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