```
/*******************************************************************************
 System Interrupts File

  File Name:
    system_interrupt.c

  Summary:
    Raw ISR definitions.

  Description:
    This file contains a definitions of the raw ISRs required to support the
    interrupt sub-system.

  Summary:
    This file contains source code for the interrupt vector functions in the
    system.

  Description:
    This file contains source code for the interrupt vector functions in the
    system.  It implements the system and part specific vector "stub" functions
    from which the individual "Tasks" functions are called for any modules
    executing interrupt-driven in the MPLAB Harmony system.

  Remarks:
    This file requires access to the systemObjects global data structure that
    contains the object handles to all MPLAB Harmony module objects executing
    interrupt-driven in the system.  These handles are passed into the individual
    module "Tasks" functions to identify the instance of the module to maintain.
 *******************************************************************************/

// DOM-IGNORE-BEGIN
/*******************************************************************************
Copyright (c) 2011-2014 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
```

```
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 "system/common/sys_common.h"
#include "app.h"
#include "system_definitions.h"
#include "bsp.h"

// *****************************************************************************
// *****************************************************************************
// Section: System Interrupt Vector Functions
// *****************************************************************************
// *****************************************************************************


// ************** TIMER 1 **************
void __ISR(_TIMER_1_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance0(void)
{
    static int16_t waitCount = 0;
    static int16_t waitCountHour = 0;
    static int compteur = 0;

    DoDebounce (&DescrSW1, S_SW1);
    DoDebounce (&DescrSW2, S_SW2);
    DoDebounce (&DescrSW3, S_SW3);
    DoDebounce (&DescrSW4, S_SW4);

    //Appel de la fonction APP_UpdateState() après 3s, tous les 10 cycles
    if(waitCount >= 3000)
    {

        if(compteur >= 10)
        {
            APP_UpdateState (APP_STATE_SERVICE_TASKS);
            compteur = 0;
        }
        else
        {
            compteur ++;
        }
    }
    else
    {
        waitCount ++;
```

```
    }

    //Appel de la fonction "incrementSeconde" pour avoir l'heure à jour après réglages
    //toutes les 1 seconde
    if (waitCountHour >= 1000)
    {
        //Indique que l'on est dans le menu principal et que l'heure s'incrémente
        if(etatReglHour == 1)
        {
            incrementSeconde();
        }
        waitCountHour = 0;
    }
    else
    {
        waitCountHour ++;
    }
    PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_1);
}

// ************** TIMER 2 - 77.5kHz **************
void __ISR(_TIMER_2_VECTOR, ipl3AUTO) IntHandlerDrvTmrInstance1(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_2);
}

// ************** MODULATION **************
void __ISR(_TIMER_3_VECTOR, ipl4AUTO) IntHandlerDrvTmrInstance2(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_3);
    CMD_SW_R = 0;

}

 // OC3
void __ISR(_OUTPUT_COMPARE_3_VECTOR, ipl1AUTO) _IntHandlerDrvOCInstance0(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_OUTPUT_COMPARE_3);
}

//OC4
void __ISR(_OUTPUT_COMPARE_4_VECTOR, ipl4AUTO) _IntHandlerDrvOCInstance1(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_OUTPUT_COMPARE_4);
}
 /*******************************************************************************
 End of File
*/
```