```c
/*******************************************************************************
  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;


// *****************************************************************************
// *****************************************************************************
// 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_task ( void );

void sd_BNO_scheduleWrite (s_bno055_data * data);

APP_FAT_STATES sd_getState( void );

void sd_setState( APP_FAT_STATES newState );

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


```