```
/*******************************************************************************
  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 "app.h"
#include "Mc32DriverLcd.h"
#include "MenuJeu.h"
#include "Jeux.h"
#include "Buzzer.h"

#include "peripheral/usart/plib_usart.h"
#include <stdbool.h>
//#include <math.h>
#include <stdio.h>
#include "Mc32gest_RS232.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;


// *****************************************************************************
// *****************************************************************************
// 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 )
{
    uint8_t TxBuffer[20];       //Buffer de transmistion
    uint8_t RxBuffer[10];       //Buffer de reception
    uint8_t c ;                 //Reception de caractère

    static uint8_t i= 0;
    /* Check the application's current state. */
    switch ( appData.state )
    {
        /* Application's initial state. */
        case APP_STATE_INIT:
        {
            //Allumer la LED de vie
            LED_VIEOn();

            //Initialisation du Menu
            MENU_Initialize();

            //TIMER
            DRV_TMR0_Start();  //Timer 1 1ms
```

```
            DRV_TMR0_Start();  //TIMER 1 1ms

            //Ecrire le titre du projet sur Putty
            sprintf(TxBuffer, "Santiago-BuzzerWireGame\r\n");
            Send_Message(TxBuffer);

            //Aller dans l'etat WAIT
            APP_UpdateState(APP_STATE_WAIT);
            break;
        }


        case APP_STATE_WAIT:
        {
            //Ne rien faire
            break;
        }


        case APP_STATE_SERVICE_TASKS:
        {

            //Test Communication UART
            while(PLIB_USART_ReceiverDataIsAvailable(USART_ID_1))
            {
                //Recetpionne le caractere
                c = PLIB_USART_ReceiverByteReceive(USART_ID_1);
                //Enregistre dans le tb
                RxBuffer[i]=c;

                //Test de la terminaison de l'envoi
                if (RxBuffer[i] == '\r' || RxBuffer[i] == '\n')
                {
                    //Envoi Reception OK
                     sprintf(TxBuffer, "Reception Ok\r\n");
                     Send_Message(TxBuffer);
                     i=0;
                }
                else
                {
                    i++;
                }
            }

            //Test  Wire Detect
            //TouchDetectWire();

            //Apppel de la fonction pour la gestion du menu
            MENU_Execute();

            //Aller dans l'etat WAIT
            APP_UpdateState(APP_STATE_WAIT);
            break;
        }

        /* TODO: implement your application state machine.*/
```

```
        /* The default state should never be executed. */
        default:
        {
            /* TODO: Handle error in application's state machine. */
            break;
        }
    }
}

//Fonction pour mettre à jour les états
void APP_UpdateState(APP_STATES newState)
{
    appData.state = newState;
}



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