```
/*******************************************************************************
  MPLAB Harmony Application Source File

  Company:
    Microchip Technology Inc.

  File Name:
    app.c

  Summary:
    This file contains the source code for the MPLAB Harmony application.

  Description:
    This file contains the source code for the MPLAB Harmony application.  It
    implements the logic of the application's state machine and it may call
    API routines of other MPLAB Harmony modules in the system, such as drivers,
    system services, and middleware.  However, it does not call any of the
    system interfaces (such as the "Initialize" and "Tasks" functions) of any of
    the modules in the system or make any assumptions about when those functions
    are called.  That is the responsibility of the configuration-specific system
    files.
 *******************************************************************************/

// DOM-IGNORE-BEGIN
/*******************************************************************************
Copyright (c) 2013-2014 released Microchip Technology Inc.  All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute
Software only when embedded on a Microchip microcontroller or digital signal
controller that is integrated into your product or third party product
(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for
additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
 *******************************************************************************/
// DOM-IGNORE-END


// *****************************************************************************
// *****************************************************************************
// Section: Included Files
// *****************************************************************************
// *****************************************************************************

#include "app.h"
#include "bno055.h"
#include "bno055_support.h"
#include "Mc32_I2cUtilCCS.h"
#include "Mc32_serComm.h"
#include "Mc32_sdFatGest.h"
#include "Mc32_PressAdc.h"
#include "Mc32Debounce.h"
#include <stdio.h>

// *****************************************************************************
// *****************************************************************************
// Section: Global Data Definitions
// *****************************************************************************
// *****************************************************************************
/* Switch descriptor */
S_SwitchDescriptor switchDescr;
```

```
// *****************************************************************************
/* Application Data

  Summary:
    Holds application data

  Description:
    This structure holds the application's data.

  Remarks:
    This structure should be initialized by the APP_Initialize function.

    Application strings and buffers are be defined outside this structure.
*/

APP_DATA appData;
TIMER_DATA timerData;

// *****************************************************************************
// *****************************************************************************
// Section: Application Callback Functions
// *****************************************************************************
// *****************************************************************************
void MainTimer_callback(){
    /* Increment delay timer */
    timerData.TmrCnt ++;
}

void DisplayTimer_callback()
{
    /* Increment utility timers */
    timerData.TmrDisplay ++;
    timerData.TmrMeas ++;
    timerData.TmrTickFlag = true;
    /* If button is pressed, count pressed time */
    if(timerData.flagCountBtnPressed){
        timerData.TmrBtnPressed++;
    }
     /* Do debounce every 10 ms */
     DoDebounce(&switchDescr, ButtonMFStateGet());
    /* Start a measure set each 90ms */
    if ( ( timerData.TmrMeas % 9 ) == 0)
        timerData.measTodoFlag = true;
}
/* TODO:  Add any necessary callback functions.
*/

// *****************************************************************************
// *****************************************************************************
// Section: Application Local Functions
// *****************************************************************************
// *****************************************************************************


/* TODO:  Add any necessary local functions.
*/


// *****************************************************************************
// *****************************************************************************
// Section: Application Initialization and State Machine Functions
// *****************************************************************************
// *****************************************************************************

/*******************************************************************************
  Function:
    void APP_Initialize ( void )

  Remarks:
    See prototype in app.h.
 */

void APP_Initialize ( void )
{
```

```
147     /* Place the App state machine in its initial state. */
148     appData.state = APP_STATE_INIT;
149     /* Init all counters and flags */
150     timerData.mainTmrCnt = 0;
151     timerData.TmrCnt = 0;
152     timerData.TmrTickFlag = false;
153     timerData.TmrDisplay = 0;
154     timerData.measTodoFlag = false;
155     timerData.flagCountBtnPressed = false;
156     timerData.TmrBtnPressed = 0;
157 
158     /* Hold the device on */
159     PwrHoldOn();
160     /* Peripherals init */
161     DRV_TMR0_Start();
162     DRV_TMR1_Start();
163     i2c_init(1);
164     Press_InitADC();
165 
166     /* System ON display */
167     LED_BOn();
168     BNO055_delay_msek(500);
169     LED_BOff();
170 
171     /* Reset IMU */
172     RstImuOff();
173     BNO055_delay_msek(100);
174     RstImuOn();
175     BNO055_delay_msek(100);
176 
177     /* Demuliplexer config */
178     DemulCBOff();
179     DemulCCOn();
180 
181     /* Enable 5V regulator */
182     EN_5VOn();
183 
184 
185 }
186 
187 
188 /*******************************************************************************
189   Function:
190     void APP_Tasks ( void )
191 
192   Remarks:
193     See prototype in app.h.
194  */
195 
196 void APP_Tasks ( void )
197 {
198     /* Local bno055 data */
199     s_bno055_data bno055_local_data;
200     static bool Hold = false;
201     static uint8_t flagMeas = false;
202      /* Check the application's current state. */
203     switch ( appData.state )
204     {
205         /* Application's initial state. */
206         case APP_STATE_INIT:
207         {
208             // Init delay
209             BNO055_delay_msek(500);
210             // Init and Measure set
211             bno055_init_readout();
212             /* go to service task */
213             appData.state = APP_STATE_LOGGING;
214             /* Init ltime counter */
215             timerData.ltime = 0;
216             /* Init first measure flag */
217             flagMeas = FLAG_MEAS_OFF;
218             break;
219         }
220 
221         case APP_STATE_LOGGING:
```

```
222     {
223         /* Display period */
224         if(timerData.TmrDisplay >= 320)
225             timerData.TmrDisplay = 0;
226         // --- Display LED ---
227         if((timerData.TmrDisplay <= 1)&&(sd_getState() != APP_MOUNT_DISK))
228             LED_GOn();
229         else
230             LED_GOff();
231 
232         if((timerData.measTodoFlag == true )&&(sd_getState() == APP_IDLE))
233         {
234             /* BNO055 Read all important info routine */
235             bno055_local_data.comres = bno055_read_routine(&bno055_local_data);
236             /* Delta time */
237             bno055_local_data.d_time = timerData.TmrMeas - timerData.ltime;
238             /* Pressure measure */
239             bno055_local_data.pressure = Press_readPressure();
240             /* Flag measure value */
241             bno055_local_data.flagImportantMeas = flagMeas;
242             /* Display value via UART */
243             //serDisplayValues(&bno055_local_data);
244             /* Write value to sdCard */
245             sd_BNO_scheduleWrite(&bno055_local_data);
246             /* Reset measure flag */
247             if(flagMeas == FLAG_MEAS_ON){
248                 /* Rest important measure flag */
249                 flagMeas = FLAG_MEAS_OFF;
250                 LED_BOff();
251             }
252             /* Reset measure flag */
253             timerData.measTodoFlag = false;
254             /* Update last time counter */
255             timerData.ltime = timerData.TmrMeas;
256         }
257         else
258         {
259             /* No comm, so no error */
260             bno055_local_data.comres = 0;
261         }
262 
263         /* If error detected : error LED */
264         if((bno055_local_data.comres != 0)||(sd_getState() == APP_MOUNT_DISK))
265             LED_ROn();
266         else
267             LED_ROff();
268 
269         /* --- SD FAT routine --- */
270         sd_fat_task();
271 
272         /* Button management : if rising edge detected */
273         if(((ButtonMFStateGet()))||(Hold == true))
274         {
275             /* Hold until falling edge */
276             Hold = true;
277             /* Start counting pressed time */
278             timerData.flagCountBtnPressed = true;
279             /* If falling edge detected */
280             if (ButtonMFStateGet() == 0)
281             {
282                 /* Reset flag and switchdescr */
283                 timerData.flagCountBtnPressed = false;
284                 DebounceClearReleased(&switchDescr);
285                 /* If pressed time less than power off time */
286                 if((timerData.TmrBtnPressed <= TIME_POWER_OFF)&&(sd_getState() != APP_MOUNT_DISK)){
287                     flagMeas = FLAG_MEAS_ON;
288                     LED_BOn();
289                 }
290                 else{
291                     /* Power off the system */
292                     appData.state = APP_STATE_SHUTDOWN;
293                 }
294                 timerData.TmrBtnPressed = 0;
295                 Hold = false;
296             }
```

```
297           }
298
299           break;
300       }
301       case APP_STATE_SHUTDOWN:
302       {
303           /* Display shutting off mode */
304           LED_BOff();
305           LED_GOff();
306           LED_ROn();
307
308           /* If and SD card is mounted */
309           if(sd_getState() != APP_MOUNT_DISK){
310               /* Wait until SD availaible */
311               while(sd_getState() != APP_IDLE){
312                   /* SD FAT routine */
313                   sd_fat_task();
314               }
315               /* Unmount disk */
316               sd_setState(APP_UNMOUNT_DISK);
317               /* Wait until unmounted*/
318               while(sd_getState() != APP_IDLE){
319                   sd_fat_task();
320               }
321           }
322
323           /* turn off the device */
324           PwrHoldOff();
325
326           break;
327       }
328
329       /* TODO: implement your application state machine.*/
330
331
332       /* The default state should never be executed. */
333       default:
334       {
335           /* TODO: Handle error in application's state machine. */
336           break;
337       }
338   }
339 }
340
341 void App_resetMeasFlag( void )
342 {
343     timerData.measTodoFlag = false;
344 }
345
346
347 /*******************************************************************************
348  End of File
349  */
350
```

```
/*******************************************************************************
  MPLAB Harmony Application Header File

  Company:
    Microchip Technology Inc.

