Izabella Arredondo
CBZ848
ENEE 4710
HW 7 Source Code

mainasm_skeleton:

```
/*-------------------------------------------------------------------------------
HW #7:  LED Toggle Using Interrupts (Assembly-Code version)
Author: D. Loveless
Editor: I. Arredondo
Date of Last Update: 11/18/2016
Developed for MSP430FR6989

This program is used to introduce interrupts and ISRs, specifically using TIMER and PORT interrupts.
The RED LED is toggled at 0.5 sec intervals (1 sec blink rate), and accepts 4 button presses from P1.1,
speeding
up the blink rate by 2x each time, resetting on the last press (1 Hz, 2 Hz, 4 Hz, 8 Hz, 16 Hz, repeat). Each
button
press toggles the GREEN LED.
*-------------------------------------------------------------------------------*/
#include "msp430.h"
;-------------------------------------------------------------------------------
; Housekeeping
;-------------------------------------------------------------------------------
            name    main                    ; Module name

            rseg    CSTACK                  ; Pre-declaration of a segment
            rseg    CODE                    ; Place program in 'CODE' segment

            public  main                    ; Make the main visible

            extern  configureClocks         ; Declare sub-module configureClocks
            extern  configurePorts          ; Declare sub-module configurePorts
            extern  configureTimer          ; Declare sub-module configureTimer

;-------------------------------------------------------------------------------
; Main Program
;-------------------------------------------------------------------------------
main:       mov.w   #WDTPW+WDTHOLD,&WDTCTL      ; Stop watchdog timer
            mov.w   #750,R14                ; Store toggleCycles (used to set # of counts before entering
timer ISR)
            mov.w   #5,R13                  ; Store toggleTimes+1 (used to hold # of button presses used to
decrement timer)
            call    #configureClocks        ; Call configureClocks sub-module
            call    #configurePorts         ; Call configurePorts sub-module
            call    #configureTimer         ; Call configureTimer sub-module
            bis.w   #LPM3+GIE,SR            ; Enable global interrupts and go into LPM3
            nop                             ; No operation for when return from ISRs

;-------------------------------------------------------------------------------
```

```
; Port_1 ISR
;-------------------------------------------------------------------------------

;
; Code for Port_1 ISR
Port_1:   mov.w   #0x14B, R15          ; Delay cycles
          add.w   #0xFFFF, R15         ; by 1000
          bic.b   #0x2, P1IFG          ; Clear flag
          bit.b   #0x2, &P1IN          ; Check button press
          jc      Port_1               ; Reset if invalid press

          xor.b   #0x80, &P9OUT        ; Toggle Green LED
          dec.b   R13                  ; Decrement toggleTimes+1
          rra.w   R14                  ; Divide toggleCycles by 2, increase blink rate
          mov.w   R14, &TA0CCR0        ; Set TimerA to new toggleCycles
          cmp.w   #0, R13              ; Compare if on fifth button press
          jz      main                 ; If toggleCycles = 0, jump to main and reset blink rate
          reti                         ; Return from interrupt
;
;-------------------------------------------------------------------------------
; Timer_A0 ISR
;-------------------------------------------------------------------------------

;
; Code for Timer_A0 ISR
Timer_A0:     bic.w   #0x1, TA0CCTL0         ; Clear flag
          xor.b   #0x1, &P1OUT               ; Toggle Red LED
          reti                               ; Return from interrupt
;
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
          common  INTVEC                  ; Interrupt vectors
;-------------------------------------------------------------------------------
          org   TIMER0_A0_VECTOR          ; Timer0_A0 Vector
          dw    Timer_A0                  ; Define label for ISR

          org   PORT1_VECTOR              ; Port1 Vector
          dw    Port_1                    ; Define label for ISR

          end
```

configureTimer:

```
/*-------------------------------------------------------------------------------
Sub-routine configureTimer
Author: D. Loveless, 11/7/2016
Editor: I. Arredondo
  C-code equivalent:
  TA0CTL |= TASSEL__ACLK + MC__UP;          // Timer A0, source from ACLK @ 1.5KHz, count UP
mode
  TA0CTL &= ~ID__1;                         // Timer A0 divide-by-1 (1500 Hz)
  TA0CCR0 = toggleCycles;                   // Set the compare register 0 to 750 cycles (for a toggle
every 0.5 sec)
  TA0CCTL0 |= CCIE;                         // Enable timer A interrupt
*-------------------------------------------------------------------------------*/

#include "msp430.h"                         ; C-style pre-processor directive

            name    configureTimer          ; Module name
            public  configureTimer          ; Make the routine visible
            rseg    CODE                    ; Place program in 'CODE' segment

configureTimer:
            bis.w   #TASSEL__ACLK+MC__UP, &TA0CTL          ; Timer A0, source from ACLK @ 1.5KHz,
count UP mode
            bic.w   #ID__1, &TA0CTL         ; Timer A0 divide-by-1 (1500 Hz)
            mov.w   R14, &TA0CCR0           ; Set the compare register 0 to 750 cycles (for a toggle every 0.5
sec)
            bis.w   #CCIE, &TA0CCTL0        ; Enable timer A interrupt
            ret                             ; Return to main
            end
```

