```
/*
 * File:   main.c
 * Author: jduffy5
 *
 * Created on May 1, 2013, 6:51 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <p33FJ64GS606.h>

_FOSCSEL(FNOSC_FRC)
_FOSC(FCKSM_CSECMD & OSCIOFNC_ON)
_FWDT(FWDTEN_OFF)
_FPOR(FPWRT_PWR128)
_FICD(ICS_PGD1 & JTAGEN_OFF)

#define green1LED LATDbits.LATD1
#define green2LED LATDbits.LATD9
#define green3LED LATDbits.LATD8
#define yellow1LED LATDbits.LATD4
#define yellow2LED LATDbits.LATD3
#define yellow3LED LATDbits.LATD2
#define red1LED LATDbits.LATD7
#define red2LED LATDbits.LATD6
#define red3LED LATDbits.LATD5
#define LED1T LATBbits.LATB3
#define LED1TR  LATBbits.LATB4
#define LED1BR  LATBbits.LATB5
#define LED1B   LATBbits.LATB1
#define LED1BL  LATBbits.LATB2
#define LED1TL  LATBbits.LATB6
#define LED1M   LATBbits.LATB7
#define LED2T   LATEbits.LATE3
#define LED2TR  LATEbits.LATE4
#define LED2BR  LATEbits.LATE6
#define LED2B   LATGbits.LATG8
#define LED2BL  LATGbits.LATG7
#define LED2TL  LATGbits.LATG6
#define LED2M   LATEbits.LATE7


//LED 1 Top RB3
//LED 1 Top Right RB4
//LED 1 Bot Right RB5
//LED 1 Bot RB1
//LED 1 Bot Left RB2
//LED 1 Top Left RB6
//LED 1 Middle RB7

//LED 2 Top RE3
//LED 2 Top Right RE4
//LED 2 Bot Right RE6
//LED 2 Bot RG8
//LED 2 Bot Left RG7
```

```
//LED 2 Top Left RG6
//LED 2 Middle RE7

int ADC_Readout;
int counts = 0;
int counts_between_breaths;
int detect_first_breath=0;
int silence_after_breath = 0;
double secs_between_breaths;
double breaths_per_sec=0;
int silence_counts;

/*
 *
 */
void __attribute__((__interrupt__,no_auto_psv)) _ADCP0Interrupt ()
 {          IFS6bits.ADCP0IF=0;
       counts = counts + 1;
            ADC_Readout = ADCBUF0;
       if (ADC_Readout > 655)
       {

          red1LED = 1;
          red2LED = 1;
          red3LED = 1;
          yellow1LED = 1;
          yellow2LED = 1;
          yellow3LED = 1;
          green1LED = 1;
          green2LED = 1;
          green3LED = 1;
       }
       else if (ADC_Readout > 650)
       {
          red1LED = 1;
          red2LED = 1;
          red3LED = 1;
          yellow1LED = 1;
          yellow2LED = 1;
          yellow3LED = 1;
          green1LED = 1;
          green2LED = 1;
          green3LED = 0;
       }
       else if (ADC_Readout > 645)
       {
          red1LED = 1;
          red2LED = 1;
          red3LED = 1;
          yellow1LED = 1;
          yellow2LED = 1;
          yellow3LED = 1;
          green1LED = 1;
          green2LED = 0;
          green3LED = 0;
       }
```

```
else if (ADC_Readout > 636)
{
   red1LED = 1;
   red2LED = 1;
   red3LED = 1;
   yellow1LED = 1;
   yellow2LED = 1;
   yellow3LED = 1;
   green1LED = 0;
   green2LED = 0;
   green3LED = 0;
}
else if (ADC_Readout > 627)
{
   red1LED = 1;
   red2LED = 1;
   red3LED = 1;
   yellow1LED = 1;
   yellow2LED = 1;
   yellow3LED = 0;
   green1LED = 0;
   green2LED = 0;
   green3LED = 0;
}
else if (ADC_Readout > 618)
{
   red1LED = 1;
   red2LED = 1;
   red3LED = 1;
   yellow1LED = 1;
   yellow2LED = 0;
   yellow3LED = 0;
   green1LED = 0;
   green2LED = 0;
   green3LED = 0;
}
else if (ADC_Readout > 609)
{
   red1LED = 1;
   red2LED = 1;
   red3LED = 1;
   yellow1LED = 0;
   yellow2LED = 0;
   yellow3LED = 0;
   green1LED = 0;
   green2LED = 0;
   green3LED = 0;
}
else if (ADC_Readout > 600)
{
   red1LED = 1;
   red2LED = 1;
   red3LED = 0;
   yellow1LED = 0;
   yellow2LED = 0;
   yellow3LED = 0;
```

```
    green1LED = 0;
    green2LED = 0;
    green3LED = 0;
}
else
{
    red1LED = 1;
    red2LED = 0;
    red3LED = 0;
    yellow1LED = 0;
    yellow2LED =0;
    yellow3LED = 0;
    green1LED = 0;
    green2LED =0;
    green3LED = 0;
    if (detect_first_breath == 1){