  File Name:
    app.h

  Summary:
    This header file provides prototypes and definitions for the application.

  Description:
    This header file provides function prototypes and data type definitions for
    the application.  Some of these are required by the system (such as the
    "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
    internally by the application (such as the "APP_STATES" definition).  Both
    are defined here for convenience.
*******************************************************************************/

//DOM-IGNORE-BEGIN
/*******************************************************************************
Copyright (c) 2013-2014 released Microchip Technology Inc.  All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute
Software only when embedded on a Microchip microcontroller or digital signal
controller that is integrated into your product or third party product
(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for
additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
 *******************************************************************************/
//DOM-IGNORE-END

#ifndef _APP_H
#define _APP_H

// *****************************************************************************
// *****************************************************************************
// Section: Included Files
// *****************************************************************************
// *****************************************************************************

#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include "system_config.h"
#include "system_definitions.h"
#include "bno055.h"

// DOM-IGNORE-BEGIN
#ifdef __cplusplus  // Provide C++ Compatibility

extern "C" {

#endif
// DOM-IGNORE-END

#define TIME_OUT 80000000U
#define TIME_POWER_OFF   500
```

```
73
74 // *****************************************************************************
75 // *****************************************************************************
76 // Section: Type Definitions
77 // *****************************************************************************
78 // *****************************************************************************
79 typedef struct {
80     s32 comres;
81     bool flagMeasReady;
82     uint8_t flagImportantMeas;
83     struct bno055_gravity_double_t gravity;
84     struct bno055_linear_accel_double_t linear_accel;
85     struct bno055_euler_double_t euler;
86     struct bno055_gyro_double_t gyro;
87     struct bno055_mag_double_t mag;
88     struct bno055_quaternion_t quaternion;
89     unsigned long time;
90     unsigned long l_time;
91     uint16_t d_time;
92     float pressure;
93 }s_bno055_data;
94 // *****************************************************************************
95 /* Application states
96
97   Summary:
98     Application states enumeration
99
100   Description:
101     This enumeration defines the valid application states.  These states
102     determine the behavior of the application at various times.
103 */
104
105 typedef enum
106 {
107         /* Application's state machine's initial state. */
108         APP_STATE_INIT=0,
109     APP_STATE_LOGGING,
110     APP_STATE_FLAG_MEAS,
111         APP_STATE_SHUTDOWN
112         /* TODO: Define states used by the application state machine. */
113
114 } APP_STATES;
115
116
117 // *****************************************************************************
118 /* Application Data
119
120   Summary:
121     Holds application data
122
123   Description:
124     This structure holds the application's data.
125
126   Remarks:
127     Application strings and buffers are be defined outside this structure.
128  */
129
130 typedef struct
131 {
132     /* The application's current state */
133     APP_STATES state;
134 } APP_DATA;
135
136 typedef struct
137 {
138     /* Main Timer (1ms) */
139     uint32_t mainTmrCnt;
140     /* Timer precis (1us) */
141     bool TmrTickFlag;
142     uint32_t TmrCnt;
143     /* Measure todo flag */
144     unsigned long TmrMeas;
145     unsigned long ltime;
146     bool measTodoFlag;
147     /* Timer display */
```

```
148     uint32_t TmrDisplay;
149     /* Tmr wait shutdown */
150     bool flagCountBtnPressed;
151     uint32_t TmrBtnPressed;
152 }TIMER_DATA;
153 
154 // *****************************************************************************
155 // *****************************************************************************
156 // Section: Application Callback Routines
157 // *****************************************************************************
158 // *****************************************************************************
159 // *****************************************************************************
160 // *****************************************************************************
161 // Section: Application Initialization and State Machine Functions
162 // *****************************************************************************
163 // *****************************************************************************
164 
165 /*******************************************************************************
166   Function:
167     void APP_Initialize ( void )
168 
169   Summary:
170      MPLAB Harmony application initialization routine.
171 
172   Description:
173     This function initializes the Harmony application.  It places the
174     application in its initial state and prepares it to run so that its
175     APP_Tasks function can be called.
176 
177   Precondition:
178     All other system initialization routines should be called before calling
179     this routine (in "SYS_Initialize").
180 
181   Parameters:
182     None.
183 
184   Returns:
185     None.
186 
187   Example:
188     <code>
189     APP_Initialize();
190     </code>
191 
192   Remarks:
193     This routine must be called from the SYS_Initialize function.
194 */
195 
196 void APP_Initialize ( void );
197 
198 void prepareBuffer( char * buffer );
199 
200 void App_resetMeasFlag( void );
201 
202 /*******************************************************************************
203   Function:
204     void APP_Tasks ( void )
205 
206   Summary:
207     MPLAB Harmony Demo application tasks function
208 
209   Description:
210     This routine is the Harmony Demo application's tasks function.  It
211     defines the application's state machine and core logic.
212 
213   Precondition:
214     The system and application initialization ("SYS_Initialize") should be
215     called before calling this.
