_Write(0);
       Reset_rqst_CLR_WR_Write(1);
       TC_CLR_WR_Write(1);
       Comp_CLR_WR_Write(1);
       CyDelay(1);
       Reset_rqst_CLR_WR_Write(0);
       Comp_CLR_WR_Write(0);
       TC_CLR_WR_Write(0);
    }
```

Here is the new, commented and cleaned code:

```
/* File Name: main.c
* working code for Hardware Watchdog
*E1500315 (Set version number on line 42)
* Description:
* Sets up a complete hardware watchdog system
********************************************************************************  */

#include <device.h>
#include <project.h>
#include <math.h>
#define REQUEST_PER_BURST            1u
#define BYTES_PER_BURST                     1u
#define UPPER_SRC_ADDRESS            CYDEV_PERIPH_BASE
#define UPPER_DEST_ADDRESS           CYDEV_PERIPH_BASE
#define LED_TRUE   1
#define LED_FALSE  0
#define PD_TRUE    1
#define PD_FALSE   0
void DMA_Config(void);
void PD_RMS1(void);
void Pin_Init(void);
void Zero_Init(void);


void main()
{
    int     i, j,k,l,m, u, v, version;
    int     N=63, Nx, Ny, Nz;
    float   mean, max[6], min[6], s[6], rms_ratio;
    float   PD_IN[6][63];
    float   out[6][63], LED_IN[6], x[6], kf;
    float   PD_rms[6];
       int     rms_cal[6];
    int     timing_length, rms_length;
    float   timing_constant, rms_constant;
    int8    Vpd_rms_ref=30;
    float   Vled_ref=0.5;
    int16   output[6][63];
            timing_constant=1.0;
            rms_constant=1.0;
                      rms_ratio=0.55;
            version=2; //: Enter new code version number here after any revisions://
            LCD_Start();
    X0:  Zero_Init(); //:Calls the "Zero Init" function on line 644 to set all the output
                        selector bits to "0"://

            CyDelay(2000); //:Pause for 2 seconds://
            u=0;
            v=0;
            l=0;
           kf=0;
                     for (i=0; i<6; i++)//:Initializes all of the rms calculation variables to
                                              "0"://
                    {
                                     rms_cal[i]=0;
                                     }
            for(i=0; i<version; i++)//: Flashes the front panel "LED" and "PD" lights to indicate
                                       the number of current code version://
            {
            PD_ER_Write(1);
            LED_ER_Write(1);
            CyDelay(1000);
```

```
            PD_ER_Write(0);
            LED_ER_Write(0);
            CyDelay(1000);

            }


        X1:             if(!Remote_Reset_Read()) //:If, upon power up, the control room is
                                                    sending a "Reset" request, goto xyz1://
                        goto xyz1;
                        else//: Otherwise, go to the main code, xyz://
                        goto xyz;
            xyz1:        Nx=0;
                         while(!Remote_Reset_Read()) //:Count the duration of the control room's
                                                      "Reset" request://
                        {
                           Nx=Nx+1;
                           CyDelay(20);


                      if(Nx>1000) //:If the "Reset" Request duration is greater than 1000 loops of
                                                   the xyz1 code, flash error lights, and goto xyz://

//:xyz is the main code area://
                              {
                               PD_ER_Write(0);
                               LED_ER_Write(0);
                               CyDelay(500);
                               PD_ER_Write(1);
                               LED_ER_Write(1);
                               CyDelay(500);
                               PD_ER_Write(0);
                               LED_ER_Write(0);
                               CyDelay(500);
                               PD_ER_Write(1);
                               LED_ER_Write(1);
                               CyDelay(500);
                               PD_ER_Write(0);
                               LED_ER_Write(0);
                               CyDelay(500);

                              goto xyz;
                              }   //:End the if(Nx>1000) loop://
                           }   //:End the while(!Remote_Reset_Read) loop://
                              if((Nx>50)&(Nx<255)) //:If the Nx timer stopped between 50 and 255
                                                    loops, goto read_setting subroutine://
                             goto read_setting;
                          if((Nx>300)&(Nx<525)) //:If the Nx timer stopped between 300 and 525
                                                    loops, goto rms_writingting subroutine://
                             goto rms_writing;
                          if((Nx>575)&(Nx<775)) //:If the Nx timer stopped between 575 and 775
                                                    loops, goto timing_writing subroutine://
                             goto timing_writing;
                             goto xyz;


