```
/*!
*
*  @mainpage Airbag Evaluation Platform
*
*  @section Intro Introduction
*
*  AIRBAG EVALUATION PLATFORM SHOWCASES A COMPLEX SOFTWARE INTENDED ONLY TO
*  DEMONSTRATE THE FUNCTIONALITY OF THE REFERENCE DESIGN HARDWARE AND IS NOT
*  INTENDED TO REPRESENT A TRUE AIRBAG APPLICATION.
*
*  AEP system settings:
*
*
*  For more information about the functions and configuration items refer
*  to the following documents:
*
*******************************************************************************
*
* @attention
*            1. Software is intended only to demonstrate the functionality of
*              the reference design hardware and is not intended to represent
*              a true airbag application.
*
*
* @attention
*            2. For more information about the package see the Release notes
*
*******************************************************************************/
/**
 Copyright (c) 2011 - 2012 Freescale Semiconductor
 Freescale Confidential Proprietary
 \file      main.c
 \brief     main application
 \author    Freescale Semiconductor
 \author    ASG Automotive
 \author    R70173
 \version
 \date

 * History:

 */

#include "Project_Option.h"
#include "main.h"
/*RD software platform includes*/
#include "CONFIG_OS_SCH.h"
#include "CONFIG_MCAL.h"
#include "CONFIG_COMPLEXDRV.h"
/* GUI communication */
#include "CONFIG_GUI.h"
/*application includes*/
#include "CONFIG_APPLICATION.h"
#include "Global.h"

/*
 *******************************************************************************
 * Constants
 *******************************************************************************
 */
/*! Firmware revision value. */
/*
 *******************************************************************************
 * Globals
 *******************************************************************************
 */

/*********** GUI Communication VARIABLES ***************************************
/* FreeMASTER control variables and TSA table */
```

```
FMSTR_TSA_TABLE_BEGIN(table)
        FMSTR_TSA_RW_VAR(Asbc_SetGuiParametersFlag, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Acc_SetGuiParametersFlag, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Sqb1_SetGuiParametersFlag, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Sqb2_SetGuiParametersFlag, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Application_ResetAirbagsFlag, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold0, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt0, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold1, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt1, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold2, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt2, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold3, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt3, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold4, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt4, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold5, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt5, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold6, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt6, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingThreshold7, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingDwellExt7, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingModeRequest, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingTestEnable, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_SafingLevel, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_VregSyncSupply, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_VregBoost, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_VregBuck, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_VregEnergyReserve, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_LinSlewRate, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_LinRXDMode, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_LinRXOut, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann1Mode, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann1Enable, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann1SynPuls, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann2Mode, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann2Enable, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann2SynPuls, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann3Mode, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann3Enable, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann3SynPuls, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann4Mode, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann4Enable, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_PSI5Chann4SynPuls, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_Gpo1DriverConfig, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_Gpo2DriverConfig, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_Gpo1PwmDutyCycle, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_Gpo2PwmDutyCycle, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_DcsMuxSource, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_DcsMuxVoltage, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_AnlMuxSource, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Asbc_WatchdogEnable, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(deviceTabSelected, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(intCurrentTab, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Sqb1_Fen1Delay, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Sqb1_Fen2Delay, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Sqb2_Fen1Delay, FMSTR_TSA_UINT8)
        FMSTR_TSA_RW_VAR(Sqb2_Fen2Delay, FMSTR_TSA_UINT8)
        FMSTR_TSA_TABLE_END()

    /* TSA table list describes all TSA tables which should be exported to the
    FreeMASTER application */
    FMSTR_TSA_TABLE_LIST_BEGIN()
        FMSTR_TSA_TABLE(table)
    FMSTR_TSA_TABLE_LIST_END()
/*
 *****************************************************************************
 * main
 *****************************************************************************
```

```
 */

void main(void)
{
    /* First of all, we need to initialize Platform HW devices */
    Platform_init();


    while (1)
    {
    /*************************/
    /* Communication with PC */
    /*************************/
      FMSTR_Poll(); /* FreeMASTER "polling" call */

    /*****************************************************************
    *                Read all devices parameters
    *****************************************************************/

      switch(intCurrentTab){

        /*****************************/
        /* Debug Mode (main bookmark) */
        /*****************************/
        case 0:
          switch(deviceTabSelected){
            case 0: /* MC33789 Airbag System Basis Chip (inside the Debug Mode
            bookmark) */
              /* Read ASBC status-es */
                GUI_counter++;
              if(GUI_counter == 0xF00){
                    GUI_counter = 0;
              ret_asbc = Asbc_GetStatus(ARD_SPI_ASBC, &Asbc_Status); /* common status
              of the ASBC device */
              ret_asbc = Asbc_GetLinStatus(ARD_SPI_ASBC, &Asbc_LinStatus); /* get LIN
              physical layer settings */
              ret_asbc = Asbc_GetPsi5Status(ARD_SPI_ASBC, &Asbc_Psi5Status); /* the
              status of the ASBC PSI5 interface */
              ret_asbc = Asbc_GetVregStatus(ARD_SPI_ASBC, ASBC_VREG_ESR_EN,
              &Asbc_VregStatus); /* read status of the ASBC voltage regulators and
              measure state of the Energy Reserve capacitor */
              ret_asbc = Asbc_GetGpoStatus(ARD_SPI_ASBC, ASBC_GPO_1, &Asbc_Gpo1Status);
              /* read status of the selected general purpose driver */
              ret_asbc = Asbc_GetGpoStatus(ARD_SPI_ASBC, ASBC_GPO_2, &Asbc_Gpo2Status);
              /* read status of the selected general purpose driver */
                }
              break;
            case 1: /* MMA6801QR2 Central Accelerometer (inside the Debug Mode
            bookmark) */
              /* Read ACC status */
              GUI_counter++;
              if(GUI_counter == 0xFFFF){
                GUI_counter = 0;