216 
217   Parameters:
218     None.
219 
220   Returns:
221     None.
```

```
222
223   Example:
224     <code>
225     APP_Tasks();
226     </code>
227
228   Remarks:
229     This routine must be called from SYS_Tasks() routine.
230  */
231
232 void APP_Tasks( void );
233
234
235 // Callback main timer
236 void MainTimer_callback( void );
237
238 // Callback display timer
239 void DisplayTimer_callback( void );
240
241 #endif /* _APP_H */
242
243 //DOM-IGNORE-BEGIN
244 #ifdef __cplusplus
245 }
246 #endif
247 //DOM-IGNORE-END
248
249 /*******************************************************************************
250  End of File
251  */
252
253
```

```
/**
* Copyright (c) 2020 Bosch Sensortec GmbH. All rights reserved.
*
* BSD-3-Clause
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.
*
* 3. Neither the name of the copyright holder nor the names of its
*    contributors may be used to endorse or promote products derived from
*    this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
* COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
* @file bno055_support.c
* @date 10/01/2020
* @version  2.0.6
*
*/

/*---------------------------------------------------------------------------*
 *  Includes
 *---------------------------------------------------------------------------*/
#include "app.h"
#include "bno055.h"
#include "bno055_support.h"
#include "Mc32_I2cUtilCCS.h"
#include "driver/tmr/drv_tmr_static.h"

// Global variable
TIMER_DATA timerData;

#ifdef  BNO055_API

s32 bno055_read_routine(s_bno055_data *data)
{
    /* Variable used to return value of
     * communication routine*/
    s32 comres = BNO055_ERROR;

    /* variable used to set the power mode of the sensor*/
    //u8 power_mode = BNO055_INIT_VALUE;

    /*  For initializing the BNO sensor it is required to the operation mode
     * of the sensor as NORMAL
     * Normal mode can set from the register
     * Page - page0
     * register - 0x3E
     * bit positions - 0 and 1*/
    //power_mode = BNO055_POWER_MODE_NORMAL;

    /* set the power mode as NORMAL*/
    //comres += bno055_set_power_mode(power_mode);

```

```
  /*-------------------------------------------------------------*
   ************************* END INITIALIZATION *************************
   *-------------------------------------------------------------*/

  /************************* START READ RAW FUSION DATA ********
   * For reading fusion data it is required to set the
   * operation modes of the sensor
   * operation mode can set from the register
   * page - page0
   * register - 0x3D
   * bit - 0 to 3
   * for sensor data read following operation mode have to set
   * FUSION MODE
   * 0x08 - BNO055_OPERATION_MODE_IMUPLUS
   * 0x09 - BNO055_OPERATION_MODE_COMPASS
   * 0x0A - BNO055_OPERATION_MODE_M4G
   * 0x0B - BNO055_OPERATION_MODE_NDOF_FMC_OFF
   * 0x0C - BNO055_OPERATION_MODE_NDOF
   * based on the user need configure the operation mode*/
  //comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_NDOF);

  /*  Raw Quaternion W, X, Y and Z data can read from the register
   * page - page 0
   * register - 0x20 to 0x27 */
  comres += bno055_read_quaternion_wxyz(&data->quaternion);
  /************************* END READ RAW FUSION DATA  ************/
  /*****************START READ CONVERTED SENSOR DATA****************/
   /*  API used to read mag data output as double  - uT(micro Tesla)
    * float functions also available in the BNO055 API */
   comres += bno055_convert_double_mag_xyz_uT(&data->mag);
   /*  API used to read gyro data output as double  - dps and rps
    * float functions also available in the BNO055 API */
   comres += bno055_convert_double_gyro_xyz_dps(&data->gyro);
   /*  API used to read Euler data output as double  - degree and radians
    * float functions also available in the BNO055 API */
   comres += bno055_convert_double_euler_hpr_deg(&data->euler);
   /*  API used to read Linear acceleration data output as m/s2
    * float functions also available in the BNO055 API */
   comres += bno055_convert_double_linear_accel_xyz_msq(&data->linear_accel);
   comres += bno055_convert_double_gravity_xyz_msq(&data->gravity);

   /*---------------------------------------------------------------------*
    ************************* START DE-INITIALIZATION **********************
    *---------------------------------------------------------------------*/

   /*  For de - initializing the BNO sensor it is required
    * to the operation mode of the sensor as SUSPEND
    * Suspend mode can set from the register
    * Page - page0
    * register - 0x3E
    * bit positions - 0 and 1*/
   //power_mode = BNO055_POWER_MODE_SUSPEND;

   /* set the power mode as SUSPEND*/
   //comres += bno055_set_power_mode(power_mode);

   /* Flag measure ready */
   data->flagMeasReady = true;

   /*-------------------------------------------------------------*
   ************************* END DE-INITIALIZATION **********************
   *-------------------------------------------------------------*/
   return (comres+1);
}

/*---------------------------------------------------------------------*
 *  The following API is used to map the I2C bus read, write, delay and
 *  device address with global structure bno055_t
 *---------------------------------------------------------------------*/

/*---------------------------------------------------------------------*
 *  By using bno055 the following structure parameter can be accessed
 *  Bus write function pointer: BNO055_WR_FUNC_PTR
 *  Bus read function pointer: BNO055_RD_FUNC_PTR
 *  Delay function pointer: delay_msec
```

```
 *  I2C address: dev_addr
 *-------------------------------------------------------------------------*/
s8 I2C_routine(void)
{
   bno055.bus_write = BNO055_I2C_bus_write;
   bno055.bus_read = BNO055_I2C_bus_read;
   bno055.delay_msec = BNO055_delay_msek;
   bno055.dev_addr = BNO055_I2C_ADDR1;
   return BNO055_INIT_VALUE;
}

/************** I2C buffer length******/

#define I2C_BUFFER_LEN 8
#define I2C0           5

/*-------------------------------------------------------------------------*
 *
 *  This is a sample code for read and write the data by using I2C
 *  Use either I2C  based on your need
 *  The device address defined in the bno055.h file
 *
 *-------------------------------------------------------------------------*/

/*  \Brief: The API is used as I2C bus write
 *  \Return : Status of the I2C write
 *  \param dev_addr : The device address of the sensor
 *  \param reg_addr : Address of the first register,
 *   will data is going to be written
 *  \param reg_data : It is a value hold in the array,
 *      will be used for write the value into the register
 *  \param cnt : The no of byte of data to be write
 */
s8 BNO055_I2C_bus_write(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt)
{
   s8 BNO055_iERROR = BNO055_INIT_VALUE;
   u8 array[I2C_BUFFER_LEN];
   u8 stringpos = BNO055_INIT_VALUE;

   array[BNO055_INIT_VALUE] = reg_addr;

   i2c_start();
   BNO055_iERROR = i2c_write(dev_addr<<1);

   for (stringpos = BNO055_INIT_VALUE; stringpos < (cnt+BNO055_I2C_BUS_WRITE_ARRAY_INDEX); stringpos++)
   {
      BNO055_iERROR = i2c_write(array[stringpos]);
      array[stringpos + BNO055_I2C_BUS_WRITE_ARRAY_INDEX] = *(reg_data + stringpos);
   }

   i2c_stop();


   /*
    * Please take the below APIs as your reference for
    * write the data using I2C communication
    * "BNO055_iERROR = I2C_WRITE_STRING(DEV_ADDR, ARRAY, CNT+1)"
    * add your I2C write APIs here
    * BNO055_iERROR is an return value of I2C read API
    * Please select your valid return value
    * In the driver BNO055_SUCCESS defined as 0
    * and FAILURE defined as -1
    * Note :
    * This is a full duplex operation,
    * The first read data is discarded, for that extra write operation
    * have to be initiated. For that cnt+1 operation done
    * in the I2C write string function
    * For more information please refer data sheet SPI communication:
    */

   /*if(BNO055_iERROR)
      BNO055_iERROR = -1;
   else
      BNO055_iERROR = 0;
```

```
223     return (s8)(BNO055_iERROR);*/
224     // Error comm return
225
226     if(BNO055_iERROR-1 != 0)
227         BNO055_iERROR = -1;
228     else
229         BNO055_iERROR = 0;
230
231     return (s8)(BNO055_iERROR);
232 }
233
234 /*  \Brief: The API is used as I2C bus read
235  *  \Return : Status of the I2C read
236  *  \param dev_addr : The device address of the sensor
237  *  \param reg_addr : Address of the first register,
238  *  will data is going to be read
239  *  \param reg_data : This data read from the sensor,
240  *   which is hold in an array
241  *  \param cnt : The no of byte of data to be read
242  */
243 s8 BNO055_I2C_bus_read(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt)
244 {
245     s8 BNO055_iERROR = BNO055_INIT_VALUE;
246     u8 array[I2C_BUFFER_LEN] = { BNO055_INIT_VALUE };
247     u8 stringpos = BNO055_INIT_VALUE;
248
249     array[BNO055_INIT_VALUE] = reg_addr;
250
251     i2c_start();
252     // Write asked register
253     BNO055_iERROR = i2c_write(dev_addr<<1);
254     BNO055_iERROR = i2c_write(reg_addr);
255     // Send read address
256     i2c_reStart();
257     dev_addr = (dev_addr<<1) | 0b00000001;
258     BNO055_iERROR = i2c_write(dev_addr);
259
260     /* Please take the below API as your reference
261      * for read the data using I2C communication
262      * add your I2C read API here.
