```c
/************************************************************************************
	Monitor program for H8/300H
		monitor.c

		November 24,2001	Start M30835F system
		January  12,2002	Combined mon0.c & mon1.c
		January  20,2002	respepate monitor routine as monitor.c
		March    14,2004	Impliment to H8/300H
		July     27,2004	for 3664F
		July     30,2004
		September18,2005	Change to H8/3694F
		October 23,2005

	Copyright (C) 2001,'02,'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.
************************************************************************************/
/***** Include File *****/
#include	"iodefine.h"
#include	"const.h"
#include	"sci.h"
#include	"iiceep.h"
#include	"task.h"

/***** Define *****/
#define ROM_RAM		0
#define EEPROM 1
#define EEP_256K 0x7ff0			// not use 0x7fff
#define EEP_64K  0x1ff0			// not use 0x1fff
#define BFSZ  40
#define ERR   0xff
#defineRAM_B  0xf780		// RAM Start address
#defineRAM_E  0xff7f		// RAM End address
#define MAP_64K 0x00fff0	// H8/2694F

/***** Function Prototype *****/
	void xanalog( void );
	void xtime( void );
	void xgpssta( void );
	void xswitch_test( void );
	void x7segtst( void );
	void x7segtstxx( void );
	void test_7seg_led(void);
	void test_7seg_ledxx(void);
	void xgeneral( void );
	void xtim_lap( void );
	void lap_cmd_srch( void );

	void cmdsrch( void );
	void opning_msg( void );
	void xhelp_OS( void );
	void xhelp_map( void );
	void xhelp( void );
	void xdumpb( void );
	void xdumpw( void );
```

```
       void xchang( void );
       void xchangb( void );
       void xchangw( void );
       void xfillmem( void );
       void xenable_pf( void );
       void xdisable_pf( void );
       void xerrcmd( void );
       void getline( void );

       void error( void );

       char *eqstrf( char * , char * );

       static void chglp( unsigned char, unsigned long );

       void chkcomma( void );
       void chkend( void );
       unsigned char chkterm( void);
       unsigned long getdata( void );
       unsigned long ogetdata( unsigned long );
       unsigned char peekb( unsigned long );
       unsigned int peekw( unsigned long );
       void poke( unsigned long, unsigned char );
       void pokew( unsigned long, unsigned int );
       void skpspc( void );
//     void puts( char * );
       void puthxb( unsigned char );
       void puthxw( unsigned int );
       void puthxl( unsigned long n);
       unsigned char chksfmt( void );
       unsigned long gethex( void );
       SIZE chkwb( void );

       void SciGetS( char *, short );
       void PutCRLF( void );
       char toupper( char );
       void SciPutS( char * );
       int Sci_GetChar(void);
       void dummy(int);

       void charcpy( char  *, char  * );

       void xrdumpb( void );
       void xrdumpw( void );
       void xrchang( void );
       void xrchangb( void );
       void xrchangw( void );
       void xedumpb( void );
       void xechangb( void );

/***** Function Prototype ( Extern )  ********/
       extern void PutCRLF( void );

/******************************************************************************************/
/*                           0         10        20        30        40        50        60  */
/*                           01234567890123456789012345678901234567890123456789012345678900  */
static char *const mntrmsg1= "     ------ H8 Simple Monitor [?=H(ret)]  ------";
static char *const exitmsg = "  Do you want to return command mode? 'Y'(RET) or 'N'";
static char *const addrno1 = "  ADDR   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F";
static char *const addrno2 = "  ADDR   0    2    4    6    8    A    C    E";