            read_setting:   //:This bit of code allows the control room to glean the settings of the
                                  current timing length and rms length://
                            //:By booting, and giving a Control Room "Reset" command that lasts
                                  between 1 second (50*20ms per loop) and ://
                            //: 5.1 seconds (255 * 20ms), you get a readback of the timing constant
                                  and rms_constant://
                            rms_length=rms_constant*100;
                            timing_length=timing_constant*200;
                            PD_ER_Write(0);  //:Initialize the PD_ER and LED_ER bits to zero://
                            LED_ER_Write(0);
                            CyDelay(1000);
                            if(rms_length>timing_length)//:If the rms_constant is bigger than twice
                                                          the timing_constant, turn on the PD_ER and
                                                          LED_ER bits://
                              {
```

```
                    PD_ER_Write(1);
                    LED_ER_Write(1);
                   for(i=0; i<timing_length; i++) //:Pause for 20ms*timing_length://
                    {

                    CyDelay(20);
                    }
                    LED_ER_Write(0); //:turn off the LED_ER bit, then pause for 20ms times
                                       the difference between the rms_length and the
                                       timing_length://
                    for(i=0; i<(rms_length-timing_length+1); i++)
                      {

                      CyDelay(20);
                      }
                    PD_ER_Write(0);//: Then turn off the PD_ER bit://
                    CyDelay(1000); //: Pause for a second://
                    for (i=0; i<5; i++)  //:Then blink the two bits on and off 5 times to
                                           signify completion://
                  {
                     PD_ER_Write(1);
                     LED_ER_Write(1);
                     CyDelay(500);
                     PD_ER_Write(0);
                     LED_ER_Write(0);
                     CyDelay(500);
                  }
                     goto xyz;  //:go to the main code://
                   }
                 else//:If the rms_constant is not bigger than twice the timing_constant,
                       turn on the PD_ER and LED_ER bits://
                  {
                    PD_ER_Write(1);
                    LED_ER_Write(1);
                    for(i=0; i<rms_length; i++)//:Pause for 20ms*rms_length://
                    {

                    CyDelay(20);
                    }
                    PD_ER_Write(0);//: Turn off the PD_ER bit (instead of the LED_ER bit
                                       above)://
                    for(i=0; i<(timing_length-rms_length+1); i++)//:pause for 20ms times
                                                                   the difference between
                                                                   the timing_length and
                                                                   the rms_length ://
                      {

                      CyDelay(20);
                      }
                      LED_ER_Write(0);//: Then turn off the LED_ER bit://
                      CyDelay(1000); //: Pause for a second://
                     for (i=0; i<5; i++)//:Then blink the two bits on and off 5 times to
                                           signify completion://
                  {
                     PD_ER_Write(1);
                     LED_ER_Write(1);
                     CyDelay(500);
                     PD_ER_Write(0);
                     LED_ER_Write(0);
                     CyDelay(500);
                  }
                     goto xyz; //:go to the main code://
                 }

rms_writing:      PD_ER_Write(0);  //:Initialize the PD_ER and LED_ER bits to zero://
                  LED_ER_Write(0);
                  CyDelay(1010); //: Pause for just over a second, assumedly waiting for
                                     another "Reset" command from the control room://
                  Ny=0;
                  while(!Remote_Reset_Read())//:Count the duration of the control room's
                                               second "Reset" request://
```

```
                        {
                        Ny=Ny+1;
                        CyDelay(20);
                        if(Ny>350)//:If the second "Reset" Request duration is greater than

                                  350 loops of the rms_writing code, goto the main code://
                           goto xyz;
                        }
                         if((Ny<30))//:If the second "Reset" Request duration is less than 30
                                    loops of the rms_writing code, goto the main code://
                           goto xyz;
                         else //: If it's between these two duration times, use its duration
                                  to set the new rms_constant value://
                         {
                         rms_constant=0.01*Ny; //:the new value of rms_constant is the 0.01
                                               times the "Reset" command duration divided
                                               by 20ms CyDelay://
                         for (i=0; i<10; i++)//:Then blink the two bits on and off 10 times to
                                             signify completion://
                      {
                         PD_ER_Write(1);
                         LED_ER_Write(1);
                         CyDelay(500);
                         PD_ER_Write(0);
                         LED_ER_Write(0);
                         CyDelay(500);
                      }
                         goto xyz;//:go to the main code://
                         }