263      * "BNO055_iERROR = I2C_WRITE_READ_STRING(DEV_ADDR,
264      * ARRAY, ARRAY, 1, CNT)"
265      * BNO055_iERROR is an return value of SPI write API
266      * Please select your valid return value
267      * In the driver BNO055_SUCCESS defined as 0
268      * and FAILURE defined as -1
269      */
270     for (stringpos = BNO055_INIT_VALUE; stringpos < cnt; stringpos++)
271     {
272
273         if(((stringpos+1) < cnt)&&(cnt > BNO055_I2C_BUS_WRITE_ARRAY_INDEX))
274             array[stringpos] = i2c_read(1);
275         else
276             array[stringpos] = i2c_read(0);
277
278         *(reg_data + stringpos) = array[stringpos];
279
280     }
281
282     i2c_stop();
283
284     // Error comm return
285     if(BNO055_iERROR-1 != 0)
286         BNO055_iERROR = -1;
287     else
288         BNO055_iERROR = 0;
289
290     return (s8)(BNO055_iERROR);
291 }
292
293 /*  Brief : The delay routine
294  *  \param : delay in ms
295  */
296 void BNO055_delay_msek(u32 msek)
297 {
```

```
    /*Delay routine*/
    DRV_TMR0_Stop();
    DRV_TMR0_CounterClear();
    timerData.TmrCnt = 0;
    DRV_TMR0_Start();
    while (timerData.TmrCnt < msek)
    { }
    DRV_TMR0_Stop();
}

#endif


s32 bno055_init_readout(void)
{
    /* Variable used to return value of
     * communication routine*/
    s32 comres = BNO055_ERROR;

    /* variable used to set the power mode of the sensor*/
    u8 power_mode = BNO055_INIT_VALUE;


    /* variable used to read the accel xyz data */
    struct bno055_accel_t accel_xyz;

    /*********read raw mag data**********/
    /* structure used to read the mag xyz data */
    struct bno055_mag_t mag_xyz;

    /**********read raw gyro data**********/
    /* structure used to read the gyro xyz data */
    struct bno055_gyro_t gyro_xyz;

    /*************read raw Euler data************/
    /* structure used to read the euler hrp data */
    struct bno055_euler_t euler_hrp;

    /************read raw quaternion data*************/
    /* structure used to read the quaternion wxyz data */
    struct bno055_quaternion_t quaternion_wxyz;

    /************read raw linear acceleration data**********/
    /* structure used to read the linear accel xyz data */
    struct bno055_linear_accel_t linear_acce_xyz;

    /*****************read raw gravity sensor data****************/
    /* structure used to read the gravity xyz data */
    struct bno055_gravity_t gravity_xyz;

    /*************read accel converted data***************/
    /* structure used to read the accel xyz data output as m/s2 or mg */
    struct bno055_accel_double_t d_accel_xyz;

    /******************read mag converted data********************/
    /* structure used to read the mag xyz data output as uT*/
    struct bno055_mag_double_t d_mag_xyz;

    /*****************read gyro converted data************************/
    /* structure used to read the gyro xyz data output as dps or rps */
    struct bno055_gyro_double_t d_gyro_xyz;

    /*******************read euler converted data********************/
    /* variable used to read the euler h data output
     * as degree or radians*/
    double d_euler_data_h = BNO055_INIT_VALUE;
    /* variable used to read the euler r data output
     * as degree or radians*/
    double d_euler_data_r = BNO055_INIT_VALUE;
    /* variable used to read the euler p data output
     * as degree or radians*/
    double d_euler_data_p = BNO055_INIT_VALUE;
    /* structure used to read the euler hrp data output
     * as as degree or radians */
    struct bno055_euler_double_t d_euler_hpr;
```

```
373
374     /*********read linear acceleration converted data**********/
375     /* structure used to read the linear accel xyz data output as m/s2*/
376     struct bno055_linear_accel_double_t d_linear_accel_xyz;
377
378     /*******************Gravity converted data*********************/
379     /* structure used to read the gravity xyz data output as m/s2*/
380     struct bno055_gravity_double_t d_gravity_xyz;
381
382     /*---------------------------------------------------------------------*
383      ********************** START INITIALIZATION ************************
384      *---------------------------------------------------------------------*/
385 #ifdef  BNO055_API
386
387     /*  Based on the user need configure I2C interface.
