```
/*******************************************************************************
  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_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 _APP_H
#define _APP_H

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

#include "stepperDriver.h"
#include "lights.h"
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include "system_config.h"
#include "system_definitions.h"

// DOM-IGNORE-BEGIN
#ifdef __cplusplus  // Provide C++ Compatibility

extern "C" {

```

```
#endif
// DOM-IGNORE-END

// *****************************************************************************
// *****************************************************************************
// Section: Type Definitions
// *****************************************************************************
// *****************************************************************************

#define SYS_CLK 40000000

#define PWM_A_CMD_CH MCPWM_CHANNEL1
#define PWM_B_CMD_CH MCPWM_CHANNEL1
#define PWM_C_CMD_CH MCPWM_CHANNEL2
#define PWM_D_CMD_CH MCPWM_CHANNEL2
#define PWM_BL_CH MCPWM_CHANNEL3
#define PWM_BUZZER_CH MCPWM_CHANNEL4
#define PWM_DIM_CH MCPWM_CHANNEL6

#define MARGIN_LED_DELAY 50

/* Intensity in percent */
#define BACKLIGHT_INTENSITY_MIN 0
#define BACKLIGHT_INTENSITY_MAX 100


/* Value used to check if the EEPROM is already writent by this code */
#define CONTROL_VALUE 0x11223344

// *****************************************************************************
/* 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. */
    APP_STATE_INIT=0,
    APP_STATE_SERVICE_TASKS,
    APP_STATE_SERVICE_CAPTURE,
    APP_STATE_WAIT,
} APP_STATES;

typedef enum{

    SYS_STATE_MENU = 0,
    SYS_STATE_MANUAL,
    SYS_STATE_AUTO
} SYSTEM_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 enum{

    ALL_LED_DISABLE = 0,
    PWR_LED1,
    PWR_LED2,
    PWR_LED3,
    PWR_LED4,
    PWR_LED5,

}LED_ID;


typedef struct
{
    /* The application's current state */
    APP_STATES appState;
    SYSTEM_STATES systemState;
    LED_ID ledId;
    uint32_t msCounter;

    /* LED config */
    uint16_t lightIntensity;
    uint16_t timeBetweenPictures;
    uint16_t exposureDuration;

    /* Auto mode param */
    uint8_t angleBwEachSeq;

    uint32_t seqClock1_ms;
    uint32_t seqClock2_ms;
    bool isFiveShotsSeqEnable;
    bool isFullImaginSeqEnable;
    bool isFirstPass;
    uint16_t nbrOfShotsPerformed;
    uint8_t valSeq;

    uint16_t backLightIntensitiy;

    uint16_t buzzerIntensity;

} APP_DATA;

typedef struct
{
    bool state[4];
    bool isPressed;

} SW;

typedef struct{

    /* Motor data */
    int16_t     stepPerSec;
    uint16_t    stepPerTurn;
    uint16_t    gearValue;
    float       anglePerStep;

    /* LEDs data */
    uint16_t lightIntensity;
    uint16_t timeBetweenPictures;
    uint16_t exposureDuration;

    uint16_t backLightIntensitiy;

    /* Security value */
    uint32_t controlValue;

```

```
208       } DATA_IN_EEPROM;
209   // *****************************************************************************
210   // *****************************************************************************
211   // Section: Application Callback Routines
212   // *****************************************************************************
213   // *****************************************************************************
214   /* These routines are called by drivers when certain events occur.
215   */
216
217   // *****************************************************************************
218   // *****************************************************************************
219   // Section: Application Initialization and State Machine Functions
220   // *****************************************************************************
221   // *****************************************************************************
222
223   /*******************************************************************************
224     Function:
225       void APP_Initialize ( void )
226
227     Summary:
228        MPLAB Harmony application initialization routine.
229
230     Description:
231       This function initializes the Harmony application.  It places the
232       application in its initial state and prepares it to run so that its
233       APP_Tasks function can be called.
234
235     Precondition:
236       All other system initialization routines should be called before calling
237       this routine (in "SYS_Initialize").
238
239     Parameters:
240       None.
241
242     Returns:
243       None.
244
245     Example:
246       <code>
247       APP_Initialize();
248       </code>
249
250     Remarks:
251       This routine must be called from the SYS_Initialize function.
252   */
253
254   void APP_Initialize ( void );
255
256
257   /*******************************************************************************
258     Function:
259       void APP_Tasks ( void )
260
261     Summary:
262       MPLAB Harmony Demo application tasks function
263
264     Description:
265       This routine is the Harmony Demo application's tasks function.  It
266       defines the application's state machine and core logic.
267
268     Precondition:
269       The system and application initialization ("SYS_Initialize") should be
270       called before calling this.
271
272     Parameters:
273       None.
274
275     Returns:
276       None.
```

```
277
278     Example:
279       <code>
280       APP_Tasks();
281       </code>
282
283     Remarks:
284       This routine must be called from SYS_Tasks() routine.
285     */
286
287       void APP_Tasks( void );
288       void APP_Delay_ms(uint32_t ms);
289
290       void setBlIntensity(int32_t *backLightIntensitiy);
291       int32_t getBlIntensity(void);
292
293
294       void scanSwitch(void);
295       bool getSwitchEvent(void);
296
297       void initLcd(void);
298
299       void updateMcpwmDuty(void);
300
301
302
303   #endif /* _APP_H */
304
305   //DOM-IGNORE-BEGIN
306   #ifdef __cplusplus
307   }
308   #endif
309   //DOM-IGNORE-END
310
311   /*******************************************************************************
312    End of File
313    */
314
315
```