timing_writing:       PD_ER_Write(0);  //:Initialize the PD_ER and LED_ER bits to zero://
                      LED_ER_Write(0);
                      CyDelay(1010);  //: Pause for just over a second, assumedly waiting
                                        for another "Reset" command from the control room://
                      Nz=0;
                      while(!Remote_Reset_Read())//:Count the duration of the control room's
                                                   second "Reset" request://
                      {
                      Nz=Nz+1;
                      CyDelay(20);
                      if(Nz>350)//:If the second "Reset" Request duration is greater than
                                  350 loops of the rms_writing code, goto the main code://
                      goto xyz;
                      }
                       if(Nz<18)//:If the second "Reset" Request duration is less than 18
                                  loops of the rms_writing code, goto the main code://
                         goto xyz;
                       else //: If it's between these two duration times, use its duration
                                to set the new timing_constant value://
                       {
                       timing_constant=0.005*Nz;//:the new value of timing_constant is the
                                                  0.005 times the "Reset" command duration
                                                  divided by 20ms CyDelay://
                       for (i=0; i<15; i++)//:Then blink the two bits on and off 15 times to
                                           signify completion://
                      {
                         PD_ER_Write(1);
                         LED_ER_Write(1);
                         CyDelay(500);
                         PD_ER_Write(0);
                         LED_ER_Write(0);
                         CyDelay(500);
                      }
                         goto xyz;//:go to the main code://
                         }
//:Main Code://
  xyz:      for (j=0; j<N; j++) //:Loop N times (N currently=63)://
            {
             for (i=0; i<6; i++) //:Loop 6 times, and set the value of the correct ADC PD Input
                                 into the "out" array out[i][j]://
               {
```

```
      CyDelay(1);
      LED_IN1_Sel_Write(0);
      LED_IN2_Sel_Write(0);
      LED_IN3_Sel_Write(0);
      LED_IN4_Sel_Write(0);
      LED_IN5_Sel_Write(0);
      LED_IN6_Sel_Write(0);
      PD_IN1_Sel_Write(0);
      PD_IN2_Sel_Write(0);
      PD_IN3_Sel_Write(0);
      PD_IN4_Sel_Write(0);
      PD_IN5_Sel_Write(0);
      PD_IN6_Sel_Write(0);
      CyDelay(1);
        if(i==0)
        PD_IN1_Sel_Write(1);
        if(i==1)
        PD_IN2_Sel_Write(1);
        if(i==2)
        PD_IN3_Sel_Write(1);
        if(i==3)
        PD_IN4_Sel_Write(1);
        if(i==4)
        PD_IN5_Sel_Write(1);
        if(i==5)
        PD_IN6_Sel_Write(1);
        CyDelay(1);
            ADC_DelSig_IRQ_Start();//:Enables interrupts at the end of conversion://
            ADC_DelSig_Start();//:Start the ADC://
            ADC_DelSig_StartConvert();//:Starts conversion://
        Filter_Start();//:Starts the filter://
        DMA_Config();//:Configure the DMA by calling the DMA_Config function://
        CYGlobalIntEnable;//:Enable Global Interrupts://
        if(ADC_DelSig_IsEndConversion(ADC_DelSig_WAIT_FOR_RESULT))//:If the ADC is
                                                                   ready with a
                                                                    result://
          {
           out[i][j]=ADC_DelSig_GetResult16();//:Set the value of the correct ADC PD
                                               Input into the "out" array
                                               out[i][j]://

          }
          CyDelay(1);
       }
}
      for (i=0; i<6; i++)//: initialize each of the 6 PD_rms and mean://
        {
         PD_rms[i]=0.0;
         mean=0.0;
   for(j=1; j<N; j++)//: calculate the mean value of all of the PD inputs://
          {
           mean=mean+out[i][j];
          }
          mean=mean/(N-1);
         for(j=1; j<N; j++)//: calculate the rms value of each of the PD inputs://
          {
           PD_rms[i]=PD_rms[i]+(out[i][j]-mean)*(out[i][j]-mean);
          }
          PD_rms[i]=PD_rms[i]/(N-1);
          PD_rms[i]=sqrt(PD_rms[i]);
                              rms_cal[i]=PD_rms[i];
          PD_rms[i]=PD_rms[i]/rms_ratio;
          PD_rms[i]=PD_rms[i]/rms_constant;
         }