388      *  It is example code to explain how to use the bno055 API*/
389     I2C_routine();
390 #endif
391
392     /*---------------------------------------------------------------------*
393      *  This API used to assign the value/reference of
394      *  the following parameters
395      *  I2C address
396      *  Bus Write
397      *  Bus read
398      *  Chip id
399      *  Page id
400      *  Accel revision id
401      *  Mag revision id
402      *  Gyro revision id
403      *  Boot loader revision id
404      *  Software revision id
405      *---------------------------------------------------------------------*/
406     comres = bno055_init(&bno055);
407
408     /*  For initializing the BNO sensor it is required to the operation mode
409      * of the sensor as NORMAL
410      * Normal mode can set from the register
411      * Page - page0
412      * register - 0x3E
413      * bit positions - 0 and 1*/
414     power_mode = BNO055_POWER_MODE_NORMAL;
415
416     /* set the power mode as NORMAL*/
417     comres += bno055_set_power_mode(power_mode);
418
419     /*---------------------------------------------------------------*
420      ************************* END INITIALIZATION *************************
421      *---------------------------------------------------------------*/
422
423     /************************* START READ RAW SENSOR DATA****************/
424
425     /*  Using BNO055 sensor we can read the following sensor data and
426      * virtual sensor data
427      * Sensor data:
428      * Accel
429      * Mag
430      * Gyro
431      * Virtual sensor data
432      * Euler
433      * Quaternion
434      * Linear acceleration
435      * Gravity sensor */
436
437     /*  For reading sensor raw data it is required to set the
438      * operation modes of the sensor
439      * operation mode can set from the register
440      * page - page0
441      * register - 0x3D
442      * bit - 0 to 3
443      * for sensor data read following operation mode have to set
444      * SENSOR MODE
445      * 0x01 - BNO055_OPERATION_MODE_ACCONLY
446      * 0x02 - BNO055_OPERATION_MODE_MAGONLY
447      * 0x03 - BNO055_OPERATION_MODE_GYRONLY
```

```
448     * 0x04 - BNO055_OPERATION_MODE_ACCMAG
449     * 0x05 - BNO055_OPERATION_MODE_ACCGYRO
450     * 0x06 - BNO055_OPERATION_MODE_MAGGYRO
451     * 0x07 - BNO055_OPERATION_MODE_AMG
452     * based on the user need configure the operation mode*/
453    comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_AMG);
454
455    /*  Raw accel X, Y and Z data can read from the register
456     * page - page 0
457     * register - 0x08 to 0x0D*/
458    comres += bno055_read_accel_xyz(&accel_xyz);
459
460    /*  Raw mag X, Y and Z data can read from the register
461     * page - page 0
462     * register - 0x0E to 0x13*/
463    comres += bno055_read_mag_xyz(&mag_xyz);
464
465    /*  Raw gyro X, Y and Z data can read from the register
466     * page - page 0
467     * register - 0x14 to 0x19*/
468    comres += bno055_read_gyro_xyz(&gyro_xyz);
469
470    /************************* END READ RAW SENSOR DATA****************/
471
472    /************************* START READ RAW FUSION DATA ********
473     * For reading fusion data it is required to set the
474     * operation modes of the sensor
475     * operation mode can set from the register
476     * page - page0
477     * register - 0x3D
478     * bit - 0 to 3
479     * for sensor data read following operation mode have to set
480     * FUSION MODE
481     * 0x08 - BNO055_OPERATION_MODE_IMUPLUS
482     * 0x09 - BNO055_OPERATION_MODE_COMPASS
483     * 0x0A - BNO055_OPERATION_MODE_M4G
484     * 0x0B - BNO055_OPERATION_MODE_NDOF_FMC_OFF
485     * 0x0C - BNO055_OPERATION_MODE_NDOF
486     * based on the user need configure the operation mode*/
487    comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_NDOF);
488
489    /*  Raw Euler H, R and P data can read from the register
490     * page - page 0
491     * register - 0x1A to 0x1E */
492    //comres += bno055_read_euler_h(&euler_data_h);
493    //comres += bno055_read_euler_r(&euler_data_r);
494    //comres += bno055_read_euler_p(&euler_data_p);
495    comres += bno055_read_euler_hrp(&euler_hrp);
496
497    /*  Raw Quaternion W, X, Y and Z data can read from the register
498     * page - page 0
499     * register - 0x20 to 0x27 */
500    //comres += bno055_read_quaternion_w(&quaternion_data_w);
501    //comres += bno055_read_quaternion_x(&quaternion_data_x);
502    //comres += bno055_read_quaternion_y(&quaternion_data_y);
503    //comres += bno055_read_quaternion_z(&quaternion_data_z);
504    comres += bno055_read_quaternion_wxyz(&quaternion_wxyz);
505
506    /*  Raw Linear accel X, Y and Z data can read from the register
507     * page - page 0
508     * register - 0x28 to 0x2D */
509    //comres += bno055_read_linear_accel_x(&linear_accel_data_x);
510    //comres += bno055_read_linear_accel_y(&linear_accel_data_y);
511    //comres += bno055_read_linear_accel_z(&linear_accel_data_z);
512    comres += bno055_read_linear_accel_xyz(&linear_acce_xyz);
513
514    /*  Raw Gravity sensor X, Y and Z data can read from the register
515     * page - page 0
516     * register - 0x2E to 0x33 */
517    //comres += bno055_read_gravity_x(&gravity_data_x);
518    //comres += bno055_read_gravity_y(&gravity_data_y);
519    //comres += bno055_read_gravity_z(&gravity_data_z);
520    comres += bno055_read_gravity_xyz(&gravity_xyz);
521
522    /************************* END READ RAW FUSION DATA  ************/
```

```
523   /*****************START READ CONVERTED SENSOR DATA****************/
524
525   /*  API used to read accel data output as double  - m/s2 and mg
526    * float functions also available in the BNO055 API */
527   //comres += bno055_convert_double_accel_x_msq(&d_accel_datax);
528   //comres += bno055_convert_double_accel_x_mg(&d_accel_datax);
529   //comres += bno055_convert_double_accel_y_msq(&d_accel_datay);
530   //comres += bno055_convert_double_accel_y_mg(&d_accel_datay);
531   //comres += bno055_convert_double_accel_z_msq(&d_accel_dataz);
532   //comres += bno055_convert_double_accel_z_mg(&d_accel_dataz);
533   comres += bno055_convert_double_accel_xyz_msq(&d_accel_xyz);
534   comres += bno055_convert_double_accel_xyz_mg(&d_accel_xyz);
535
536   /*  API used to read mag data output as double  - uT(micro Tesla)
537    * float functions also available in the BNO055 API */
538   //comres += bno055_convert_double_mag_x_uT(&d_mag_datax);
539   //comres += bno055_convert_double_mag_y_uT(&d_mag_datay);
540   //comres += bno055_convert_double_mag_z_uT(&d_mag_dataz);
541   comres += bno055_convert_double_mag_xyz_uT(&d_mag_xyz);
542
543   /*  API used to read gyro data output as double  - dps and rps
544    * float functions also available in the BNO055 API */
545   //comres += bno055_convert_double_gyro_x_dps(&d_gyro_datax);
546   //comres += bno055_convert_double_gyro_y_dps(&d_gyro_datay);
547   //comres += bno055_convert_double_gyro_z_dps(&d_gyro_dataz);
548   //comres += bno055_convert_double_gyro_x_rps(&d_gyro_datax);
549   //comres += bno055_convert_double_gyro_y_rps(&d_gyro_datay);
550   //comres += bno055_convert_double_gyro_z_rps(&d_gyro_dataz);
551   comres += bno055_convert_double_gyro_xyz_dps(&d_gyro_xyz);
552   //comres += bno055_convert_double_gyro_xyz_rps(&d_gyro_xyz);
553
554   /*  API used to read Euler data output as double  - degree and radians
555    * float functions also available in the BNO055 API */
556   comres += bno055_convert_double_euler_h_deg(&d_euler_data_h);
557   comres += bno055_convert_double_euler_r_deg(&d_euler_data_r);
558   comres += bno055_convert_double_euler_p_deg(&d_euler_data_p);
559   //comres += bno055_convert_double_euler_h_rad(&d_euler_data_h);
560   //comres += bno055_convert_double_euler_r_rad(&d_euler_data_r);
561   //comres += bno055_convert_double_euler_p_rad(&d_euler_data_p);
562   comres += bno055_convert_double_euler_hpr_deg(&d_euler_hpr);
563   //comres += bno055_convert_double_euler_hpr_rad(&d_euler_hpr);
564
565   /*  API used to read Linear acceleration data output as m/s2
566    * float functions also available in the BNO055 API */
567   //comres += bno055_convert_double_linear_accel_x_msq(&d_linear_accel_datax);
568   //comres += bno055_convert_double_linear_accel_y_msq(&d_linear_accel_datay);
569   //comres += bno055_convert_double_linear_accel_z_msq(&d_linear_accel_dataz);
570   comres += bno055_convert_double_linear_accel_xyz_msq(&d_linear_accel_xyz);
571
572   /*  API used to read Gravity sensor data output as m/s2
573    * float functions also available in the BNO055 API */
574   //comres += bno055_convert_gravity_double_x_msq(&d_gravity_data_x);
575   //comres += bno055_convert_gravity_double_y_msq(&d_gravity_data_y);
576   //comres += bno055_convert_gravity_double_z_msq(&d_gravity_data_z);
577   comres += bno055_convert_double_gravity_xyz_msq(&d_gravity_xyz);
578
579   /*---------------------------------------------------------------------*
580    ************************ START DE-INITIALIZATION ***********************
581    *---------------------------------------------------------------------*/
582
583   /*  For de - initializing the BNO sensor it is required
584    * to the operation mode of the sensor as SUSPEND
585    * Suspend mode can set from the register
586    * Page - page0
587    * register - 0x3E
588    * bit positions - 0 and 1*/
589   //power_mode = BNO055_POWER_MODE_SUSPEND;
590
591   /* set the power mode as SUSPEND*/
592   //comres += bno055_set_power_mode(power_mode);
593   comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_NDOF);
594   /*---------------------------------------------------------------------*
595   ************************ END DE-INITIALIZATION **********************
596   *---------------------------------------------------------------------*/
597   return comres;
```

```
598 }
599
```

```
/**
* Copyright (c) 2020 Bosch Sensortec GmbH. All rights reserved.
