```
/*******************************************************************************
 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
(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 "lcd_spi.h"
#include "pec12.h"

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

```
70  // Section: System Interrupt Vector Functions
71  // *****************************************************************************
72  // *****************************************************************************
73
74  extern APP_DATA appData;
75  extern STEPPER_DATA stepperData;
76
77  //-------------------------------------------------------------------------// TMR ID 1
78  /* This timer clocks the capture sequence */
79  /* Frequency = 1000Hz */
80  void __ISR(_TIMER_1_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance0(void){
81
82      PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_1);
83
84      /* Control sequence of the 5 LEDs */
85      if(appData.seqClock1_ms == 0){
86          appData.ledId = PWR_LED1;
87          startSimpleShotProcess();
88
89      } else if(appData.seqClock1_ms == 1 * appData.timeBetweenPictures){
90          appData.ledId = PWR_LED2;
91          startSimpleShotProcess();
92
93      } else if(appData.seqClock1_ms == 2 * appData.timeBetweenPictures){
94          appData.ledId = PWR_LED3;
95          startSimpleShotProcess();
96
97      } else if(appData.seqClock1_ms == 3 * appData.timeBetweenPictures){
98          appData.ledId = PWR_LED4;
99          startSimpleShotProcess();
100
101     } else if(appData.seqClock1_ms == 4 * appData.timeBetweenPictures){
102         appData.ledId = PWR_LED5;
103         startSimpleShotProcess();
104     }
105     if(appData.seqClock1_ms >= 5 * appData.timeBetweenPictures){
106
107         DRV_TMR0_Stop();
108         appData.seqClock1_ms = 0;
109         appData.isFiveShotsSeqEnable = false;
110         appData.valSeq = 1;
111         //return 1;
112     } else {
113     appData.seqClock1_ms++;
114     }
115 }
116
117 //-------------------------------------------------------------------------// TMR ID 2
118 /* This timer clocks the main state machine */
119 /* Frequency = 10000Hz */
120 void __ISR(_TIMER_2_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance1(void){
121
122     PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_2);
123
124     /* States machines update */
125     APP_UpdateAppState(APP_STATE_SERVICE_TASKS);
126 }
127
128 //-------------------------------------------------------------------------// TMR ID 3
129 /* This timer clocks the stepper sequence */
130 /* Variable frequency */
131 void __ISR(_TIMER_3_VECTOR, ipl4AUTO) IntHandlerDrvTmrInstance2(void){
132
133 //    SIGN_LED_CMDOff();
134     PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_3);
135
136     changeSpeed(getMyStepperStruct());
137     processStepper(getMyStepperStruct());
138 //    SIGN_LED_CMDOn();
```

```
139 }
140
141 //------------------------------------------------------------------------// TMR ID 4
142 /* This timer is used for the APP_Delay_ms() function */
143 /* Frequency = 1000Hz */
144 void __ISR(_TIMER_4_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance3(void){
145
146     PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_4);
147
148     /* Timer for ms delay */
149     appData.msCounter++;
150 }
151
152 //------------------------------------------------------------------------// TMR ID 5
153 /* Frequency = 1000Hz */
154 void __ISR(_TIMER_5_VECTOR, ipl3AUTO) IntHandlerDrvTmrInstance4(void)
155 {
156     PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_5);
157
158     //------------------------------------------------------------------------// Start
        of sequence
159     if(appData.seqClock2_ms == 0){
160
161         switch (appData.ledId){
162             /* Turn on LED */
163             case PWR_LED1:
164                 turnOffAllPwrLeds();
165                 LED1_CMDOn();
166                 break;
167
168             case PWR_LED2:
169                 turnOffAllPwrLeds();
170                 LED2_CMDOn();
171                 break;
172
173             case PWR_LED3:
174                 turnOffAllPwrLeds();
175                 LED3_CMDOn();
176                 break;
177
178             case PWR_LED4:
179                 turnOffAllPwrLeds();
180                 LED4_CMDOn();
181                 break;
182
183             case PWR_LED5:
184                 turnOffAllPwrLeds();
185                 LED5_CMDOn();
186                 break;
187         }
188     }
189     if(appData.seqClock2_ms == MARGIN_LED_DELAY){
190
191         /* Capture the target */
192         FOCUS_CMDOn();
193         TRIGGER_CMDOn();
194         appData.nbrOfShotsPerformed++;
195 //        SIGN_LED_CMDOn();
196     }
197     if(appData.seqClock2_ms == appData.exposureDuration + MARGIN_LED_DELAY){
198
199         TRIGGER_CMDOff();
200         FOCUS_CMDOff();
201     }
202     //------------------------------------------------------------------------// End of
        sequence
203     if(appData.seqClock2_ms >= appData.exposureDuration + (2 * MARGIN_LED_DELAY)){
204
205         /* Turn off TMR4 */
```

```
206             DRV_TMR4_Stop();
207             turnOffAllPwrLeds();
208             appData.seqClock2_ms = 0;
209             appData.ledId = ALL_LED_DISABLE;
210         } else {
211             appData.seqClock2_ms++;
212         }
213     }
214     /*******************************************************************************
215      End of File
216     */
217
```