```
/************************************************************************
      Event and Cyclic Task's for
            Lap-Timer project using RTM_H8/3694F

            May      25,1998      New version on H8/3048F
            November  4,2001      Start 3664F project
            December 23,2001
            July     11,2004      Use RENESAS HEW functions
            August   12,2004      confirmed basic function then add new function
                                          likes Dataset change, Lapdata display and so on
            November  6,2004      Start with New PCB
            November 14,2004      Create Data entry mode
            January  29,2005      Change base unit from 20mS to 1mS
            January  30,2005      latest modification
            September18,2005      Change to H8/3694F
            October  1,2005       Delete most of function such as SW check
            October 23,2005

      Copyright (C) 1998, 2001,'04,'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 ed0_ISR */

/* -----< Include Files >--------------------------------------------- */
#include      "iodefine.h"
#include      "rtm_H8_3664.h"
#include      "sci.h"
#include      "const.h"
#include      "led.h"
#include      "ringbuf.h"
#include      "task.h"
#include      <machine.h>          // Use RENESAS HEW functions

/* -----< Function Prototype >---------------------------------------- */
void main( void );
void abort(void);

void ed0_main( void );
void ed1_main( void );
void ed2_main( void );
void ed3_main( void );
void cy0_init( void );
void cy0_main( void );
void cy1_init( void );
void cy1_main( void );
void cy2_init( void );
void cy2_main( void );
void cy3_init( void );
void cy3_main( void );
void cy4_init( void );
void cy4_main( void );
void cy5_init( void );
void cy5_main( void );
void cy6_init( void );
```

```
void cy6_main( void );
void cy7_init( void );
void cy7_main( void );

void general_init( void );
void port_init ( void );
void IdleNop( void );
void help_msg( void );
void PutCRLF( void );

int swchk_mode(unsigned char);
int swchk_enty(unsigned char);
void swich_status_check(unsigned char);
void eeprom_rd_req(unsigned char);

void conv_id(unsigned char, char *);
void sw_status(void);

void ad_init( void );
void ad_prnt( void );
void ad_out( int );

/*  ----< Function Prototype (Extern) >---------------------------------- */
extern void irq_enable( void );
extern void irq_disable( void );

extern void led_bright ( void );
extern void tim_dsp_to_led(unsigned long, unsigned char, unsigned char);
extern void hold_dsp_to_led( void );

extern char *eqstrf( char * , char * );

extern void opning_msg( void );
extern void cmdsrch( void );

extern void reset_timerV(void);
extern void reset_timerW(void);

/*  ----< Constant data in ROM >---------------------------------------- */
static char *const copyright =
"Arai,Kenji/JH1PJL(c)2004,'05 kenjia@sannet.ne.jp <H8/3694 ver.1 LED type>";

/*      Cyclic Task Period                  */
const unsigned char cyclic_period[8] = {
       T_20MS,        /* cyclic task no.0 */ // Swiches
       T_10MS,        /* cyclic task no.1 */ // Check operation mode
       T_100MS,       /* cyclic task no.2 */ //
       T_200MS,       /* cyclic task no.3 */ // LED Bright Control
       T_500MS,       /* cyclic task no.4 */ // NOT USE
       T_500MS,       /* cyclic task no.5 */ // NOT USE
       T_500MS,       /* cyclic task no.6 */ // NOT USE
       T_500MS        /* cyclic task no.7 */ // NOT USE
};

// ONLY FOR POWER ON INITIALIZE
const unsigned char cyclic_init_period[8] = {
       T_500MS,              /* cyclic task no.0 */
       T_500MS+T_10MS,       /* cyclic task no.1 */
```

```
        T_500MS+T_50MS,      /* cyclic task no.2 */
        T_500MS+T_50MS,      /* cyclic task no.3 */
        T_200MS,             /* cyclic task no.4 */
        T_200MS,             /* cyclic task no.5 */
        T_500MS,             /* cyclic task no.6 */
        T_500MS              /* cyclic task no.7 */
};

/* ----< RAM assign >---------------------------------------------------- */
unsigned long datap;
char rxBuf[20];                                 // RS232 receive char
char Buf;
char line[BFSZ], *lp;

// lap time
one_lap lap_for_led;                // Latest lap dat for LCD display
unsigned int latest_dt;                     // which tool is used for latest data
unsigned long lap_ir;               // Lap timer by using InfraRed sensor input
unsigned long lap_start;            // lap start time (from timer base)
unsigned long lap_best;             // best lap data
unsigned long lap_latest;           // latest data current
unsigned long lap_last1;            // latest data -1
unsigned long lap_last2;            // latest data -2
unsigned long time_ir_temp;

char lap_1st_one;                           // clear at power on, if 0 then 1st
char timbuf[20];
one_lap  lap_data_buf[BZ4LAP];      // lap data buffer from EEPROM
unsigned char lap_ptr_buf_top;      // pointer for lap_data_buf[] top data
unsigned char lap_ptr_rd_num;       // read pointer for lap_data_buf[]
unsigned char lap_ptr_reach_end;// reach end of buffer
unsigned char eepdt_md;                     // transfer complete flag, EEPROM to lap_data_buf[]
unsigned char eepdt_flg;            // transfer complete flag, EEPROM to lap_data_buf[]
unsigned char tmp_datain[6]; // work area for data manuplation

unsigned char mode_led_flg;         // Mode LED flag

// switch condition
static union {
        unsigned char BYTE;
        struct {
                        unsigned char modify:1;      //     Bit 7
                        unsigned char status:1;      //  Bit 6
                        unsigned char actual:1;      //  Bit 5
                        unsigned char dirctn:1;      //  Bit 4
                        unsigned char cunter:4;      //  Bit 3-0
        } BIT;
} sw_condition[2];
unsigned char mode;                             // function and display mode
unsigned char bz_onoff_flg;                 // Buzzer On/Off control flag

/* ----< RAM assign (External data) >----------------------------------- */
extern unsigned long base_timer;    // base timer for measuring
extern int ir_state;                        // InfraRed sensor detect condition
// capture data and base timer data(@ captured)
extern unsigned int  capture_b_data;// InfraRed
extern unsigned int  capture_b_d;   // InfraRed
extern unsigned long btimr_b;               // InfraRed
```

```
extern unsigned char id_save;              // number of Lapdata
extern unsigned int tim_usr0,
                                    tim_usr1,            // use for wait control,
                                    tim_usr2,            // timer1 is drived by RTM (2mS tick)
                                    tim_usr3;            // 30Sec for data display hold
///// debug ///////
extern int flag_printf;                            // for debug
/*****************************************************************************************
******************** Event driven task and Cyclic task ***********************************
*****************************************************************************************/

/*****************************************************************************************
       Event Driven Tasks
*****************************************************************************************/
/* ---------------------------------------------------------------------------------------
       ED0 Task  --- Lap time calculation by InfraRed sensor with TimerW interrupt
 ---------------------------------------------------------------------------------------*/
void ed0_main(void)
{
       unsigned long temp;

       if (tim_usr1 == 0){
              temp = (btimr_b * BCHG) + (capture_b_d / B500uS_BASE);
              lap_ir = temp  - time_ir_temp;
              lap_start = btimr_b;
              time_ir_temp = temp;
              tim_usr1 = T_5S_16B;
              tim_usr2 = T_200MS;                          // for Buzzer ON
              tim_usr3 = T_10S_16B;               // Hold Lap data display
              mode = M_HOLDL;                                    // mode is hold
              if(lap_1st_one == 0){               // this is first time from power on
                     lap_1st_one = 1;                    // next data save to EEPROM
                     mode = M_NORML;                           // mode is normal
              } else {                                         // can save to EEPROM

                     ready_ir_data();
              }
       } else {
              if (ir_state == 0){
                     tim_usr1 = T_5S_16B;
              }
       }
}

/* ---------------------------------------------------------------------------------------
       ED1 Task
 ---------------------------------------------------------------------------------------*/
void ed1_main(void)
{
       ;
}

/* ---------------------------------------------------------------------------------------
       ED2 Task