*
* BSD-3-Clause
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.
*
* 3. Neither the name of the copyright holder nor the names of its
*    contributors may be used to endorse or promote products derived from
*    this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
* COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
* @file bno055_support.c
* @date 10/01/2020
* @version  2.0.6
*
*/

/*---------------------------------------------------------------------------*
*  Includes
*---------------------------------------------------------------------------*/
#include "bno055.h"

#define  BNO055_API

#define  FLAG_MEAS_ON    1
#define  FLAG_MEAS_OFF   0
/*---------------------------------------------------------------------------*
*  The following APIs are used for reading and writing of
*   sensor data using I2C communication
*---------------------------------------------------------------------------*/
#ifdef  BNO055_API
#define BNO055_I2C_BUS_WRITE_ARRAY_INDEX ((u8)1)

/*  \Brief: The API is used as I2C bus read
 *  \Return : Status of the I2C read
 *  \param dev_addr : The device address of the sensor
 *  \param reg_addr : Address of the first register,
 *   will data is going to be read
 *  \param reg_data : This data read from the sensor,
 *   which is hold in an array
 *  \param cnt : The no of byte of data to be read
 */
s8 BNO055_I2C_bus_read(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt);

/*  \Brief: The API is used as SPI bus write
 *  \Return : Status of the SPI write
 *  \param dev_addr : The device address of the sensor
 *  \param reg_addr : Address of the first register,
 *   will data is going to be written
 *  \param reg_data : It is a value hold in the array,
```

```
 *  will be used for write the value into the register
 *  \param cnt : The no of byte of data to be write
 */
s8 BNO055_I2C_bus_write(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt);

/*
 * \Brief: I2C init routine
 */
s8 I2C_routine(void);

/*  Brief : The delay routine
 *  \param : delay in ms
 */
void BNO055_delay_msek(u32 msek);

#endif

/**********************End of I2C APIs declarations***********************/

/* This API is an example for reading sensor data
 *  \param: None
 *  \return: communication result
 */
s32 bno055_init_readout(void);

s32 bno055_read_routine(s_bno055_data *data);

/*-------------------------------------------------------------------------*
 *  struct bno055_t parameters can be accessed by using BNO055
 *  BNO055_t having the following parameters
 *  Bus write function pointer: BNO055_WR_FUNC_PTR
 *  Bus read function pointer: BNO055_RD_FUNC_PTR
 *  Burst read function pointer: BNO055_BRD_FUNC_PTR
 *  Delay function pointer: delay_msec
 *  I2C address: dev_addr
 *  Chip id of the sensor: chip_id
 *-------------------------------------------------------------------------*/
struct bno055_t bno055;

```

```c
/* ************************************************************************** */
/** Descriptive File Name

  @Company
    Company Name

  @File Name
    filename.c

  @Summary
    Brief description of the file.

  @Description
    Describe the purpose of this file.
 */
/* ************************************************************************** */

/* ************************************************************************** */
/* ************************************************************************** */
/* Section: Included Files                                                    */
/* ************************************************************************** */
/* ************************************************************************** */
#include "Mc32_PressAdc.h"
#include "app.h"
#include "peripheral/adc/plib_adc.h"
/* This section lists the other files that are included in this file.
 */

/* TODO:  Include other files here if needed. */


/* ************************************************************************** */
/* ************************************************************************** */
/* Section: File Scope or Global Data                                         */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

/* ************************************************************************** */



/* ************************************************************************** */
/* ************************************************************************** */
// Section: Local Functions                                                   */
/* ************************************************************************** */
/* ************************************************************************** */

/* ************************************************************************** */


/* ************************************************************************** */
/* ************************************************************************** */
// Section: Interface Functions                                               */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

// *****************************************************************************

void Press_InitADC (void){
    //Configuration de l'adresse choisi ADC
    PLIB_ADC_InputScanMaskAdd(ADC_ID_1, ADC_AN_SCAN_ADDRES);
    // Configure l'ADC en mode alterné
    PLIB_ADC_ResultFormatSelect(ADC_ID_1, ADC_RESULT_FORMAT_INTEGER_16BIT);
    //Choisir ce mode -> Buffer alterné
```

```
73    PLIB_ADC_ResultBufferModeSelect(ADC_ID_1, ADC_BUFFER_MODE_TWO_8WORD_BUFFERS);
74    //mode multiplexage
75    PLIB_ADC_SamplingModeSelect(ADC_ID_1, ADC_SAMPLING_MODE_MUXA);
76
77    //la lecture des ADC est cadensée par le timer interne
78    PLIB_ADC_ConversionTriggerSourceSelect(ADC_ID_1, ADC_CONVERSION_TRIGGER_INTERNAL_COUNT);
79    //Tension de réference de l'ADC alimentation 3V3
80    PLIB_ADC_VoltageReferenceSelect(ADC_ID_1, ADC_REFERENCE_VDD_TO_AVSS);
81    PLIB_ADC_SampleAcquisitionTimeSet(ADC_ID_1, 0x1F);
82    PLIB_ADC_ConversionClockSet(ADC_ID_1, SYS_CLK_FREQ, 32);
83
84    //ADC fait 3 mesures par interruption (car 3 canaux utilisés) -> adapter en fct des ADC utilisés
85    PLIB_ADC_SamplesPerInterruptSelect(ADC_ID_1, ADC_1SAMPLE_PER_INTERRUPT);
86    //active le scan en mode multiplexage des entrées AN
87    PLIB_ADC_MuxAInputScanEnable(ADC_ID_1);
88
89    // Enable the ADC module
90    PLIB_ADC_Enable(ADC_ID_1);
91
92 }
93
94 S_ADCResults Press_ReadAllADC( void ) {
95    //structure de valeurs brutes des ADCs
96    volatile S_ADCResults rawResult;
97    // Traitement buffer
98    ADC_RESULT_BUF_STATUS BufStatus;
99    //stop sample/convert
100    PLIB_ADC_SampleAutoStartDisable(ADC_ID_1);
101    // traitement avec buffer alterné
102    BufStatus = PLIB_ADC_ResultBufferStatusGet(ADC_ID_1);
103    //Buffer 8 bits -> 0 à 7 -> expliqué après
104    if (BufStatus == ADC_FILLING_BUF_0TO7) {
105       rawResult.AN3 = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 0);
106    }
107    else //Buffer 8 bits -> 8 à 15
108    {
109       rawResult.AN3 = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 8);
110    }
111    // Retablit Auto start sampling
112    PLIB_ADC_SampleAutoStartEnable(ADC_ID_1);
113
114    //retourner valeurs lue
115    return rawResult;
116 }
117
118 float Press_RawToVoltage(float raw){
119    float voltage = 0;
120    /* Raw ADC to voltage */
121    voltage = raw * ADC_RES;
122    /* Voltage before op-amp */
123    voltage = voltage / OPAMP_GAIN;
124    return voltage;
125 }
126
127 float Press_voltageToPressure(float voltage) {
128    float pressure = 0;
129       /* Convet voltage to pressure in bar */
130    pressure = ((voltage - V_MIN)*P_RANGE)/V_MAX;
131
132    return pressure;
133 }
134
135 float Press_readPressure( void ) {
136    //structure de valeurs brutes des ADCs
137    volatile S_ADCResults rawResult;
138    /* Voltage variable */
139    float voltage = 0;
140    /* Pressure variable */
141    float pressure = 0;
142    /* Read ADC */
143    rawResult = Press_ReadAllADC();
144    /* Convert raw data to voltage */
145    voltage = Press_RawToVoltage(rawResult.AN3);
146    /* Get the pressure from the voltage */
```

```
147     pressure = Press_voltageToPressure(voltage);
148 
149     return pressure;
150 }
151 
152 
153 /* ************************************************************************
154  End of File
155  */
156 
```

```
/* ************************************************************************** */
/** Descriptive File Name

  @Company
    Company Name

  @File Name
    filename.h

  @Summary
    Brief description of the file.

