```
/*****************************************************************************
       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,
              January  29,2005     Change base unit from 20mS to 1mS
              January  30,2005
              September 18,2005    Change to H8/3694F
              October  23,2005     Add LED drive interrupt

       Copyright (C) 2004,'05 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 "iodefine.h"
#include "const.h"
#include "task.h"
#include "rtm_H8_3664.h"

/*  ----< Function Prototype >-------------------------------------------*/
void reset_timerW(void);
void rd_eep_osc(void);
extern void led_drive(void);
extern void   reset_led(void);     // in tim_w.c

/*  ----< Definition >---------------------------------------------------*/
#define XTAL_ERR    0 // Xtal frequency error compensation

#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"

/* ----< 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_b_d;          // InfraRed
unsigned long btimr_b;                    // InfraRed
char osc_cmp;                             // Xtal freq. offset

/*  ----< Control program >----------------------------------------------*/
/*      ------------------------
       Timer W Initalize routine
        ------------------------*/
```

```
void reset_timerW(void)
{
//      TW.TCRW.BIT.CKS = 3;            // select clock source as /8 (16MHz/8 = 500nS)
        TW.TCRW.BIT.CKS = 3;            // select clock source as /8 (20MHz/8 = 400nS)
//?????????????????????????????????
//      rd_eep_osc();                           // read osc. freq. compensation data from EEPROM
//      if (osc_cmp == 0xff){
//              osc_cmp = 0;
//      }
        osc_cmp = 0;
//      TW.GRA = B_CLK_20MS - XTAL_ERR;// timer sets 20mSec period (400nS*50,000=20mS)
        TW.GRA = B_CLK_20MS + (unsigned int)osc_cmp;
        TW.TCRW.BIT.CCLR = 1;           // counter clear by GRA
        TW.TIERW.BIT.IMIEA = 1;                 // enable output compare interrupt A
        TW.TIERW.BIT.IMIEB = 1;                 // enable input capture interrupt B
        TW.TIERW.BIT.IMIEC = 1;                 // enable input capture interrupt C
        TW.TIOR0.BIT.IOA  = 0;                  // set output compare mode / A
        TW.TIOR0.BIT.IOB  = 5;                  // set input capture mode with falling edge /B
        TW.TIOR1.BIT.IOC  = 0;                  // set output compare mode /C
        TW.TMRW.BIT.CTS = 1;            // start TimerW count action
        /********** LED *********************************/
        reset_led();                            // in tim_w.c
}

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

        if (TW.TSRW.BIT.IMFC == 1){                     // ---- 7 segments LED drive ----
                TW.TSRW.BIT.IMFC = 0;                   // clear interrupt flag
                led_drive();
        }
        if (TW.TSRW.BIT.IMFA == 1){                     // ---- Base timer control -----
                TW.TSRW.BIT.IMFA = 0;                   // clear interrupt flag
                base_timer++;                                   // count up the timer
        }
        if (TW.TSRW.BIT.IMFB == 1){                     // ---- InfraRed sensor control ----
                TW.TSRW.BIT.IMFB = 0;                   // clear interrupt flag
                capture_b_data = TW.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;
                                        LED_IR_DET = OFF_DR_HS;                 // LED OFF
                                } else if (ir_state == SURE_IN){
                                        LED_IR_DET = ON_DR_HS;                  // LED ON
                                        if (TW.TCNT > capture_b_data){
                                                btimr_b = base_timer; // save base timer data
                                        } else {
                                                btimr_b = base_timer - 1;// save base timer data
                                        }
                                        capture_b_d = capture_b_data;
                                        reqed0();                       // request for event driven task0

                                } else {
                                        ir_state--;
                                        LED_IR_DET = OFF_DR_HS;                 // LED OFF
                                }
                        } else {
                                ir_state = ALONE;
                                LED_IR_DET = OFF_DR_HS;                 // LED OFF
                        }
                }
                old_grb = capture_b_data;               // save old data
                old_ir = base_timer;                    // save old data
        }
}
```