```
/***************************************************************************
       tim_W.c           H8/300H Timer W control program

       Programmed by Kenji Arai/JH1PJL
       E-mail:     kenjia@sannet.ne.jp  jh1pjl@arrl.net
       URL:        http://www.page.sannet.ne.jp/kenjia/

              July     24,2004     start coding
              July     31,2004     LED control for IR detection
              August    1,2004    change old_ir as unsigned long (old int) bugfix
              August   12,2004     Xtal error compensation
              November  7,2004     Start with New PCB,

       Copyright (C) 2004 Kenji Arai/JH1PJL
       All rights reserved. Permission is granted to use, modify,or redistribute
    this software so long as it is not sold or exploited for profit.

       THIS SOFTWARE IS PROVIDED AS IS AND WITHOUT WARRANTY OF ANY KIND,
       EITHER EXPRESSED OR IMPLIED.
***************************************************************************/
#pragma interrupt (TimWISR(vect=21))

/*  -----< Include Files >---------------------------------------------- */
#include "3664f.h"
#include "const.h"
#include "rtm_H8_3664.h"

/*  -----< Function Prototype >----------------------------------------- */
void reset_timerW(void);
void rd_eep_osc(void);

/*  -----< Definition >------------------------------------------------- */
#define XTAL_ERR   32 // Xtal frequency error compensation
// calculation on Aug.5  = 9  -> 16=25-9->NG then 34=25+9
//               Aug.9  = 2  -> 34=25+9 32=34-2

#define B_CLK_20MS 40000// 20mS time base
#define DIF40MS      2      // Check interval bigger than 40mSec
#define MIN1200  2300 // 1.2mS pulse detect (min. 1150uS)
#define MAX1200  2500 //                   (max. 1250uS)

#define ALONE     -1  // State of detection  -- Noise and others
#define EXPCT_IN    5 //          --- expectation in real data
#define SURE_IN     0       //          --- at this time "It's sure"

#define LED_IR_DET PDR7.BIT.B6 // LED port for InfraRed detection indicator

/* -----< RAM assign >------------------------------------------------- */
unsigned long base_timer;          // base timer for measuring
int ir_state;                           // InfraRed sensor detect condition
// capture data and base timer data(@ captured)
unsigned int capture_b_data; // InfraRed
unsigned int capture_c_data; // Magnetic
unsigned int capture_d_data; // Switch
unsigned long btimr_b;                  // InfraRed
unsigned long btimr_c;                  // Magnetic
unsigned long btimr_d;                  // Switch
char osc_cmp;                           // Xtal freq. offset
```

```
/* -----< Control program >------------------------------------------- */
/*      ------------------------
        Timer W Initalize routine
        ------------------------ */
void reset_timerW(void)
{
        TCRW.BIT.CKS = 3;                   // select clock source as /8 (16MHz/8 = 500nS)
//      GRA = B_CLK_20MS  - XTAL_ERR;// timer sets 20mSec period (500nS*40,000=20mS)
        rd_eep_osc();                       // read osc. freq. compensation data from EEPROM
        if (osc_cmp == 0xff){
                osc_cmp = 0;
        }
        GRA = (unsigned int)(B_CLK_20MS + osc_cmp);
        TCRW.BIT.CCLR = 1;                  // counter clear by GRA
        TIERW.BIT.IMIEA = 1;          // enable output compare interrupt A
        TIERW.BIT.IMIEB = 1;          // enable input capture interrupt B
        TIERW.BIT.IMIEC = 1;          // enable input capture interrupt C
        TIERW.BIT.IMIED = 1;          // enable input capture interrupt D
        TIOR0.BIT.IOA  = 0;                 // set output compare mode
        TIOR0.BIT.IOB  = 5;                 // set input capture mode with falling edge
        TIOR1.BIT.IOC  = 5;                 // set input capture mode with falling edge
        TIOR1.BIT.IOD  = 5;                 // set input capture mode with falling edge
        TMRW.BIT.CTS = 1;                   // start TimerW count action
}

/*      ------------------------
        Timer W Interrupt handler
        ------------------------ */
void TimWISR( void )
{
        static unsigned long old_ir;
        static unsigned int  old_grb;

        if (TSRW.BIT.IMFA == 1){     //  ---- Base timer control -----
                TSRW.BIT.IMFA = 0;          // clear interrupt flag
                base_timer++;               // count up the timer
        }
        if (TSRW.BIT.IMFB == 1){     //  ---- InfraRed sensor control ----
                TSRW.BIT.IMFB = 0;          // clear interrupt flag
                capture_b_data = GRB; // save capture data
                if (base_timer  - old_ir > DIF40MS){
                        ir_state = ALONE;
                        LED_IR_DET = OFF_DR_HS;// LED OFF
                } else {
                        if ((capture_b_data  - old_grb > MIN1200) &&
                            (capture_b_data  - old_grb < MAX1200)){
                                if (ir_state == ALONE){
                                        ir_state = EXPCT_IN;
                                } else if (ir_state == SURE_IN){
                                        LED_IR_DET = ON_DR_HS;       // LED ON
                                        btimr_b = base_timer; // save base timer data
                                        reqed0();     // starting request for event driven task0

                                } else {
                                        ir_state --;
                                }
                        } else {
```

```
                        ir_state = ALONE;
                }
        }
        old_grb = capture_b_data;    // save old data
        old_ir = base_timer;         // save old data
    }
    if (TSRW.BIT.IMFC == 1){     //  ---- Magnetic sensor control  ----
        TSRW.BIT.IMFC = 0;           // clear interrupt flag
        reqed1();                           // starting request for event driven task1
        capture_c_data = GRC; // save capture data
        btimr_c = base_timer; // save base timer data
    }
    if (TSRW.BIT.IMFD == 1){     //  ---- Manual switch control  ----
        TSRW.BIT.IMFD = 0;           // clear interrupt flag
        reqed2();                           // starting request for event driven task2
        capture_d_data = GRD; // save capture data
        btimr_d = base_timer; // save base timer data
    }
}
```