                              VDAC_Start();
                              if (PD_rms[0]>(0.2*Vpd_rms_ref)) //:If the rms voltage
                                                                of the first PD is above 6
                                                                (=0.2*Vpd_rms_ref(currently
                                                                30)):://
```

```
                {
                                rms_cal[0]=0.69*rms_cal[0]+0.57*kf;//:set the rms_cal of
                                                the first PD to 0.69 times its
                                           old value, plus0.57*kf (initially 0)://
                                kf=rms_cal[0];//: change kf to this new rms_cal value
                                                for the next go-round://
                 VDAC_SetValue(rms_cal[0]); //:send the value of the first pd rms
                                           calculation out to be monitored on the back
                                           panel://
                                CyDelay(10);
                               }
                 else
                {
                                VDAC_SetValue(3);//:If the rms voltage of the first PD
                                                is not above 6, send out a DAC value
                                                of 3 to the back panel://
                                CyDelay(10);
                }


          m=0;
      for (i=0; i<6; i++)//:Check if any of the PD_rms values are greater than the reference
                          voltage://
         {

           if (PD_rms[i]>Vpd_rms_ref)
             {
             m=1;//:If any are greater, set the "m" error bit to one://

             }
          else
              m=m;//:otherwise, leave "m" alone (awkward coding)://
         }
             if(m==1)//:If the error bit is set, write out the PD _ER bit://
             {
               PD_ER_Write(1);

               CyDelay(1);
             }
             else//:Otherwise, don't://
             {
               PD_ER_Write(0);

               CyDelay(1);
             }

X2:     for (i=0; i<6; i++)//:Loop 6 times, and compare the value of the correct ADC LED Input
                            to the DAC value of 2640mV from "DAC"://
             {
              Pin_Init();

              DAC_Start();
              Comp1_Start();
              CyDelay(1);
              if(i==0)
              LED_IN1_Sel_Write(1);
              if(i==1)
              LED_IN2_Sel_Write(1);
              if(i==2)
              LED_IN3_Sel_Write(1);
              if(i==3)
              LED_IN4_Sel_Write(1);
              if(i==4)
              LED_IN5_Sel_Write(1);
              if(i==5)
              LED_IN6_Sel_Write(1);
              CyDelay(1);


           if (Comp1_GetCompare())//:If any are less than the DAC value, set the "k" error bit to
```

```
                                one://
                {
                   k=1;
                }
                else//:otherwise, leave "k" alone (awkward coding)://
                {
                    k=k;
                }
               }
                 if (k==1)//:If the error bit is set, and the error timer is not zero, write out
                           the LED _ER bit://
                  {
                   if(l==0)
                   {
                    LED_ER_Write(0);
                    }
                    else
                  {  LED_ER_Write(1); }

                   }
                 else
                   {
                    LED_ER_Write(0);//:Otherwise, don't://

                   }

  X3:            if((m==1) || (k==1))//:If the PD rms error, OR the LED error have been thrown,
                                       start the "l" counter, and reset the "m" and "k"
                                       errors://
                   {
                     l=l+1;
                     m=0;
                     k=0;
                   }
                 else//:Otherwise, reset "l" to 0://
                   {
                      l=0;
                   }

  X4:            if(l>(687*timing_constant))//:if "l" has become bigger that 687 loops, times
                                             the timing_constant (currently 1), throw the HEPI
                                             error.  This takes about 20 min.)://
                   {
                     HEPI_ERROR_Write(1);

                     goto X5;//:Then proceed to the next step://
                   }
                 else//:If "l" is less than 687 loops, reset the HEPI error, and go back to the
                        very beginning://
                   {
                     HEPI_ERROR_Write(0);
                     goto X1;
                   }
  X5:
                  if (l>(1330*timing_constant))//:if "l" has become bigger than 1330 loops of
                                                 the code below, times the timing_constant
                                                 (currently 1),throw the SUS error, and set
                                                 "l" to 3000.  This takes about 40 min.)://
                    {
                     SUS_ERROR_Write(1);
                     l=3000;

                    }
                 else
                    {
                     SUS_ERROR_Write(0);

                    }
//:the following (almost identical) code runs during a HEPI fault or SUS fault://
  X6:       for (j=0; j<N; j++)//:Loop N times (N currently=63)://
            {
```

```
 for (i=0; i<6; i++)//:Loop 6 times, and set the value of the correct ADC PD Input
                        into the "out" array out[i][j]://
   {