  @Description
    Describe the purpose of this file.
 */
/* ************************************************************************** */

#ifndef _PRESS_ADC_H    /* Guard against multiple inclusion */
#define _PRESS_ADC_H


/* ************************************************************************** */
/* ************************************************************************** */
/* Section: Included Files                                                    */
/* ************************************************************************** */
/* ************************************************************************** */
#include "app.h"
/* This section lists the other files that are included in this file.
 */

/* TODO:  Include other files here if needed. */


/* Provide C++ Compatibility */
#ifdef __cplusplus
extern "C" {
#endif


    /* ************************************************************************** */
    /* ************************************************************************** */
    /* Section: Constants                                                         */
    /* ************************************************************************** */
    /* ************************************************************************** */
#define     V_MIN               0.5
#define     V_MAX               4
#define     P_RANGE             10.0
#define     OPAMP_GAIN          0.5913
#define     ADC_RES             (3.3/1024)
#define     ADC_AN_SCAN_ADDRES  0x0008
    /* ************************************************************************** */


    // *****************************************************************************
    // *****************************************************************************
    // Section: Data Types
    // *****************************************************************************
    // *****************************************************************************
    typedef struct{
        uint16_t AN3;
    }S_ADCResults;
    // *****************************************************************************


    // *****************************************************************************
    // *****************************************************************************
    // Section: Interface Functions
    // *****************************************************************************
    // *****************************************************************************

    /* Convert voltage into pressure in [Bar] */
    float Press_voltageToPressure(float voltage);
```

```
73
74    /* Convert raw adc value to voltage */
75    float Press_RawToVoltage(float raw);
76
77    void Press_InitADC (void);
78
79    S_ADCResults Press_ReadAllADC( void );
80
81    float Press_readPressure( void );
82    // ********************************************************************************
83
84
85
86    /* Provide C++ Compatibility */
87 #ifdef __cplusplus
88 }
89 #endif
90
91 #endif /* _EXAMPLE_FILE_NAME_H */
92
93 /* ********************************************************************************
94  End of File
95  */
96
```

```
/* ************************************************************************** */
/** Descriptive File Name

  @Company
    ETML-ES

  @File Name
    sd_fat_gest.c

  @Summary
    SD card fat system management

  @Description
    SD card fat system management
 */
/* ************************************************************************** */

/* ************************************************************************** */
/* ************************************************************************** */
/* Section: Included Files                                                    */
/* ************************************************************************** */
/* ************************************************************************** */

/* This section lists the other files that are included in this file.
 */

#include "Mc32_sdFatGest.h"
#include <stdio.h>
#include "app.h"
#include "bno055_support.h"

/* ************************************************************************** */
/* ************************************************************************** */
/* Section: File Scope or Global Data                                         */
/* ************************************************************************** */
/* ************************************************************************** */

APP_FAT_DATA COHERENT_ALIGNED appFatData;
/* ************************************************************************** */
/** Descriptive Data Item Name

  @Summary
    Brief one-line summary of the data item.

  @Description
    Full description, explaining the purpose and usage of data item.
    <p>
    Additional description in consecutive paragraphs separated by HTML
    paragraph breaks, as necessary.
    <p>
    Type "JavaDoc" in the "How Do I?" IDE toolbar for more information on tags.

