Code for LE2361

12/2/2016

```
/* external interrupt code for PIC24 to light led
 * for a pulse duration

   T. Posbergh, 11/28/2016
   PIC24FJ64GA002                                  */

#include <xc.h>

#pragma config FNOSC=FRCPLL, POSCMOD=NONE
#pragma config FWDTEN=OFF, GCP=OFF, JTAGEN=OFF

int main(void) {

/* configure I/O pins */
    AD1PCFG = 0xFFFF;       // make pins digital
    TRISB = 0x7FFF;         // RB15 is output, RB<0:3> are input
    RPOR7bits.RP15R = 18;   // assign OC1 to RP15

/* configure Timer2    */
    T2CON = 0x0000;         // prescale 1:1,
    TMR2 = 0x0000;          // initialize to 0
    PR2 = 16000;            // for 1 kHz (exact is 15999)

/* output compare      */
    OC1CON = 0x0000;        // turn off OC1
    OC1R = 0x0000;          // initialize
    OC1RS = 0x0000;
    OC1CON = 0x0006;        // PWM, no fault

    IFS0bits.T2IF = 0;      // reset flag
    T2CONbits.TON = 1;      // enable Timer2

    OC1RS = (PORTB&0x000F)*1000;

    while(1){                                   // forever loop

        while(!IFS0bits.T2IF);                  // wait
        IFS0bits.T2IF = 0;                      // reset flag
        OC1RS  = (PORTB&0x000F)*1000;           // select duty cycle
    }

    return 0;
}
```

user polling

This example reads the 4-bits on RB<0:3> to determine one of 16 duty cycles (i.e. 0%, 6.25%, 12.5%, 18.75%, ..., 100%) with which to drive an LED.

```
/* external interrupt code for PIC24 to light led
 * for a pulse duration, uses interrupts

   T. Posbergh, 12/2/2016
   PIC24FJ64GA002                                */

#include <xc.h>

#pragma config FNOSC=FRCPLL, POSCMOD=NONE
#pragma config FWDTEN=OFF, GCP=OFF, JTAGEN=OFF

#undef  _ISR
#define _ISR __attribute__ ((interrupt, no_auto_psv))

int main(void) {

/* configure I/O pins    */
    AD1PCFG = 0xFFFF;       // make pins digital
    TRISB = 0x7FFF;         // RB15 is output, RB<0:3> are input
    RPOR7bits.RP15R = 18;   // assign OC1 to RP15

/* configure Timer2      */
    T2CON = 0x0000;         // prescale 1:1,
    TMR2 = 0x0000;          // initialize to 0
    PR2 = 16000;            // for 1 kHz (exact is 15999)

/* output compare        */
    OC1CON = 0x0000;        // turn off OC1
    OC1R = 0x0000;          // initialize
    OC1RS = 0x0000;
    OC1CON = 0x0006;        // PWM, no fault

/* configure interrupts */
    IPC1bits.T2IP = 4;      // Timer2 interrupt priority
    IFS0bits.T2IF = 0;      // reset flag
    IEC0bits.T2IE = 1;      // enable interrupts

    T2CONbits.TON = 1;      // enable Timer2

    OC1RS = (PORTB&0x000F)*1000;

    while(1);               // forever loop

    return 0;
}


/* Timer2 interrupt service routine  */
void _ISR _T2Interrupt(void)
{
        IFS0bits.T2IF = 0;              // reset flag
        OC1RS  = (PORTB&0x000F)*1000;   // select duty cycle
```

← used an ISR

}