     CyDelay(1);
     LED_IN1_Sel_Write(0);
     LED_IN2_Sel_Write(0);
     LED_IN3_Sel_Write(0);
     LED_IN4_Sel_Write(0);
     LED_IN5_Sel_Write(0);
     LED_IN6_Sel_Write(0);
     PD_IN1_Sel_Write(0);
     PD_IN2_Sel_Write(0);
     PD_IN3_Sel_Write(0);
     PD_IN4_Sel_Write(0);
     PD_IN5_Sel_Write(0);
     PD_IN6_Sel_Write(0);
     CyDelay(1);
       if(i==0)
       PD_IN1_Sel_Write(1);
       if(i==1)
       PD_IN2_Sel_Write(1);
       if(i==2)
       PD_IN3_Sel_Write(1);
       if(i==3)
       PD_IN4_Sel_Write(1);
       if(i==4)
       PD_IN5_Sel_Write(1);
       if(i==5)
       PD_IN6_Sel_Write(1);
       CyDelay(1);

       ADC_DelSig_IRQ_Start();//:Enables interrupts at the end of conversion://
           ADC_DelSig_Start();//:Start the ADC://
           ADC_DelSig_StartConvert();//:Starts conversion://
       Filter_Start();//:Starts the filter://
       DMA_Config//:Configure the DMA by calling the DMA_Config function://
       CYGlobalIntEnable;//:Enable Global Interrupts://
       if(ADC_DelSig_IsEndConversion(ADC_DelSig_WAIT_FOR_RESULT)) //:If the ADC is
                                                                  ready with a result://
         {
          out[i][j]=ADC_DelSig_GetResult16();//:Set the value of the correct ADC PD
                                             Input into the "out" array
                                             out[i][j]://
         }

         CyDelay(1);

     }
}
     for (i=0; i<6; i++)//: initialize each of the 6 PD_rms and mean://
       {
        PD_rms[i]=0.0;
        mean=0.0;
        for(j=1; j<N; j++)//: calculate the mean value of all of the PD inputs://
         {
          mean=mean+out[i][j];
         }
         mean=mean/(N-1);
        for(j=1; j<N; j++)//: calculate the rms value of each of the PD inputs://
         {
          PD_rms[i]=PD_rms[i]+(out[i][j]-mean)*(out[i][j]-mean);
         }
         PD_rms[i]=PD_rms[i]/(N-1);
         PD_rms[i]=sqrt(PD_rms[i]);
                          rms_cal[i]=PD_rms[i];
                          PD_rms[i]=PD_rms[i]/rms_ratio;
                          PD_rms[i]=PD_rms[i]/rms_constant;
        }

                          VDAC_Start();
```

```
                                if (PD_rms[0]>(0.2*Vpd_rms_ref))//:If the rms voltage of
                                                                  the first PD is above
                                                                  6(=0.2*Vpd_rms_ref
                                                                  (currently 30))://
            {

                                rms_cal[0]=0.69*rms_cal[0]+0.57*kf;//:set the rms_cal of
                                                                  the first PD to 0.69 times
                                                                  its old value, plus0.57*kf
                                                                  (set earlier, on line 299)://
                                kf=rms_cal[0];//:change kf to this new rms_cal value
                                                                 for the next go-round://
             VDAC_SetValue(rms_cal[0]);//:send the value of the first pd rms
                                                                 calculation out to be monitored on the back
                                                                 panel://
                                CyDelay(10);
                               }
             else
            {
                                VDAC_SetValue(3);//:If the rms voltage of the first PD
                                                                 is not above 6, send out a DAC value
                                                                 of 3 to the back panel://
                                CyDelay(10);
            }



m=0;
for (i=0; i<6; i++)//:Check if any of the PD_rms values are greater than the reference
                     voltage://
   {
     if (PD_rms[i]>Vpd_rms_ref)//:If any are greater, set the "m" error bit to one://
                      {
                      m=1;
                     }
                else
                      m=m;//:Otherwise, don't://
            }