 ---------------------------------------------------------------------------------------*/
void ed2_main(void)
{
```

```
	;
}

/* ----------------------------------------------------------------------------------------
	ED3 Task  --- Not use
 ----------------------------------------------------------------------------------------*/
void ed3_main(void) { ; }

/*****************************************************************************************
	Cyclic Tasks  !!! WARNING !!! CY0 & CY1 is opposite order for only description point of view
*****************************************************************************************/
/* ----------------------------------------------------------------------------------------
	CY1 Task  --- check operation mode & read switch status
 ----------------------------------------------------------------------------------------*/
void cy1_init( void )
{
	sw_condition[MODESW].BIT.cunter = 3;
	sw_condition[MODESW].BIT.dirctn = DR_ON;
	sw_condition[MODESW].BIT.status = SWOFF;
	sw_condition[MODESW].BIT.modify = SWOFF;
	sw_condition[ENTRSW].BYTE = sw_condition[MODESW].BYTE;
}

void cy1_main( void )
{
	if (tim_usr2 == 0){
		BUZZER = OFF_DR_HS;			// Buzzer OFF
	} else {
		if (bz_onoff_flg == ON){
			BUZZER = ON_DR_HS;		// Buzzer ON
		} else {
			BUZZER = OFF_DR_HS;		// Buzzer OFF
		}
	}
	swich_status_check(MODESW);
	swich_status_check(ENTRSW);
}

void swich_status_check(unsigned char swno)
{
	unsigned char i;

	i = SWOFF;
	switch (swno){
		case MODESW:{
			if (SW1_MODE == SWON){
				i = SWON;
			}
			break;
		}
		case ENTRSW:{
			if (SW2_ENTR == SWON){
				i = SWON;
			}
			break;
		}
		default:{
			;//  Error
```

```
		}
	}
	if (i == SWON){
		if (sw_condition[swno].BIT.status == SWOFF){
			if (sw_condition[swno].BIT.dirctn == DR_ON){
				sw_condition[swno].BIT.cunter -= 1;
				if (sw_condition[swno].BIT.cunter == 0){
					sw_condition[swno].BIT.cunter = 3;
					sw_condition[swno].BIT.dirctn = DR_OFF;
					sw_condition[swno].BIT.status = SWON;
					sw_condition[swno].BIT.modify = SWON;
				}
			}
		} else {
			if (sw_condition[swno].BIT.dirctn == DR_OFF){
				sw_condition[swno].BIT.cunter = 3;
			}
		}
	} else {
		if (sw_condition[swno].BIT.status == SWON){
			if (sw_condition[swno].BIT.dirctn == DR_OFF){
				sw_condition[swno].BIT.cunter -= 1;
				if (sw_condition[swno].BIT.cunter == 0){
					sw_condition[swno].BIT.cunter = 3;
					sw_condition[swno].BIT.dirctn = DR_ON;
					sw_condition[swno].BIT.status = SWOFF;
				}
			}
		} else {
			if (sw_condition[swno].BIT.dirctn == DR_ON){
				sw_condition[swno].BIT.cunter = 3;
			}
		}
	}
}

/*-------------------------------------------------------------------------------------------
	CY0 Task  --- control for Front switches and check operation mode
-------------------------------------------------------------------------------------------*/
void cy0_init( void )
{
	lap_ptr_buf_top = 0;
	lap_ptr_rd_num = 0;
	eepdt_flg = 0;
	lap_ptr_reach_end = 0;
}

void cy0_main( void )
{
	switch (mode){
		case M_NORML:{                        // Display normal lap time count
			if (swchk_enty(M_NODSP)){// Number display then show lapdata
				lap_ptr_buf_top = 0;
				lap_ptr_rd_num = 0;
				eeprom_rd_req(IR_LAP);
			}
			swchk_mode(M_DELET);
			break;
```

```
		}
		case M_INTRO:{                      // Just after power on
			if (swchk_enty(M_NODSP)){// Number display then show lapdata
				lap_ptr_buf_top = 0;
				lap_ptr_rd_num = 0;
				eeprom_rd_req(IR_LAP);
			}
			swchk_mode(M_DELET);
			break;
		}
		case M_NODSP:{                      // Display EEPROM next save ID
			swchk_enty(M_HISTL);   // Show lapdata
			swchk_mode(M_NORML);   // Return Normal data display
			break;
		}
		case M_HOLDL:{                      // Stay static display of laped data
			if (swchk_enty(M_NODSP)){// Number display then show lapdata
				lap_ptr_buf_top = 0;
				lap_ptr_rd_num = 0;
				eeprom_rd_req(IR_LAP);
			}
			swchk_mode(M_NORML);   // Release Hold condition any key
			break;
		}
		case M_HISTL:{                      // Show history of lapdata
			if (swchk_enty(M_NODSP)){// Number display then next lapdata
				if (lap_ptr_reach_end == 0){ // not reach to end
					lap_ptr_rd_num++;
				}
				if ( lap_ptr_rd_num == BZ4LAP ){
					lap_ptr_rd_num = 0;
					lap_ptr_buf_top += BZ4LAP;
					eeprom_rd_req(IR_LAP);
				}
			}
			swchk_mode(M_NORML);   // Return Normal data display
			break;
		}
		case M_DELET:{                      // Delete EEPROM lapdata
			swchk_enty(M_Y_DLT);   // make sure delete action
			swchk_mode(M_NORML);   // not to delete!!