  @Remarks
    Any additional remarks
 */


/* ************************************************************************** */
/* ************************************************************************** */
// Section: Local Functions                                                   */
/* ************************************************************************** */
/* ************************************************************************** */

/* ************************************************************************** */




/* ************************************************************************** */
/* ************************************************************************** */
// Section: Interface Functions                                               */
/* ************************************************************************** */
/* ************************************************************************** */

void sd_fat_task ( void )
{
    /* The application task state machine */
    switch(appFatData.state)
    {
        case APP_MOUNT_DISK:
            if(SYS_FS_Mount("/dev/mmcblka1", "/mnt/myDrive", FAT, 0, NULL) != 0)
            {
                /* The disk could not be mounted. Try
                 * mounting again untill success. */

                appFatData.state = APP_MOUNT_DISK;
            }
            else
            {
                /* Mount was successful. Unmount the disk, for testing. */

                appFatData.state = APP_SET_CURRENT_DRIVE;
            }
            break;

        case APP_SET_CURRENT_DRIVE:
            if(SYS_FS_CurrentDriveSet("/mnt/myDrive") == SYS_FS_RES_FAILURE)
            {
                /* Error while setting current drive */
                appFatData.state = APP_ERROR;
            }
            else
            {
                /* Open a file for reading. */
                appFatData.state = APP_IDLE;
            }
            break;
```

```
107
108         case APP_WRITE_MEASURE_FILE:
109             appFatData.fileHandle = SYS_FS_FileOpen("MESURES.csv",
110                     (SYS_FS_FILE_OPEN_APPEND_PLUS));
111             if(appFatData.fileHandle == SYS_FS_HANDLE_INVALID)
112             {
113                 /* Could not open the file. Error out*/
114                 appFatData.state = APP_ERROR;
115             }
116             else
117             {
118                 /* Create a directory. */
119                 appFatData.state = APP_WRITE_TO_MEASURE_FILE;
120             }
121             break;
122
123         case APP_WRITE_TO_MEASURE_FILE:
124             /* If read was success, try writing to the new file */
125             if(SYS_FS_FileStringPut(appFatData.fileHandle, appFatData.data) == -1)
126             {
127                 /* Write was not successful. Close the file
128                  * and error out.*/
129                 SYS_FS_FileClose(appFatData.fileHandle);
130                 appFatData.state = APP_ERROR;
131             }
132             else
133             {
134                 appFatData.state = APP_CLOSE_FILE;
135             }
136             break;
137
138         case APP_CLOSE_FILE:
139             /* Close both files */
140             SYS_FS_FileClose(appFatData.fileHandle);
141              /* The test was successful. Lets idle. */
142             appFatData.state = APP_IDLE;
143             break;
144
145         case APP_IDLE:
146             /* The appliction comes here when the demo
147              * has completed successfully. Switch on
148              * green LED. */
149             //BSP_LEDOn(APP_SUCCESS_LED);
150             LED_ROff();
151             break;
152         case APP_ERROR:
153             /* The appliction comes here when the demo
154              * has failed. Switch on the red LED.*/
155             //BSP_LEDOn(APP_FAILURE_LED);
156             LED_ROn();
157             break;
158         default:
159             break;
160
161         case APP_UNMOUNT_DISK:
162             if(SYS_FS_Unmount("/mnt/myDrive") != 0)
163             {
164                 /* The disk could not be un mounted. Try
165                  * un mounting again untill success. */
166
167                 appFatData.state = APP_UNMOUNT_DISK;
168             }
169             else
170             {
171                 /* UnMount was successful. Mount the disk again */
172                 appFatData.state = APP_IDLE;
173             }
174             break;
175
176     }
177
178 //    SYS_FS_Tasks();
179 } //End of APP_Tasks
180
181 void sd_BNO_scheduleWrite (s_bno055_data * data)
182 {
183     /* If sd Card available */
184     if(appFatData.state == APP_IDLE)
185     {
186         /* Next state : write to file */
187         appFatData.state = APP_WRITE_MEASURE_FILE;
188         /* Write the buffer */
189         sprintf(appFatData.data, "%d;%d0;%f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%.4f;%d;%d;%d;%d;"
190                                 ,data->flagImportantMeas, (data->d_time), data->pressure, data->gravity.x, data->gravity.y, data->gravity.z, data->gyro.x, data->gyro.y, data->gyro.z
191                                 ,data->mag.x, data->mag.y, data->mag.z, data->linear_accel.x, data->linear_accel.y, data->linear_accel.z
192                                 ,data->euler.h, data->euler.p, data->euler.r, data->quaternion.w, data->quaternion.x, data->quaternion.y, data->quaternion.z);
193         /* Compute the number of bytes to send */
194         appFatData.nBytesToWrite = strlen(appFatData.data);
195     }
196 }
197
198 APP_FAT_STATES sd_getState( void )
199 {
200     return appFatData.state;
201 }
202
203 void sd_setState( APP_FAT_STATES newState )
204 {
205     appFatData.state = newState;
206 }
207
208 /* *****************************************************************************
209  End of File
210  */
211
```

```
/*******************************************************************************
  MPLAB Harmony Application Header File

  Company:
    Microchip Technology Inc.

  File Name:
    app.h

  Summary:
    This header file provides prototypes and definitions for the application.

  Description:
    This header file provides function prototypes and data type definitions for
    the application.  Some of these are required by the system (such as the
    "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
    internally by the application (such as the "APP_FAT_STATES" definition).  Both
    are defined here for convenience.
*******************************************************************************/

//DOM-IGNORE-BEGIN
/*******************************************************************************
Copyright (c) 2013-2014 released Microchip Technology Inc.  All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute
Software only when embedded on a Microchip microcontroller or digital signal
controller that is integrated into your product or third party product
(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for
additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
 *******************************************************************************/
//DOM-IGNORE-END

#ifndef _SD_FAT_GEST_H
#define _SD_FAT_GEST_H


// *****************************************************************************
// *****************************************************************************
// Section: Included Files
// *****************************************************************************
// *****************************************************************************

#include "app.h"

// *****************************************************************************
// *****************************************************************************
// Section: Type Definitions
// *****************************************************************************
// *****************************************************************************

#ifdef DRV_SDHC_USE_DMA
#define DATA_BUFFER_ALIGN           __attribute__((coherent, aligned(32)))
#else
#define DATA_BUFFER_ALIGN           __attribute__((aligned(32)))
#endif

// *****************************************************************************
/* Application States

```

```
  Summary:
    Application states enumeration

  Description:
    This enumeration defines the valid application states.  These states
    determine the behavior of the application at various times.
*/

typedef enum
{
        /* Application's state machine's initial state. */
        /* The app mounts the disk */
    APP_MOUNT_DISK = 0,

    /* Set the current drive */
    APP_SET_CURRENT_DRIVE,

        /* The app opens the file to read */
    APP_WRITE_MEASURE_FILE,

    /* The app reads from a file and writes to another file */
    APP_WRITE_TO_MEASURE_FILE,

    /* The app closes the file*/
    APP_CLOSE_FILE,

    /* The app closes the file and idles */
     APP_IDLE,

     /* An app error has occurred */
     APP_ERROR,

     /* Unmount disk */
     APP_UNMOUNT_DISK

} APP_FAT_STATES;


// *****************************************************************************
/* Application Data

  Summary:
    Holds application data

  Description:
    This structure holds the application's data.

  Remarks:
    Application strings and buffers are be defined outside this structure.
 */

typedef struct
{
    /* SYS_FS File handle for 1st file */
    SYS_FS_HANDLE      fileHandle;

    /* SYS_FS File handle for 2nd file */
    SYS_FS_HANDLE      fileHandle1;

    /* Application's current state */
    APP_FAT_STATES        state;

    /* Application data buffer */
    char               data[256] DATA_BUFFER_ALIGN;

    uint32_t           nBytesWritten;

    uint32_t           nBytesRead;

    uint32_t           nBytesToWrite;
} APP_FAT_DATA;


// *****************************************************************************
// *****************************************************************************
```

```
148 // Section: Application Callback Routines
149 // *****************************************************************************
150 // *****************************************************************************
151 /* These routines are called by drivers when certain events occur.
152 */
153
154
155 // *****************************************************************************
156 // *****************************************************************************
157 // Section: Application Initialization and State Machine Functions
158 // *****************************************************************************
159 // *****************************************************************************
160
161 /*******************************************************************************
162
163   Function:
164     void APP_Tasks ( void )
165
166   Summary:
167     MPLAB Harmony Demo application tasks function
168
169   Description:
170     This routine is the Harmony Demo application's tasks function.  It
171     defines the application's state machine and core logic.
172
173   Precondition:
174     The system and application initialization ("SYS_Initialize") should be
175     called before calling this.
176
177   Parameters:
178     None.
179
180   Returns:
181     None.
182
183   Example:
184     <code>
185     APP_Tasks();
186     </code>
187
188   Remarks:
189     This routine must be called from SYS_Tasks() routine.
190  */
191
192 void sd_fat_task ( void );
193
194 void sd_BNO_scheduleWrite (s_bno055_data * data);
195
196 APP_FAT_STATES sd_getState( void );
197
198 void sd_setState( APP_FAT_STATES newState );
199
200 #endif /* _APP_H */
201 /*******************************************************************************
202  End of File
203  */
204
205
```