                ret_acc = Acc_GetStatus(ARD_SPI_ACC, &Acc_Status); /* read the complete
                statuses of the ACC device */
              }
              break;
            case 2: /* MC33797 SQUIB1 Driver (inside the Debug Mode bookmark) */
              /* Read SQUIB1 status */
              GUI_counter++;
              if(GUI_counter == 0xFFFF){
                GUI_counter = 0;
                ret_squib = Squib_GetStatus(ARD_SPI_SQUIB1, &Sqb1_Status); /* get
                status of the 1A, 1B, 2A and 2B of the SQUIB1 */
              }
              break;
            case 3: /* MC33797 SQUIB2 Driver (inside the Debug Mode bookmark) */
              /* Read SQUIB2 status */
              GUI_counter++;
```

```
196               if(GUI_counter == 0xFFFF){
197                 GUI_counter = 0;
198                 ret_squib = Squib_GetStatus(ARD_SPI_SQUIB2, &Sqb2_Status); /* get
                    status of the 1A, 1B, 2A and 2B of the SQUIB2 */
199               }
200               break;
201             default:
202               break;
203           }
204           break;
205         /***********************************/
206         /* Application Mode (main bookmark) */
207         /***********************************/
208         case 1:
209           /* Synchronization pulse starts */
210           SyncPulseStart(); /* rising edge of the SATSYNC pulse */
211          /* Acquisition Sequence #0 and #1 - read central accelerometer X-axis and
             Y-axis data */
212           ret_acc = Acc_GetAccelData(ARD_SPI_ACC, ACC_X_OFFSETCANCEL_SIGNED_ARMENABLE,
              ACC_Y_OFFSETCANCEL_SIGNED_ARMENABLE, &AccelerationData); /* read X and Y axis
              accelerometer moving value and error status */
213           if((AccelerationData.AccelDataX < 200)){
214             if(AccelerationData.AccelDataX > CentralAccel_DataX){
215               CentralAccel_DataX = AccelerationData.AccelDataX; /* X-axis acceleration
                  data from central accelerometer */
216             }
217           }
218           if((AccelerationData.AccelDataY < 200)){
219             if(AccelerationData.AccelDataY > CentralAccel_DataY){
220               CentralAccel_DataY = AccelerationData.AccelDataY; /* Y-axis acceleration
                  data from central accelerometer */
221             }
222           }
223           /* Acquisition Sequence #2 - dummy reading (PSI5 LC = 0011) */
224           Asbc_ReadSensor(ARD_SPI_ASBC, ASBC_SEQUENCE_IDENTIFIER_02,
              ASBC_LOG_PSI5_CHAN1_DUMMY, &SensorDummy, SensStatus); /* dummy reading */
225           /* Acquisition Sequence #3 - dummy reading (PSI5 LC = 0111) */
226           Asbc_ReadSensor(ARD_SPI_ASBC, ASBC_SEQUENCE_IDENTIFIER_03,
              ASBC_LOG_PSI5_CHAN2_DUMMY, &SensorDummy, SensStatus); /* dummy reading */
227           /* Acquisition Sequence #4 - read front-left satellite (PSI5 LC = 0000) */
228           Asbc_ReadSensor(ARD_SPI_ASBC, ASBC_SEQUENCE_IDENTIFIER_04,
              ASBC_LOG_PSI5_CHAN1_SLOT1, &SensorDataTemporary, SensStatus); /* acceleration
              data from front left satellite sensor */
229           if(SensorDataTemporary < 200){
230             if(SensorDataTemporary > SensorData_Driver){
231               SensorData_Driver = SensorDataTemporary;
232             }
233           }
234           /* Acquisition Sequence #5 - read front-right satellite (PSI5 LC = 0100) */
235           Asbc_ReadSensor(ARD_SPI_ASBC, ASBC_SEQUENCE_IDENTIFIER_05,
              ASBC_LOG_PSI5_CHAN2_SLOT1, &SensorDataTemporary, SensStatus); /* acceleration
              data from front right satellite sensor */
236           if(SensorDataTemporary < 200){
237             if(SensorDataTemporary > SensorData_Passenger){
238               SensorData_Passenger = SensorDataTemporary;
239             }
240           }
241           /* Acquisition Sequence #6 - read side-right satellite (PSI5 LC = 1000) */
242           Asbc_ReadSensor(ARD_SPI_ASBC, ASBC_SEQUENCE_IDENTIFIER_06,
              ASBC_LOG_PSI5_CHAN3_SLOT1, &SensorDataTemporary, SensStatus); /* acceleration
              data from rear right satellite sensor */