			break;
		}
		case M_Y_DLT:{                      // Delete EEPROM lapdata
			swchk_enty(M_NORML);   // deleted!!
			swchk_mode(M_NORML);   // deleted!!
			break;
		}
		default: {
			//  Error
			mode = M_NORML;
		}
	}
}

void eeprom_rd_req(unsigned char md)
{
	eepdt_md = md;
```

```
	eepdt_flg = REQEST;
}

// for debug
static void debug_print( void)
{
	if ((flag_printf == 1) || (flag_printf == 0)){	// Output control for Debug
		switch (mode){
			case M_NORML:{			// Display normal lap time count
				SciPutS("M_NORML, ");
				break;
			}
			case M_INTRO:{			// Just after power on
				SciPutS("M_INTRO, ");
				break;
			}
			case M_NODSP:{			// Display EEPROM next save ID
				SciPutS("M_NODSP, ");
				break;
			}
			case M_HOLDL:{			// Stay static display of laped data
				SciPutS("M_NORML");
				break;
			}
			case M_HISTL:{			// Show history of lapdata
				SciPutS("M_HISTL, ");
				break;
			}
			case M_DELET:{			// Delete EEPROM lapdata
				SciPutS("M_DELET, ");
				break;
			}
			case M_Y_DLT:{			// Delete EEPROM lapdata
				SciPutS("M_Y_DLT, ");
				break;
			}
			default: {
				SciPutS("?, ");
			}
		}
	}
}

// ----------- Switch condition check  -------------
int swchk_mode(unsigned char md)
{
	if (sw_condition[MODESW].BIT.modify == SWON){
		sw_condition[MODESW].BIT.modify = SWOFF;
		mode = md;
		tim_usr2 = T_50MS;			// for Buzzer ON
		debug_print();
		return 1;
	} else {
		return 0;
	}
}

int swchk_enty(unsigned char md)
```

```
{
	if (sw_condition[ENTRSW].BIT.modify == SWON){
		sw_condition[ENTRSW].BIT.modify = SWOFF;
		mode = md;
		tim_usr2 = T_50MS;			// for Buzzer ON
		debug_print();
		return 1;
	} else {
		return 0;
	}
}

/* ------------------------------------------------------------------------------------------
	CY2 Task  --- LED control
 ------------------------------------------------------------------------------------------*/
void cy2_init( void )
{
	;
}

void cy2_main( void )
{
	switch (mode){
		case M_NORML:{
			mode_led_flg = 0;			// Normal mode
			LED_MODE = OFF_DR_HS;		// Mode LED OFF
			tim_dsp_to_led((base_timer  - lap_start)* BCHG,0,1);
			break;
		}
		case M_INTRO:{
			mode_led_flg = 0;			// Normal mode
			LED_MODE = OFF_DR_HS;		// Mode LED OFF
			id_dsp_to_led(id_save+1);
			break;
		}
		case M_NODSP:{
			mode_led_flg = 1;			// Control mode
			LED_MODE = ON_DR_HS;		// Mode LED ON
			id_dsp_to_led(lap_data_buf[lap_ptr_rd_num].id);
			break;
		}
		case M_HOLDL:{
			mode_led_flg = 0;			// Normal mode
			LED_MODE = OFF_DR_HS;		// Mode LED OFF
			if (tim_usr3 == 0){
				mode = M_NORML;
			} else {
				tim_dsp_to_led(lap_ir,1,1);
			}
			break;
		}
		case M_HISTL:{
			mode_led_flg = 1;			// Control mode
			LED_MODE = ON_DR_HS;		// Mode LED ON
			if (eepdt_flg == READY){
				tim_dsp_to_led(lap_data_buf[lap_ptr_rd_num].laptim, 1, 1);
			}
			break;
```

```
		}
		case M_DELET:{
			if (mode_led_flg){
				mode_led_flg = 0;		// Normal mode
				LED_MODE = OFF_DR_HS;	// Mode LED OFF
			} else {
				mode_led_flg = 1;		// Control mode
				LED_MODE = ON_DR_HS;	// Mode LED ON
			}
			delete_ok_dsp_to_led();
			break;
		}
		case M_Y_DLT:{
			if (mode_led_flg){
				mode_led_flg = 0;		// Normal mode
				LED_MODE = OFF_DR_HS;	// Mode LED OFF
			} else {
				mode_led_flg = 1;		// Control mode
				LED_MODE = ON_DR_HS;	// Mode LED ON
			}
			mode_led_flg = 1;			// Control mode
			LED_MODE = ON_DR_HS;		// Mode LED ON
			allzero_dsp_to_led();
			req_lapdata_clear();
			break;
		}
		default: {
			/*  Error */
		}
	}
}

/* ------------------------------------------------------------------------------------------
	CY3 Task   --- LED Bright Control
 ------------------------------------------------------------------------------------------*/
void cy3_init( void ) {	; }
void cy3_main( void )
{
