```
/*******************************************************************************
  MPLAB Harmony Application Source File

  Company:
    Microchip Technology Inc.

  File Name:
    app.c

  Summary:
    This file contains the source code for the MPLAB Harmony application.

  Description:
    This file contains the source code for the MPLAB Harmony application.  It
    implements the logic of the application's state machine and it may call
    API routines of other MPLAB Harmony modules in the system, such as drivers,
    system services, and middleware.  However, it does not call any of the
    system interfaces (such as the "Initialize" and "Tasks" functions) of any of
    the modules in the system or make any assumptions about when those functions
    are called.  That is the responsibility of the configuration-specific system
    files.
 *******************************************************************************/

// 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


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

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


#include <stddef.h>                     // Defines NULL
#include <stdbool.h>                    // Defines true
#include <stdlib.h>                     // Defines EXIT_FAILURE
#include "app.h"
#include "Gest_LED.h"
#include "dac_ad5620.h"
#include "Gestion_Filtre.h"

// *****************************************************************************
// *****************************************************************************
// Section: Global Data Definitions
// *****************************************************************************
// *****************************************************************************

// *****************************************************************************
/* Application Data

  Summary:
    Holds application data

  Description:
    This structure holds the application's data.

  Remarks:
    This structure should be initialized by the APP_Initialize function.

    Application strings and buffers are be defined outside this structure.
*/

APP_DATA appData;


uint8_t wait = 0;
uint8_t ON_OFF = 1;
uint8_t Color = 1;
uint16_t value = 1241 ;
uint32_t value_ADC = 0;
uint16_t Value_Mes [100000];
uint16_t Index = 0;
bool one_iteration = false;
// *****************************************************************************
// *****************************************************************************
// Section: Application Callback Functions
// *****************************************************************************
// *****************************************************************************

/* TODO:  Add any necessary callback functions.
*/
```

```
// *****************************************************************************
// *****************************************************************************
// Section: Application Local Functions
// *****************************************************************************
// *****************************************************************************


/* TODO:  Add any necessary local functions.
*/


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

/*******************************************************************************
  Function:
    void APP_Initialize ( void )

  Remarks:
    See prototype in app.h.
 */

void APP_Initialize ( void )
{
    /* Place the App state machine in its initial state. */
    appData.state = APP_STATE_INIT;



    /* TODO: Initialize your application's state machine and other
     * parameters.
     */
}



/******************************************************************************
  Function:
    void APP_Tasks ( void )

  Remarks:
    See prototype in app.h.
 */

void APP_Tasks ( void )
{

    /* Check the application's current state. */
    switch ( appData.state )
    {
```

```
        /* Application's initial state. */
        case APP_STATE_INIT:
        {
            bool appInitialized = true;
            DRV_TMR0_Start();
            DRV_TMR1_Start();
            All_LED_Off();

            DRV_ADC0_Open();
            DRV_ADC_Start();
            Dac_Init();


            if (appInitialized)
            {
                appData.state = APP_STATE_WAIT;
            }
            break;
        }

        case APP_STATE_SERVICE_TASKS:
        {

            DRV_TMR0_Stop();
            uint16_t Value_max = 0;
            uint16_t Value_min = 0;
            uint16_t Val_Zero = 0;
            uint16_t Tension_max  = 0;
            uint16_t Nb_ech_pos = 0;

            if (one_iteration == true)
            {
                Value_search(Value_Mes,&Value_max,&Value_min,&Val_Zero);

                //calcul de la tension max
                Tension_max = Value_max - Val_Zero;
                //mesure temps demi-pédiode
                Conv_Values (Value_Mes,&Nb_ech_pos,Val_Zero);
                //affichage LED d'aprés fréquence
                Calcul_Frequence_Led(Nb_ech_pos,Tension_max);

                Dac_Write(Val_Zero);

                Nb_ech_pos = 0;
            }
            else
            {
                one_iteration = true;
            }

            //LED0_GToggle();
```

```
            DRV_TMR0_Start();




             UpdateAppState(APP_STATE_WAIT);
            break;
        }

        case APP_STATE_WAIT:
        {
            //attend de reçevoir 10'000 échantillons
            break;
        }
        default:
        {
            /* TODO: Handle error in application's state machine. */
            break;
        }
    }
}
//fonction de la machine d'état
void UpdateAppState(APP_STATES newState)
{
    appData.state = newState;
}
//fonction de lecture de l'ADC

void Get_ADC_Values(uint16_t index)
{
    Value_Mes[index] = DRV_ADC_SamplesRead(2U);
}



/*******************************************************************************
 End of File
 */
```