243           if(SensorDataTemporary < 200){
244             if(SensorDataTemporary > SensorData_RearRight){
245               SensorData_RearRight = SensorDataTemporary;
246             }
247           }
248           /* Acquisition Sequence #7 - read side-left satellite (PSI5 LC = 1100) */
249           Asbc_ReadSensor(ARD_SPI_ASBC, ASBC_SEQUENCE_IDENTIFIER_07,
              ASBC_LOG_PSI5_CHAN4_SLOT1, &SensorDataTemporary, SensStatus); /* acceleration
```

```
            data from rear left satellite sensor */
250         if(SensorDataTemporary < 200){
251           if(SensorDataTemporary > SensorData_RearLeft){
252             SensorData_RearLeft = SensorDataTemporary;
253           }
254         }
255         /* Complete synchronization pulse */
256         SyncPulseEnd(); /* falling edge of the SATSYNC pulse */
257
258         /*******************************************************/
259         /************** AIRBAG LEDs Section ********************/
260         /*******************************************************/
261
262         /* Turn ON relevant LEDs to synchronize with GUI */
263         //if(((SensorData_Driver > AIRBAG_THRESHOLD_FRONT) || (SensorData_Passenger >
            AIRBAG_THRESHOLD_FRONT)) && ((CentralAccel_DataX > AIRBAG_TRESHOLD_FRONT_XY)
            || (CentralAccel_DataY > AIRBAG_TRESHOLD_FRONT_XY))){
264           //  vfnARD_ControlLEDs(FrontAirbags_LED, TRUE);
265         //}
266         /* Use of Low Thresholds Demo */
267         if(((SensorData_Driver > AIRBAG_THRESHOLD_FRONT) || (SensorData_Passenger >
            AIRBAG_THRESHOLD_FRONT)) && ((CentralAccel_DataX > AIRBAG_TRESHOLD_FRONT_XY)
            )){
268              vfnARD_ControlLEDs(FrontAirbags_LED, TRUE);
269          }
270
271         if (SensorData_RearRight > AIRBAG_THRESHOLD_REAR_RIGHT){
272             vfnARD_ControlLEDs(RearRight_LED, TRUE);
273          }
274
275         if (SensorData_RearLeft > AIRBAG_THRESHOLD_REAR_LEFT){
276             vfnARD_ControlLEDs(RearLeft_LED, TRUE);
277          }
278
279         /* Turn OFF all LEDs if Reset Airbag is applied on GUI*/
280         if (Application_ResetAirbagsFlag == 1){
281           vfnARD_ControlLEDs(FrontAirbags_LED, CLEAR);
282           vfnARD_ControlLEDs(RearRight_LED, CLEAR);
283           vfnARD_ControlLEDs(RearLeft_LED, CLEAR);
284           Application_ResetAirbagsFlag = 6;   /* Clear Flag*/
285         }
286         break;
287       }
288
289     /*************************************************************
290     *                  Write devices parameters
291     *************************************************************/
292
293       /************************/
294       /* Write ASBC parameters */
295       /************************/
296       if(Asbc_SetGuiParametersFlag == 1){ /* are update of the ASBC parameters
          required? */
297         /* Update configuration structures from GUI page */
298         Asbc_Conf.Asbc_SafingThreshold0 = Asbc_SafingThreshold0;
299         Asbc_Conf.Asbc_SafingDwellExt0 = Asbc_SafingDwellExt0;
300         Asbc_Conf.Asbc_SafingThreshold1 = Asbc_SafingThreshold1;
301         Asbc_Conf.Asbc_SafingDwellExt1 = Asbc_SafingDwellExt1;
302         Asbc_Conf.Asbc_SafingThreshold2 = Asbc_SafingThreshold2;
303         Asbc_Conf.Asbc_SafingDwellExt2 = Asbc_SafingDwellExt2;
304         Asbc_Conf.Asbc_SafingThreshold3 = Asbc_SafingThreshold3;
305         Asbc_Conf.Asbc_SafingDwellExt3 = Asbc_SafingDwellExt3;
306         Asbc_Conf.Asbc_SafingThreshold4 = Asbc_SafingThreshold4;
307         Asbc_Conf.Asbc_SafingDwellExt4 = Asbc_SafingDwellExt4;
308         Asbc_Conf.Asbc_SafingThreshold5 = Asbc_SafingThreshold5;
309         Asbc_Conf.Asbc_SafingDwellExt5 = Asbc_SafingDwellExt5;
310         Asbc_Conf.Asbc_SafingThreshold6 = Asbc_SafingThreshold6;
311         Asbc_Conf.Asbc_SafingDwellExt6 = Asbc_SafingDwellExt6;
312         Asbc_Conf.Asbc_SafingThreshold7 = Asbc_SafingThreshold7;
```

```
313             Asbc_Conf.Asbc_SafingDwellExt7 = Asbc_SafingDwellExt7;
314             Asbc_Conf.Asbc_SafingModeRequest = Asbc_SafingModeRequest;