          for (i=0; i<6; i++)//:Loop 6 times, and compare the value of the correct ADC
                                LED Input to the DAC value of 2640mV from "DAC"://
           {
             CyDelay(1);
             Pin_Init();
             CyDelay(5);
             if(i==0)
             LED_IN1_Sel_Write(1);
             if(i==1)
             LED_IN2_Sel_Write(1);
             if(i==2)
             LED_IN3_Sel_Write(1);
             if(i==3)
             LED_IN4_Sel_Write(1);
             if(i==4)
             LED_IN5_Sel_Write(1);
             if(i==5)
             LED_IN6_Sel_Write(1);
                DAC_Start();
                Comp1_Start();
                CyDelay(1);
       if (LED_info_Read())
     if (Comp1_GetCompare())//:If any are less than the DAC value, set the "k" error bit
                             to one://
                k=1;
             else
                k=k;//:Otherwise, don't do anything, right here.://
          }

             if((m==1) || (k==1))//:If the PD rms error, OR the LED error have been
                                   thrown, continue the "l" counter://
          {
```

```
                    l=l+1;

                }
                  else
                {
                   l=l;//:Otherwise, don't do anything, right here.://
                }
                  if (m==0)//:Then, by logic, k==1, right?://
                      PD_ER_Write(0);
                  if (m==1)
                      PD_ER_Write(1);//:If the "m" error bit is set, write out the PD _ER
                                       bit://
                  if (k==0)
                      LED_ER_Write(0);
                  if (k==1)
                      LED_ER_Write(1);//:I think you get this, by now.://


                  if(((!Remote_Reset_Read()) || Reset_rqst_in_Read()) & (m==0) & (k==0))

                 {  //:If there is a "Reset" request, and no lingering errors, go back to
                      the very beginning (after sitting here for 2 seconds, of course.://

                  CyDelay(2000);
                       l=0;//:Reset the timer://
                       goto X0;
                  }

                   else//:Otherwise, reset the errors, and go back to the "we're inside the
                         HEPI fault" loop://
                   {
                   m=0;
                   k=0;
                   goto X5;
                  }


 }
/*****************************************************************************
* Function Name: Pin_Init
*****************************************************************************/
/* Summary:
*  Initializes Hardware MUX channels
*
*****************************************************************************/
                void Pin_Init(void)
                {
                 CyDelay(10);
                 LED_IN1_Sel_Write(0);
                 LED_IN2_Sel_Write(0);
                 LED_IN3_Sel_Write(0);
                 LED_IN4_Sel_Write(0);
                 LED_IN5_Sel_Write(0);
                 LED_IN6_Sel_Write(0);
                 PD_IN1_Sel_Write(0);
                 PD_IN2_Sel_Write(0);
                 PD_IN3_Sel_Write(0);
                 PD_IN4_Sel_Write(0);
                 PD_IN5_Sel_Write(0);
                 PD_IN6_Sel_Write(0);
                 CyDelay(1);
               }

/*****************************************************************************
* Function Name: DMA_Config
*****************************************************************************/
/* Summary:
*  Initializes and sets up DMA for use
*
*****************************************************************************/
void DMA_Config(void)
```

```
{
        /* Declare variable to hold the handle for DMA channel */
        uint8 channelHandle;
        /* Declare DMA Transaction Descriptor for memory transfer into
         * High Pass Filter Channel.
         */
        uint8 tdChanA;
        /* Configure the DMA to Transfer the data in 1 burst with individual trigger
         * for each burst.
         */
        channelHandle = DMA_DmaInitialize(BYTES_PER_BURST, REQUEST_PER_BURST,
                                          HI16(UPPER_SRC_ADDRESS), HI16(UPPER_DEST_ADDRESS));

        tdChanA = CyDmaTdAllocate();

        CyDmaTdSetConfiguration(tdChanA, 1, DMA_INVALID_TD, 0);
        /* Set the source address as ADC_DelSig and the destination as
         * Filter Channel A.   */
        CyDmaTdSetAddress(tdChanA, LO16((uint32)ADC_DelSig_DEC_SAMP_PTR),
LO16((uint32)Filter_STAGEAH_PTR));
        /* Set tdChanA to be the initial TD associated with channelHandle */
        CyDmaChSetInitialTd(channelHandle, tdChanA);

        /* Enable the DMA channel represented by channelHandle and preserve the TD */
        CyDmaChEnable(channelHandle, 1);
}
/*******************************************************************************
* Function Name: Zero_Init
********************************************************************************/
/* Summary:
*  Initializes Output Variables upon startup
*
*******************************************************************************/
   void Zero_Init(void)
    {
       HEPI_ERROR_Write(0);
       SUS_ERROR_Write(0);
       PD_ER_Write(0);
       LED_ER_Write(0);
    }
```