	led_bright();
}

/* ------------------------------------------------------------------------------------------
	CY4 Task   --- Not use
 ------------------------------------------------------------------------------------------*/
void cy4_init( void ) {	; }
void cy4_main( void ) {	; }
/* ------------------------------------------------------------------------------------------
	CY5 Task   --- Not use
 ------------------------------------------------------------------------------------------*/
void cy5_init( void ) {	; }
void cy5_main( void ) { ; }
/* ------------------------------------------------------------------------------------------
	CY6 Task   --- Not use
 ------------------------------------------------------------------------------------------*/
void cy6_init( void ) {	; }
void cy6_main( void ) { ; }
/* ------------------------------------------------------------------------------------------
	CY7 Task   --- Not use
```

```
 ---------------------------------------------------------------------------------------------*/
void cy7_init( void ) {     ; }
void cy7_main( void ) { ; }

/*********************************************************************************************
************** General initialization for each task and base task for monitor ******************
*********************************************************************************************/
void main( void )
{
       /********** Interrupt ***************************/
       irq_disable();
       /********** TimerV ******************************/
       reset_timerV();
       /********** RTM *********************************/
       rtm_h8_init();
       /********** EEPROM ******************************/
       init_eep_lap();
       /********** Hardware related initialize *********/
       port_init();  // Port
       ad_init();           // ADC
       /********** SCI *********************************/
       SciInit(rxBuf, sizeof( rxBuf ), IdleEEP );
       PutCRLF();    PutCRLF();
       opning_msg();
       /********** others ******************************/
       general_init();
       /********** Interrupt ***************************/
       irq_enable();
       /********** LED *********************************/
       //     reset_led();  // in tim_w.c
       /********** TimerW ******************************/
       reset_timerW();

       /* Initialize each task */
       cy0_init();
       cy1_init();
       cy2_init();
       cy3_init();
       cy4_init();
    time_ir_temp = 0;

       while(!DOOMSDAY)
       {
              // Monitor
//            cmdsrch();                  // start from H8 monitor
              lap_cmd_srch();             // start from Lap monitor
       }      // End of while
}

/*********************************************************************************************
       General subroutine
*********************************************************************************************/
void general_init( void )
{
       latest_dt = 0;
       lap_1st_one = 0;
       lap_start = base_timer = 0;
       lap_latest = lap_last1 = lap_last2 = TIM_MAX_DT;
```

```
	lap_best  = TIM_MIN_DT;
	flag_printf = 0xff;
	BUZZER = OFF_DR_HS;		// Buzzer off
	mode_led_flg = 0;		// Mode LED flag
	LED_MODE = OFF_DR_HS;	// Mode LED off
	mode = M_INTRO;
	// Data read from EEPROM
	rd_eep_bz();			// read buzzer flag from EEPROM
}

/********** Port initialize ****************/
void port_init ( void )
{
	IO.PCR1		= 0xff;		// P17 to P10  -- output mode (7 segments LED each seg. & # of
LED)
	IO.PDR1.BYTE  = 0x00;	//
	IO.PCR2		= 0x00;		//
	IO.PDR2.BYTE  = 0;		// Set all low
	IO.PCR5		= 0x1f;		// Bit 0 to 4 are output
	IO.PUCR5.BYTE = 0x10;	// Pull-up ON
	IO.PDR5.BYTE  = 0;		// Set all low
	IO.PMR5.BYTE  = 0;		// Disable wake-up function