315             Asbc_Conf.Asbc_SafingTestEnable = Asbc_SafingTestEnable;
316             Asbc_Conf.Asbc_SafingLevel = Asbc_SafingLevel;
317             Asbc_VregConf.Asbc_VregSyncSupply = Asbc_VregSyncSupply;
318             Asbc_VregConf.Asbc_VregBoost = Asbc_VregBoost;
319             Asbc_VregConf.Asbc_VregBuck = Asbc_VregBuck;
320             Asbc_VregConf.Asbc_VregEnergyReserve = Asbc_VregEnergyReserve ;
321             Asbc_LinConf.Asbc_LinSlewRate = Asbc_LinSlewRate;
322             Asbc_LinConf.Asbc_LinRXDMode = Asbc_LinRXDMode ;
323             Asbc_LinConf.Asbc_LinRXOut = Asbc_LinRXOut;
324             Asbc_Psi5Conf.Asbc_PSI5Chann1Mode = Asbc_PSI5Chann1Mode;
325             Asbc_Psi5Conf.Asbc_PSI5Chann1Enable = Asbc_PSI5Chann1Enable;
326             Asbc_Psi5Conf.Asbc_PSI5Chann1SynPuls = Asbc_PSI5Chann1SynPuls;
327             Asbc_Psi5Conf.Asbc_PSI5Chann2Mode = Asbc_PSI5Chann2Mode;
328             Asbc_Psi5Conf.Asbc_PSI5Chann2Enable = Asbc_PSI5Chann2Enable;
329             Asbc_Psi5Conf.Asbc_PSI5Chann2SynPuls = Asbc_PSI5Chann2SynPuls;
330             Asbc_Psi5Conf.Asbc_PSI5Chann3Mode = Asbc_PSI5Chann3Mode;
331             Asbc_Psi5Conf.Asbc_PSI5Chann3Enable = Asbc_PSI5Chann3Enable;
332             Asbc_Psi5Conf.Asbc_PSI5Chann3SynPuls = Asbc_PSI5Chann3SynPuls;
333             Asbc_Psi5Conf.Asbc_PSI5Chann4Mode = Asbc_PSI5Chann4Mode;
334             Asbc_Psi5Conf.Asbc_PSI5Chann4Enable = Asbc_PSI5Chann4Enable;
335             Asbc_Psi5Conf.Asbc_PSI5Chann4SynPuls = Asbc_PSI5Chann4SynPuls;
336
337           /* Update ASBC parameters */
338           ret_asbc = Asbc_Init(ARD_SPI_ASBC, &Asbc_Conf); /* initialization of the Airbag
              System Basis Driver device MC33789 */
339           ret_asbc = Asbc_SetLinMode(ARD_SPI_ASBC, &Asbc_LinConf); /* LIN physical layer
              configuration */
340           ret_asbc = Asbc_SetVregMode(ARD_SPI_ASBC, &Asbc_VregConf); /* configure voltage
              regulators */
341           ret_asbc = Asbc_SetPsi5Mode(ARD_SPI_ASBC, &Asbc_Psi5Conf); /* configure PSI5
              interface - turn satellite sensors interface OFF */
342           ret_asbc = Asbc_SetGpo(ARD_SPI_ASBC, ASBC_GPO_1, Asbc_Gpo1PwmDutyCycle,
              Asbc_Gpo1DriverConfig ); /* send parameters to GPO1 output */
343           ret_asbc = Asbc_SetGpo(ARD_SPI_ASBC, ASBC_GPO_2, Asbc_Gpo2PwmDutyCycle,
              Asbc_Gpo2DriverConfig ); /* send parameters to GPO2 output */
344           ret_asbc = Asbc_SetDcsMuxSource(ARD_SPI_ASBC, Asbc_DcsMuxSource ,
              Asbc_DcsMuxVoltage); /* send DCS sensors MUX settings */
345           ret_asbc = Asbc_SetAnlMuxSource(ARD_SPI_ASBC, Asbc_AnlMuxSource); /* send
              analog MUX settings */
346
347           /* Enable/Disable ASBC watchdog */
348           //if(Asbc_WatchdogEnable == 1){ /* enable watchdog */
349           //  Gpt_Enable(); /* enable ASBC watchdog refresh  */
350           //}else{ /* disable watchdog */
351            // Gpt_Disable(); /* disable ASBC watchdog refresh - disable RTI interrupt */
352          // }
353           /* Clear FreeMASTER flag */
354           Asbc_SetGuiParametersFlag = 0; /* ASBC reconfiguration finished */
355         }
356         /*************************************************/
357         /* Write MMA68xx Central Accelerometer parameters */
358         /*************************************************/
359         if(Acc_SetGuiParametersFlag == 1){ /* are update of the ACC device parameters
            required? */
360           /* Update configuration structures from GUI page */
361             Acc_Conf.Acc_ConfSignData = Acc_ConfSignData;
362             Acc_Conf.Acc_OffsetMoni = Acc_OffsetMoni;
363             Acc_Conf.Acc_ArmOutput = Acc_ArmOutput;
364             Acc_Conf.Acc_XAxisSelfTest = Acc_XAxisSelfTest;
365             Acc_Conf.Acc_YAxisSelfTest = Acc_YAxisSelfTest;
366             Acc_Conf.Acc_XLowPassFilter = Acc_XLowPassFilter;
367             Acc_Conf.Acc_YLowPassFilter = Acc_YLowPassFilter;
368             Acc_Conf.Acc_XArmPulseStretch = Acc_XArmPulseStretch;
369             Acc_Conf.Acc_YArmPulseStretch = Acc_YArmPulseStretch;