        silence_counts = silence_counts + 1;
        if (silence_counts > 450)
        {
            silence_after_breath = 1;
        }
    }

}

if (ADC_Readout > 618) {
    if (detect_first_breath ==0){
        counts = 0;
        detect_first_breath = 1;
    }
    if (silence_after_breath == 1){
        counts_between_breaths = counts;
        secs_between_breaths = 0.0022 * counts_between_breaths;
        breaths_per_sec = 60/secs_between_breaths;
        detect_first_breath = 0;
        silence_counts = 0;
        silence_after_breath = 0;
        //breaths_per_sec = 13;
        if (breaths_per_sec > 24){
    LED1T = 0;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 1;
    LED1BL = 0;
    LED1B = 0;
    LED1M=0;

    LED2T = 1;
    LED2TL = 0;
    LED2TR = 0;
    LED2BR = 0;
    LED2BL = 1;
    LED2B = 1;
    LED2M = 0;
}
        else if (breaths_per_sec >= 22){
```

```
    LED1T = 0;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 1;
    LED1BL = 0;
    LED1B = 0;
    LED1M=0;

    LED2T = 0;
    LED2TL = 1;
    LED2TR = 0;
    LED2BR = 1;
    LED2BL = 0;
    LED2B = 0;
    LED2M = 0;
}
       else if (breaths_per_sec >= 20){
    LED1T = 0;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 1;
    LED1BL = 0;
    LED1B = 0;
    LED1M=0;

    LED2T = 0;
    LED2TL = 0;
    LED2TR = 0;
    LED2BR = 0;
    LED2BL = 0;
    LED2B = 0;
    LED2M = 1;
}
else if (breaths_per_sec >=18){
    LED1T = 1;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 0;
    LED1BL = 1;
    LED1B = 1;
    LED1M=1;

    LED2T = 0;
    LED2TL = 0;
    LED2TR = 0;
    LED2BR = 0;
    LED2BL = 0;
    LED2B = 0;
    LED2M = 0;

}
 else if (breaths_per_sec >=16){
    LED1T = 1;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 0;
```

```
    LED1BL = 1;
    LED1B = 1;
    LED1M=1;

    LED2T = 0;
    LED2TL = 0;
    LED2TR = 1;
    LED2BR = 0;
    LED2BL = 0;
    LED2B = 0;
    LED2M = 0;

}
else if (breaths_per_sec >=14){
    LED1T = 1;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 0;
    LED1BL = 1;
    LED1B = 1;
    LED1M=1;

    LED2T = 1;
    LED2TL = 0;
    LED2TR = 0;
    LED2BR = 0;
    LED2BL = 1;
    LED2B = 1;
    LED2M = 0;

}
else if (breaths_per_sec >=12){
    LED1T = 1;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 0;
    LED1BL = 1;
    LED1B = 1;
    LED1M=1;

    LED2T = 0;
    LED2TL = 1;
    LED2TR = 0;
    LED2BR = 1;
    LED2BL = 0;
    LED2B = 0;
    LED2M = 0;

}
else if (breaths_per_sec >=10){
    LED1T = 1;
    LED1TL = 1;
    LED1TR = 0;
    LED1BR = 0;
    LED1BL = 1;
    LED1B = 1;
```

```
        LED1M=1;

        LED2T = 0;
        LED2TL = 0;
        LED2TR = 0;
        LED2BR = 0;
        LED2BL = 0;
        LED2B = 0;
        LED2M = 1;

    }
    else if (breaths_per_sec >=8) {
        LED1T = 0;
        LED1TL = 0;
        LED1TR = 0;
        LED1BR = 0;
        LED1BL = 0;
        LED1B = 0;
        LED1M=1;

        LED2T = 0;
        LED2TL = 0;
        LED2TR = 0;
        LED2BR = 0;
        LED2BL = 0;
        LED2B = 0;
        LED2M = 0;
    }
    else
    {
      LED1T = 0;
        LED1TL = 0;
        LED1TR = 0;
        LED1BR = 0;
        LED1BL = 0;
        LED1B = 0;
        LED1M=1;

        LED2T = 0;
        LED2TL = 0;
        LED2TR = 1;
        LED2BR = 0;
        LED2BL = 0;
        LED2B = 0;
        LED2M = 0;
    }
       }
    }


}
void init_PWM(void)
{
          PTCON2bits.PCLKDIV=0b110;          //PWM input clk devider =64
          PHASE1 = 0xFFFF;                                    // Timer1 period=max, this is just
convenient for validation
```

```
        PDC1 = 0x7FFF;                                  // Duty Cycle is 50%
