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