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

  Company:
    ETML-ES

  File Name:
    MC32_sdFatGest.h

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

//DOM-IGNORE-BEGIN

//DOM-IGNORE-END

#ifndef _SD_FAT_GEST_H
#define _SD_FAT_GEST_H


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

#include "app.h"
#include "GNSS/minmea.h"
#include "usart_FIFO.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_LOG_STATES;

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

    /* Set the current drive */
    APP_CFG_SET_CURRENT_DRIVE,

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

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

    /* The app opens the file to write */
    APP_CFG_OPEN_WRITE_CONFIG_FILE,

    /* Execute write */
    APP_CFG_WRITE_CONFIG_FILE,

    /* The app closes the file*/
    APP_CFG_CLOSE_FILE,

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

    /* An app error has occurred */
    APP_CFG_ERROR,

    /* Couldnt find config file */
    APP_CFG_NO_CFG_FILE,

    /* Unmount disk */
    APP_CFG_UNMOUNT_DISK

} APP_FAT_CONFIG_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      fileMeasureHandle;

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

    /* Application's current state */
    APP_FAT_LOG_STATES         log_state;
    APP_FAT_CONFIG_STATES      cfg_state;

    /* Application data buffer */
    char                 data[FIFO_RX_SIZE+2] DATA_BUFFER_ALIGN;
    /* Application config file */
    char                cfg_data[200] DATA_BUFFER_ALIGN;

    /* Filename variable */
    char                fileName[15] DATA_BUFFER_ALIGN;

    uint32_t           nBytesWritten;

    uint32_t           nBytesRead;

    uint32_t            nBytesToWrite;
} APP_FAT_DATA;


// *****************************************************************************
// *****************************************************************************
// Section: Application Callback Routines
// *****************************************************************************
// *****************************************************************************
/* These routines are called by drivers when certain events occur.
*/


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

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

  Function:
    void APP_Tasks ( void )

  Summary:
    MPLAB Harmony Demo application tasks function

  Description:
    This routine is the Harmony Demo application's tasks function.  It
    defines the application's state machine and core logic.

  Precondition:
    The system and application initialization ("SYS_Initialize") should be
    called before calling this.

  Parameters:
    None.

  Returns:
    None.

  Example:
    <code>
    APP_Tasks();
    </code>

  Remarks:
    This routine must be called from SYS_Tasks() routine.
 */


void sd_fat_cfg_init(unsigned long *tGnss, unsigned long *tImu, bool *ledState,
uint32_t *tInactivePeriod);

void sd_fat_config_task ( bool init );
void sd_CFG_Write (uint32_t tLogGNSS_ms, uint32_t tLogIMU_ms, uint8_t ledState,
uint32_t tInactiveP, bool skipMount);
APP_FAT_CONFIG_STATES sd_cfgGetState( void );
void sd_cfgSetState( APP_FAT_CONFIG_STATES newState );
char* sd_cfgGetCfgBuffer( void );

void sd_fat_logging_task ( void );
APP_FAT_LOG_STATES sd_logGetState( void );
```

```c
void sd_logSetState( APP_FAT_LOG_STATES newState );

void sd_IMU_scheduleWrite (s_bno055_data * data);

void sd_GNSS_scheduleWrite (minmea_messages * pGnssData);

void sd_fat_readDisplayFile(const char * fileName);


#endif /* _APP_H */
/*******************************************************************************
 End of File
 */


```