        IOCON1bits.PENH = 1;            // PWM1H is controlled by PWM module
        IOCON1bits.PENL = 1;            // PWM1H is controlled by PWM module
        IOCON1bits.POLH = 0;            // Drive signals are active-high
        IOCON1bits.PMOD=1;                      // PWM pair works in Independent Mode
        PWMCON1bits.ITB = 1;                    // Select Independent Timebase mode
}
void init_ADC(void)
{
        ADCONbits.FORM = 0;             //Integer data format
        ADCONbits.EIE = 0;                      //Early Interrupt disabled
        ADCONbits.ORDER = 0;            //Convert even channel first
        ADCONbits.SEQSAMP = 0;          //Select simultaneous sampling
        ADCONbits.ADCS = 5;             //ADC clock = FADC/6 = 120MHz / 6 = 20MHz

        ADPCFGbits.PCFG0 = 0;           //select CH0 as analog pin
        ADPCFGbits.PCFG1 = 0;           //select CH1 as analog pin

        IFS6bits.ADCP0IF = 0;           //Clear ADC Pair 0 interrupt flag
        IPC27bits.ADCP0IP = 5;          //Set ADC Pair 0 interrupt priority
        IEC6bits.ADCP0IE = 1;           //Enable the ADC Pair 0 interrupt

        ADSTATbits.P0RDY = 0;                   //Clear Pair 0 data ready bit
        ADCPC0bits.IRQEN0 = 1;          //Enable ADC Interrupt pair 0
        ADCPC0bits.TRGSRC0 = 4;         //ADC Pair 0 triggered by PWM1 Trigger

        TRGCON1bits.DTM=1;              //dual trigger mode
        TRIG1bits.TRGCMP=0;             //Primary trig compare value
        STRIG1bits.STRGCMP=0xFFF;   //secondary trig compare value

        TRGCON1bits.TRGDIV = 0;         // Trigger generated every PWM cycle
        TRGCON1bits.TRGSTRT = 0;        // enable Trigger generated after 0 PWM cycles
        TRIG1 = 1;                                      // Trigger compare value
}
int main(int argc, char** argv) {
      /*Configure Oscillator to operate the device at 40MHz.
           * Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
           * Fosc= 7.37M*43/(2*2)=80Mhz for 7.37M input clock */

                // Configure PLL prescaler, PLL postscaler, PLL divisor
        PLLFBD=41;                                              // M = PLLFBD + 2
        CLKDIVbits.PLLPOST=0;                           // N1 = 2
        CLKDIVbits.PLLPRE=0;                            // N2 = 2

        __builtin_write_OSCCONH(0x01);          // New Oscillator FRC w/ PLL
        __builtin_write_OSCCONL(0x01);          // Enable Switch

     while(OSCCONbits.COSC != 0b001);
     while(OSCCONbits.LOCK != 1) {};

     ACLKCONbits.FRCSEL = 1;                    // FRC provides input for Auxiliary PLL (x16)
        ACLKCONbits.SELACLK = 1;                // Auxiliary Ocillator provides clock source for PWM
& ADC
        ACLKCONbits.APSTSCLR = 7;               // Divide Auxiliary clock by 1
        ACLKCONbits.ENAPLL = 1;                         // Enable Auxiliary PLL
```

```
    while(ACLKCONbits.APLLCK != 1);


    /*The FRC clock runs at 7.37 MHz. While setting the aux clock,
     there is an automatic 16x multiplication factor, which would
     result in a 120 MHz aux clock, so we divided that by 8 to
     get an aux clock of 15 MHz*/

    TRISD = 0;
  TRISBbits.TRISB1 = 0;
  TRISBbits.TRISB2 = 0;
  TRISBbits.TRISB3 = 0;
  TRISBbits.TRISB4 = 0;
  TRISBbits.TRISB5 = 0;
  TRISBbits.TRISB6 = 0;
  TRISBbits.TRISB7 = 0;
  TRISBbits.TRISB9 = 0;

  TRISE = 0;
  TRISG = 0;

  init_PWM();                                                    // PWM Setup
  init_ADC();
  PTCONbits.PTEN = 1;                                  // Enable the PWM
  ADCONbits.ADON = 1;                                        // Enable the ADC



  while(1);
}
```