370             Acc_Conf.Acc_XArm_PosWin_CountLimit = Acc_XArm_PosWin_CountLimit;
371             Acc_Conf.Acc_YArm_PosWin_CountLimit = Acc_YArm_PosWin_CountLimit;
372             Acc_Conf.Acc_XArm_NegWinSize = Acc_XArm_NegWinSize;
```

```
373             Acc_Conf.Acc_YArm_NegWinSize = Acc_YArm_NegWinSize;
374             Acc_Conf.Acc_XArmPositiveThreshold = Acc_XArmPositiveThreshold; /* 20G X
                positive (sensitivity 0.1024g/LSB) */
375             Acc_Conf.Acc_YArmPositiveThreshold = Acc_YArmPositiveThreshold; /* 20G Y
                positive (sensitivity 0.1024g/LSB) */
376             Acc_Conf.Acc_XArmNegativeThreshold = Acc_XArmNegativeThreshold; /* -20G X
                negative (sensitivity 0.1024g/LSB) */
377             Acc_Conf.Acc_YArmNegativeThreshold = Acc_YArmNegativeThreshold; /* -20G Y
                negative (sensitivity 0.1024g/LSB) */
378
379             /* Update Central Acceleration Accelerometer parameters */
380             ret_acc = Acc_Init(ARD_SPI_ACC, &Acc_Conf); /* resetup central accelerator
                device */
381             Acc_SetGuiParametersFlag = 0; /*  */
382         }
383         /***********************************/
384         /* Write MC33797 Squib 2 parameters */
385         /***********************************/
386         if(Sqb1_SetGuiParametersFlag == 1){ /* are update of the SQUIB1 device parameters
            required? */
387           /* Update configuration structures from GUI page - when the FEN_1 or FEN_2
              input state transitions from high to low, a programmable latching function will
              hold the FEN function active until the timeout of the FEN timer */
388           ret_squib = Squib_ProgramCmd(ARD_SPI_SQUIB1, SQB_UNLOCK_FEN1_COUNTER_REG,
              Sqb1_Fen1Delay, CLEAR, &SquibCmdResp); /* write FEN_1 counter delay value */
389           ret_squib = Squib_ProgramCmd(ARD_SPI_SQUIB1, SQB_UNLOCK_FEN2_COUNTER_REG,
              Sqb1_Fen2Delay, CLEAR, &SquibCmdResp); /* write FEN_2 counter delay value */
390           Sqb1_SetGuiParametersFlag = 0; /* SQUIB1 reconfiguration finished */
391         }
392         /***********************************/
393         /* Write MC33797 Squib 2 parameters */
394         /***********************************/
395         if(Sqb2_SetGuiParametersFlag == 1){ /* are update of the SQUIB2 device parameters
            required? */
396           /* Update configuration structures from GUI page - when the FEN_1 or FEN_2
              input state transitions from high to low, a programmable latching function will
              hold the FEN function active until the timeout of the FEN timer */
397           ret_squib = Squib_ProgramCmd(ARD_SPI_SQUIB2, SQB_UNLOCK_FEN1_COUNTER_REG,
              Sqb2_Fen1Delay, CLEAR, &SquibCmdResp); /* write FEN_1 counter delay value */
398           ret_squib = Squib_ProgramCmd(ARD_SPI_SQUIB2, SQB_UNLOCK_FEN2_COUNTER_REG,
              Sqb2_Fen2Delay, CLEAR, &SquibCmdResp); /* write FEN_2 counter delay value */
399           Sqb2_SetGuiParametersFlag = 0; /* SQUIB2 reconfiguration finished */
400         }
401
402         /* Enable/Disable ASBC watchdog */
403       //   if(Asbc_WatchdogEnable == 1){ /* enable watchdog */
404       //     if(WD_FeedSign == TRUE){ /* if HIGH WD feed is required */
405       //       ret_asbc = Asbc_FeedWatchdog(ARD_SPI_ASBC, ASBC_WD_FEED_HIGH); /* HIGH
          feeds of the ASBC watchdog */
406       //     }else{ /* if LOW WD feed is required */
407       //       ret_asbc = Asbc_FeedWatchdog(ARD_SPI_ASBC, ASBC_WD_FEED_LOW); /* LOW feeds
          of the ASBC watchdog */
408       //     }
409       //     WD_FeedSign = !(WD_FeedSign); /* flip WD flag for the next watchdog refresh */
410       //   }else{
411             //Do nothing -- Warning: THIS WILL CAUSE A SYSTEM RESET
412       //   }
413
414     } /* end of the "never-ending loop" */
415
416   }
417   /* ******************************************************************************
418    *
419    *  End of file.
420    *
421    ******************************************************************************
422    */
423
```