configurePorts:

```
/*-------------------------------------------------------------------------------
Sub-routine configurePorts
Author: D. Loveless, 11/7/2016
Editor: I. Arredondo
  C-code equivalent:
  PMMCTL0 = PMMPW;                  // Open PMM Module
  PM5CTL0 &= ~LOCKLPM5;             // Clear locked IO Pins
  P1DIR |= BIT0;                    // Set P1.0 (Red LED) to output
  P9DIR |= BIT7;                    // Configure P9.7 (Green LED) to be Output
  P1DIR &= ~BIT1;                   // Set P1.1 (Left push button) to input
  P1DIR &= ~BIT2;                   // Set P1.2 (Right push button) to input
  P1OUT |= BIT0;                    // Initialize Red LED to be ON
  P9OUT &= ~BIT7;                   // Initialize Green LED to be OFF
  P1REN |= BIT1 + BIT2;             // Enable a pull-up/down resistor on P1.1
  P1OUT |= BIT1 + BIT2;             // Set to be pull-up
  P1IE  = BIT1;                     // Enable the interrupt from P1.1 (disable all other Port 1 pin
interrupts)
  P1IES |= BIT1;                    // Set the interrupt to trigger on a hi-low transition
  P1IFG &= ~BIT1;                   // Clear the interrupt flag
*-------------------------------------------------------------------------------*/

#include "msp430.h"                 ; C-style pre-processor directive

            name    configurePorts  ; Module name
            public  configurePorts  ; Make the routine visible
            rseg    CODE            ; Place program in 'CODE' segment

configurePorts:
            mov.w   #WDTPW+WDTHOLD, &WDTCTL     ; Open PMM module
            bic.w   #LOCKLPM5, &PM5CTL0         ; Clear locked IO pins
            bis.b   #0x1, &P1DIR                ; Set P1.0 to output direction
            bis.b   #0x80, &P9DIR               ; Set P9.7 to output direction
            bis.b   #0x1, &P1OUT                ; Initialize P1.0 to be ON
            bis.b   #0x80, &P9OUT               ; Initialize P9.7 to be ON
            bic.b   #0x2, &P1DIR                ; Set P1.1 to input direction
            bis.b   #0x6, &P1REN                ; Enable pull-up/down resistors on P1.1 and
P1.2
            bis.b   #0x6, &P1OUT                ; Set to be pull-up
            bis.b   #0x2, &P1IE                 ; Enable interrupt on P1.1 (disable all other port
1 interrupts)
            bis.b   #0x2, &P1IES                ; Set interrupt to trigger on hi-low transition
            bic.b   #0x2, &P1IFG                ; Clear interrupt flag
            ret                                 ; Return to main
            end
```

configureClocks:

```
/*-------------------------------------------------------------------------------
Sub-routine configureClocks
Author: D. Loveless, 11/7/2016
Editor: I. Arredondo
  C-code equivalent:
  CSCTL0 = CSKEY;                                   // Write clock system password (0xA500)
  CSCTL1 |= DCORSEL + DCOFSEL2;                     // Set DCO to 16 MHz
  CSCTL2 |= SELM__DCOCLK + SELA__VLOCLK;            // Set MCLK to be 011 (DCO) and ACLK to be
001 (VLOCLK)
  CSCTL3 |= DIVM__1 + DIVA__8;                      // Divide ACLK by 8 (DIVA = 0011), MCLK by 1
(DIVM = 0000)
*-------------------------------------------------------------------------------*/

#include "msp430.h"                             ; C-style pre-processor directive

            name    configureClocks             ; Module name
            public  configureClocks             ; Make the routine visible
            rseg    CODE                        ; Place program in 'CODE' segment

configureClocks:
            mov.w   #0xA500, &CSCTL0                        ; Write clock system password
            bis.w   #DCORSEL + DCOFSEL2, &CSCTL1            ; Set DCO to 16 MHz
            bis.w   #SELM__DCOCLK + SELA__VLOCLK, &CSCTL2   ; Set MCLK to be 011 (DCO) and
ACLK to be 001 (VLOCLK)
            bis.w   #DIVM__1 + DIVA__8, &CSCTL3             ; Divide ACLK by 8 (DIVA = 011), MCLK
by 1 (DIVM = 000)
            ret                                             ; Return to main
            end
```