

```

1  /*****
2  MPLAB Harmony Application Header File
3
4  Company:
5  ETML-ES
6
7  File Name:
8  MC32_sdFatGest.h
9
10 *****/
11
12 //DOM-IGNORE-BEGIN
13
14 //DOM-IGNORE-END
15
16 #ifndef _SD_FAT_GEST_H
17 #define _SD_FAT_GEST_H
18
19
20 // ****
21 // ****
22 // Section: Included Files
23 // ****
24 // ****
25
26 #include "app.h"
27 #include "GNSS/minmea.h"
28 #include "usart_FIFO.h"
29 // ****
30 // ****
31 // Section: Type Definitions
32 // ****
33 // ****
34
35 #ifdef DRV_SDHC_USE_DMA
36 #define DATA_BUFFER_ALIGN __attribute__((coherent, aligned(32)))
37 #else
38 #define DATA_BUFFER_ALIGN __attribute__((aligned(32)))
39 #endif
40
41 // ****
42 /* Application States
43
44 Summary:
45 Application states enumeration
46
47 Description:
48 This enumeration defines the valid application states. These states
49 determine the behavior of the application at various times.
50 */
51
52 typedef enum
53 {
54     /* Application's state machine's initial state. */
55     /* The app mounts the disk */
56     APP_MOUNT_DISK = 0,
57
58     /* Set the current drive */
59     APP_SET_CURRENT_DRIVE,
60
61     /* The app opens the file to read */
62     APP_WRITE_MEASURE_FILE,
63
64     /* The app reads from a file and writes to another file */
65     APP_WRITE_TO_MEASURE_FILE,
66
67     /* The app closes the file*/
68     APP_CLOSE_FILE,
69
70     /* The app closes the file and idles */
71     APP_IDLE,
72
73     /* An app error has occurred */

```

```

74     APP_ERROR,
75
76     /* Unmount disk */
77     APP_UNMOUNT_DISK
78
79 } APP_FAT_LOG_STATES;
80
81 typedef enum
82 {
83     /* Application's state machine's initial state. */
84     /* The app mounts the disk */
85     APP_CFG_MOUNT_DISK = 0,
86
87     /* Set the current drive */
88     APP_CFG_SET_CURRENT_DRIVE,
89
90     /* The app opens the file to read */
91     APP_CFG_OPEN_READ_CONFIG_FILE,
92
93     /* The app opens the file to read */
94     APP_CFG_READ_CONFIG_FILE,
95
96     /* The app opens the file to write */
97     APP_CFG_OPEN_WRITE_CONFIG_FILE,
98
99     /* Execute write */
100    APP_CFG_WRITE_CONFIG_FILE,
101
102    /* The app closes the file*/
103    APP_CFG_CLOSE_FILE,
104
105    /* The app closes the file and idles */
106    APP_CFG_IDLE,
107
108    /* An app error has occurred */
109    APP_CFG_ERROR,
110
111    /* Couldnt find config file */
112    APP_CFG_NO_CFG_FILE,
113
114    /* Unmount disk */
115    APP_CFG_UNMOUNT_DISK
116
117 } APP_FAT_CONFIG_STATES;
118
119
120 // *****
121 /* Application Data
122
123     Summary:
124         Holds application data
125
126     Description:
127         This structure holds the application's data.
128
129     Remarks:
130         Application strings and buffers are be defined outside this structure.
131 */
132
133 typedef struct
134 {
135     /* SYS_FS File handle for 1st file */
136     SYS_FS_HANDLE    fileMeasureHandle;
137
138     /* SYS_FS File handle for 2nd file */
139     SYS_FS_HANDLE    fileCfgHandle;
140
141     /* Application's current state */
142     APP_FAT_LOG_STATES    log_state;
143     APP_FAT_CONFIG_STATES    cfg_state;
144
145     /* Application data buffer */
146     char    data[FIFO_RX_SIZE+2] DATA_BUFFER_ALIGN;

```

```

147     /* Application config file */
148     char          cfg_data[200] DATA_BUFFER_ALIGN;
149
150     /* Filename variable */
151     char          fileName[15] DATA_BUFFER_ALIGN;
152
153     uint32_t      nBytesWritten;
154
155     uint32_t      nBytesRead;
156
157     uint32_t      nBytesToWrite;
158 } APP_FAT_DATA;
159
160
161 // *****
162 // *****
163 // Section: Application Callback Routines
164 // *****
165 // *****
166 /* These routines are called by drivers when certain events occur.
167 */
168
169
170 // *****
171 // *****
172 // Section: Application Initialization and State Machine Functions
173 // *****
174 // *****
175
176 /*****
177
178     Function:
179         void APP_Tasks ( void )
180
181     Summary:
182         MPLAB Harmony Demo application tasks function
183
184     Description:
185         This routine is the Harmony Demo application's tasks function.  It
186         defines the application's state machine and core logic.
187
188     Precondition:
189         The system and application initialization ("SYS_Initialize") should be
190         called before calling this.
191
192     Parameters:
193         None.
194
195     Returns:
196         None.
197
198     Example:
199         <code>
200         APP_Tasks();
201         </code>
202
203     Remarks:
204         This routine must be called from SYS_Tasks() routine.
205 */
206
207
208 void sd_fat_cfg_init(unsigned long *tGnss, unsigned long *tImu, bool *ledState,
uint32_t *tInactivePeriod);
209
210 void sd_fat_config_task ( bool init );
211 void sd_CFG_Write (uint32_t tLogGNSS_ms, uint32_t tLogIMU_ms, uint8_t ledState,
uint32_t tInactiveP, bool skipMount);
212 APP_FAT_CONFIG_STATES sd_cfgGetState( void );
213 void sd_cfgSetState( APP_FAT_CONFIG_STATES newState );
214 char* sd_cfgGetCfgBuffer( void );
215
216 void sd_fat_logging_task ( void );
217 APP_FAT_LOG_STATES sd_logGetState( void );

```

```
218 void sd_logSetState( APP_FAT_LOG_STATES newState );
219
220 void sd_IMU_scheduleWrite (s_bno055_data * data);
221
222 void sd_GNSS_scheduleWrite (minmea_messages * pGnssData);
223
224 void sd_fat_readDisplayFile(const char * fileName);
225
226
227 #endif /* _APP_H */
228 /*****
229 End of File
230 */
231
232
```