	IO.PCR7		= 0x70;		// output P74,75&76
	IO.PDR7.BYTE  = 0x00;	// Set all low
	IO.PCR8		= 0xdb;		// Bit P85&82 = input
	IO.PDR8.BYTE  = 0x00;	// Set all low
	//	port B is only input
}

void help_msg( void )
{
	opning_msg();
}

void sw_status(void)
{
	SciPutS(" Left < - SW1  SW2  -> Right");	PutCRLF();
	SciPutS("         ENTR MODE         ");	PutCRLF();
	SciPutS("         ");
	if (SW2_ENTR == ON){
		SciPutS("OFF  ");
	} else {
		SciPutS("ON   ");
	}
	if (SW1_MODE == ON){
		SciPutS("OFF  ");
	} else {
		SciPutS("ON   ");
	}
}

void conv_id(unsigned char id, char *dt_save)
{
	unsigned char i;
	unsigned char dt;

	dt = id;							// output dxx
	i = (char)(dt  / 100) + '0';
```

```
        *dt_save++ = i;
        dt = dt  - ((dt / 100) * 100);                    // output xdx
        i = (char)(dt  / 10 ) + '0';
        *dt_save++ = i;
        dt = dt  - ((dt / 10 ) * 10 );                    // output xxd
        i = (char)dt + '0';
        *dt_save++ = i;
        *dt_save++ = '=';
}

void abort(void)
{
        ;
}

void PutCRLF()
{
        SciPutC(0x0d);
        SciPutC(0x0a);
}

void IdleNop( void )
{
        ;
}

void formater4timer_dbl(unsigned long tim, char *dt_save, char typ)
{
        unsigned char i;
        unsigned long dt;

        dt = tim & 0x07ffffff;                                  // output Tx:xx:xx.xx
        i = (char)(dt  / 72000000) + '0';
        if (typ == TIM_FULL){
                *dt_save++ = i;
        }
        dt = dt  - ((dt / 72000000) * 72000000);    // output xT:xx:xx.xx
        i = (char)(dt  / 7200000 ) + '0';
        if (typ == TIM_FULL){
                *dt_save++ = i;
                *dt_save++ = 'h';
        }
        dt = dt  - ((dt / 7200000 ) * 7200000 );    // output xx:Tx:xx.xx
        i = (char)(dt  / 1200000  ) + '0';
        *dt_save++ = i;
        dt = dt  - ((dt / 1200000  ) * 1200000  );  // output xx:xT:xx.xx
        i = (char)(dt  / 120000   ) + '0';
        *dt_save++ = i;
        if (typ != TIM_DEL){
                *dt_save++ = ':';
        }
        dt = dt  - ((dt / 120000   ) * 120000   );  // output xx:xx:Tx.xx
        i = (char)(dt  / 20000    ) + '0';
        *dt_save++ = i;
        dt = dt  - ((dt / 20000    ) * 20000   );   // output xx:xx:xT.xx
        i = (char)(dt  / 2000     ) + '0';
        *dt_save++ = i;
        if (typ != TIM_DEL){
```

```
                *dt_save++ = '.';
        }
        dt = dt  - ((dt / 2000    ) * 2000  );              // output xx:xx:xx.Tx
        i = (char)(dt  / 200     ) + '0';
        *dt_save++ = i;
        dt = dt  - ((dt / 200     ) * 200   );              // output xx:xx:xx.xT
        i = (char)(dt  / 20     ) + '0';
        *dt_save++ = i;
        *dt_save++ = 0;
}

//
// ADC control program
//
void ad_init( void ) {
        AD.ADCSR.BYTE = 0x30; // scan mode but only ch0
}

void ad_prnt( void ) {
        SciPutS("ADC(Vol.) = ");
        ad_out(AD.ADDRA);
        PutCRLF();
}

void ad_out( int value ) {
        unsigned int dt, tmp, zero;

        dt = (unsigned int)value/64;
        tmp = dt/1000 + '0';
        if (tmp == '0'){
        //      SciPutC(' ');
                zero = 0;
        } else {
                SciPutC(tmp);
                zero = 1;
        }
        dt = dt  - (( dt / 1000 ) * 1000);
        tmp = dt/100 + '0';
        if ((tmp == '0') & (zero == 0)){
        //      SciPutC(' ');
                zero = 0;
        } else {
                SciPutC(tmp);
                zero = 1;
        }
        dt = dt  - (( dt / 100 ) * 100);
        tmp = dt/10 + '0';
        if ((tmp == '0') & (zero == 0)){
        //      SciPutC(' ');
                zero = 0;
        } else {
                SciPutC(tmp);
                zero = 1;
        }
        SciPutC(dt%10 + '0');
}
```