

```

/*****
/* Program      : OPTICSW.C
/* Function     : Optical Switch Control Procedures
/* Author      : John F. Fitter B.E.
/*
/* These procedures are designed to operate 2 Optek Photologic switches (or equivalent).
/* The switches are connected to the ccp inputs and the software should generate interrupts
/* when the switches change state.
/*
/* Rev No.   Rev date   Test date   Test platform   Description
/* -----   -
/*    00     6jun98
/*
/*          Copyright © 1998 Eagle Air Australia Pty. Ltd. All rights reserved
*****/

#define _OPTICSW_C

#include <commdefs.h>
#include "main.h"
#include "opticsw.h"
#include "delays.h"

// Procedure to initialize the optical switches.

void init_opticsw() {

    set_led0(P_OFF); // ensure leds are turned off
    set_led1(P_OFF);
    led0_dir = B_OUT; // set the led drive pins to outputs
    led1_dir = B_OUT;
    TRISC |= 6; // ensure ccp pins are inputs
}
// Procedures to set and reset the optical switch for interrupt based period measurement.
// The technique uses the capture mode of the CCPx module and Timer 1 to do the timing.
// An auxillary register captures the Timer 1 overflows and extends the timing to 32 bits
// (1074 sec. at 16MHz, 2148 sec. at 8MHz). The Timer 1 prescaler is set so that the timer
// counts 250ns increments for a 16MHz clock and 500ns increments for an 8MHz clock.

void set_opticsw() {

    setup_timer_1(T1_INTERNAL | T1_DIV_BY_1); // setup the timer
    enable_timer_1(false); // turn the timer off
    set_timer1(0); // reset the timer low word
#ifdef SWITCH_0
    set_led0(P_ON); // turn on the led
    setup_ccp1(CCP_CAPTURE_FE); // set ccp to capture falling edge
    enable_interrupts(INT_TIMER1 | INT_CCP1); // assume peripheral interrupts enabled
    CCP1IF = false; // reccommended by Microchip
#else
    set_led1(P_ON);
    setup_ccp2(CCP_CAPTURE_FE);
    enable_interrupts(INT_TIMER1 | INT_CCP2);
    CCP2IF = false;
#endif // SWITCH_0
    enable_timer_1(true); // start the timer
}

void reset_opticsw() {

    enable_timer_1(false);
#ifdef SWITCH_0
    disable_interrupts(INT_TIMER1 | INT_CCP1);
    set_led0(P_OFF);
#else
    disable_interrupts(INT_TIMER1 | INT_CCP2);
    set_led1(P_OFF);
#endif // SWITCH_0
}

// Procedure to return the state of an optical switch. This is the slow method which does not
// use interrupts. If swnum is other than zero, switch 1 is addressed.

unsigned char get_opticsw(unsigned char swnum) {
    unsigned char swstat, ledstat;

    if(swnum) {

```

```
    ledstat = get_led1();           // get the led status
    set_led1(P_ON);                 // turn on the led
    delay_10us();                   // delay to stabilize led
    swstat = get_sense1();          // get the switch status
    set_led1(ledstat);              // restore the led status
} else {
    ledstat = get_led0();
    set_led0(P_ON);
    delay_10us();
    swstat = get_sense0();
    set_led0(ledstat);
}
return swstat;                     // return the switch status
}

// ***** EOF OPTICSW.C *****
```