Q 4.2

Enabling the interrupt

```
//  TA1CCR0 = 0x4000;
//  TA1CTL = TASSEL_1 + MC_1 + TACLR;
//  TA1CCTL0 = CCIE;
```

Stop the Watch dog

```
WDTCTL = WDTPW + WDTHOLD;
```

Low Power Mode

```
__low_power_mode_3();
```

Q4.3: 8 LEDs turn on and off

```
// Introduction to the MSP430FR5739 system
// As provided this program will pulse a blue LED on the board
// Removing the necessary line comment will set up two LEDs to toggle between them
// using a software delay to make the display visible
//
// A better setup is to use a timer and interrupts
// Comment out the __delay_cycles function
// Restore all the other lines to facilitate the timer and interrupt operations
// This uses a low power mode to wait for an interrupt to occur.
// CAM 20130213
// ********************************************************************
#include "msp430fr5739.h"
unsigned int counter = 0;
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                     // Stop WDT

 // Set up a timer and enable the interrupt
  TA1CCR0 = 0x4000;
  TA1CTL = TASSEL_1 + MC_1 + TACLR;
  TA1CCTL0 = CCIE;                      // interrupt enabled

// Set up one LED to pulse
  P3DIR |= BIT7;
  P3OUT |= BIT7;
// Remove comments below to.....Toggle between two LEDs
  P3DIR |= BIT6;
  P3OUT &= BIT6;
  P3DIR |= BIT5;
  P3OUT &= BIT5;
  P3DIR |= BIT4;
  P3OUT &= BIT4;
  PJDIR |= BIT3;
  PJOUT &= BIT3;
  PJDIR |= BIT2;
  PJOUT &= BIT2;
  PJDIR |= BIT1;
  PJOUT &= BIT1;
  PJDIR |= BIT0;
  PJOUT &= BIT0;

  while(1)
```

```
 {
        __low_power_mode_3();       // Enter LPM3 w/ interrupt
        P3OUT ^= (BIT7 + BIT6 + BIT5 + BIT4);
        PJOUT ^=  (BIT3 + BIT2 + BIT1 + BIT0);
        counter++;
        __delay_cycles(100000);                 // Delay between transmissions
 }
}
// The interrupt service routine
 #pragma vector = TIMER1_A0_VECTOR
__interrupt void Timer1_A0_ISR(void)
{
 __low_power_mode_off_on_exit();
}
```

Q4.4: Turning on external LED (BLINK WITH OTHER INTERNAL LEDs)

```
// turning external LED on
  P3DIR |= BIT0;
  P3OUT |= BIT0;

  while(1)
  {
        __low_power_mode_3();       // Enter LPM3 w/ interrupt
        P3OUT ^= (BIT7 + BIT6 + BIT5 + BIT4);
        P3OUT ^= BIT0;
        PJOUT ^=  (BIT3 + BIT2 + BIT1 + BIT0);
        counter++;
        __delay_cycles(100000);                 // Delay between transmissions
  }
```

Q4.5: Turning on in order and reverse

```
// Introduction to the MSP430FR5739 system
// As provided this program will pulse a blue LED on the board
// Removing the necessary line comment will set up two LEDs to toggle between them
// using a software delay to make the display visible
//
// A better setup is to use a timer and interrupts
// Comment out the __delay_cycles function
// Restore all the other lines to facilitate the timer and interrupt operations
// This uses a low power mode to wait for an interrupt to occur.
// CAM 20130213
// ******************************************************************
```

```
#include "msp430fr5739.h"
unsigned int counter = 0;
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                     // Stop WDT

 // Set up a timer and enable the interrupt
  TA1CCR0 = 0x4000;
  TA1CTL = TASSEL_1 + MC_1 + TACLR;
  TA1CCTL0 = CCIE;                      // interrupt enabled

// Set up one LED to pulse
  P3DIR |= BIT7;
  P3OUT |= BIT7;
// Remove comments below to.....Toggle between two LEDs
  P3DIR |= BIT6;
  P3OUT &= BIT6;
  P3DIR |= BIT5;
  P3OUT &= BIT5;
  P3DIR |= BIT4;
  P3OUT &= BIT4;
  PJDIR |= BIT3;
  PJOUT &= BIT3;
  PJDIR |= BIT2;
  PJOUT &= BIT2;
  PJDIR |= BIT1;
  PJOUT &= BIT1;
  PJDIR |= BIT0;
  PJOUT &= BIT0;
// turning external LED on
  P3DIR |= BIT0;
  P3OUT |= BIT0;

  while(1)
  {
//        P3OUT ^= BIT0; // turns on external LED
          P3OUT = BIT7;
          __delay_cycles(100000);
          __low_power_mode_3();
          P3OUT = BIT6;
          __delay_cycles(100000);
          __low_power_mode_3();
          P3OUT = BIT5;
          __delay_cycles(100000);
```

```
        __low_power_mode_3();
        P3OUT = BIT4;
        __delay_cycles(100000);
        __low_power_mode_3();
        P3OUT &= ~BIT4; // Magically clear BIT
        PJOUT = BIT3;
        __delay_cycles(100000);
        __low_power_mode_3();
        PJOUT = BIT2;
        __delay_cycles(100000);
        __low_power_mode_3();
        PJOUT = BIT1;
        __delay_cycles(100000);
        __low_power_mode_3();
        PJOUT = BIT0;
        __delay_cycles(100000); // Delay between transmissions
        __low_power_mode_3();       // Enter LPM3 w/ interrupt
        PJOUT = BIT1;
        __delay_cycles(100000);
        __low_power_mode_3();
        PJOUT = BIT2;
        __delay_cycles(100000);
        __low_power_mode_3();
        PJOUT = BIT3;
        __delay_cycles(100000);
        __low_power_mode_3();
        PJOUT &= ~BIT3;
        P3OUT = BIT4;
        __delay_cycles(100000);
        __low_power_mode_3();
        P3OUT = BIT5;
        __delay_cycles(100000);
        __low_power_mode_3();
        P3OUT = BIT6;
        __delay_cycles(100000);
        __low_power_mode_3();
        P3OUT = BIT7;
        __delay_cycles(100000);
        __low_power_mode_3();
        counter++;
  }
}
// The interrupt service routine
 #pragma vector = TIMER1_A0_VECTOR
```

```
__interrupt void Timer1_A0_ISR(void)
{
 __low_power_mode_off_on_exit();
}
```