static char *const chngmsg = "  ADDR   OLDDATA=(entry data)  RET=NEXT ^=BACK .=QUIT ";
static char *const errfill = "  Fill command only for RAM(0xf780-0xff7f) ";
static char *const openmsg = "   H8/3694F Laptimer-LED type, V.1.0  Oct.,2005";
static char *const opnmsg0 = "   By JH1PJL / K.Arai       kenjia@sannet.ne.jp";
static char *const hmsg0=    "   RTM_H8(for H8/3664F) C-Compiler Version (to 3694F)";
static char *const hmsg1=    "   Copyright (C) 1998,'99,'00,2001 K.Arai/JH1PJL ";
static char *const hmsg2=    "    ------------------------------------------- ";
static char *const hmsg3=    "   Hardware is AKI H8/3694F-Flat CPU (K-855)";
static char *const hmsg4=    "   http://akizukidenshi.com/";
static char *const hmsg5=    "   RENESAS HD64F3694FP Clock=20MHz";
static char *const hmsg6=    "   http://japan.renesas.com/homepage.jsp";
static char *const hmsg7=    "   by Kenji Arai, kenjia@sannet.ne.jp on Oct.,2005";
static char *const hmsg8=    "    -- Memory control  ------";
static char *const hmsg9=    "    -- information  ---------";
static char *const hmsga=    "    -- Debug  ---------------";
static char *const hmsgb=    "    -- Lap data  ------------";
static char *const hmsgMROM= "   ROM(32KB)  0x0000-0x7fff";
static char *const hmsgMRAM= "   RAM (2KB)  0xf780-0xff7f";
static char *const hmsgMIO = "   IO         0xff80-0xffff";
static char *const helpmsgH= "    -- Help  -- H or ?  YOU KNOW THIS";
static char *const helpmsgC= "   CW,CB,C CHANGE MEMORY";
static char *const helpmsgD= "   DW,DB,D DUMP MEMORY";
static char *const helpmsgE= "   EC,E    EEPROM Memory Change(EC) & Dump(E) (Byte)";
static char *const helpmsgL1="   [RET]   RETURN TO LAP MODE";
//static char *const helpmsgD1="   [RET]   DUMP MEMORY";
static char *const helpmsgF= "   F       FILL MEMORY f start_addr,end_addr,data";
static char *const helpmsgEP="   EP,DP   ENABLE Printf() w/#1,2,3,4,0=All & DISABLE";
static char *const helpmsgA= "   A,S     SHOW ANALOG DATA (A) and SWITCH status (S)";
static char *const helpmsgG= "   G       TEST for General purpose";
static char *const helpmsgI= "   I,M     SHOW OS information (I) and Memory (M)";
static char *const helpmsg7= "   7       TEST 7SEG LED";
//static char *const helpmsgL= "   L       SHOW LAP DATA";
extern char *const mntrmsg0;
/*                         01234567890123456789012345678901234567890123456789012345678901234567890  */
/*************************************************************************************************/

/***********  Data in RAM ****************************************************/
extern char line[BFSZ], *lp;
extern unsigned long datap;

extern unsigned char error_status;  // buffer for error status

char memory_flag;                   // Flag for normal = 0 and EEPROM = 1
int  flag_printf;
/****************************************************************************
*       Simple Monitor
*****************************************************************************/
/*  ----- Command Table ----- */
typedef const struct{
        char   *cmd;
        void   (*func)( void );
}TBLENTRY;

TBLENTRY cmdtbl[] = {
        {"7",x7segtst},             /* Check 7 seg LED */
        {"A",xanalog},       /* Show ADC data */
```

```
	{"CB",xrchangb},	/* Change Memory (Byte) */
	{"CW",xrchangw},	/* Change Memory (Word) */
	{"C",xrchang},		/* Change Memory */
	{"DB",xrdumpb},			/* Dump Memory (Byte) */
	{"DP",xdisable_pf},	/* Disable printf() */
	{"DW",xrdumpw},			/* Dump Memory (Word) */
	{"D",xrdumpb},		/* Dump Memory (Word) */
	{"EC",xechangb},	/* EEPROM Memory Change (Byte) */
	{"EP",xenable_pf},	/* Enable printf() */
	{"E",xedumpb},		/* EEPROM Memory Dump (Byte) */
	{"F",xfillmem},			/* Fill Memory */
	{"G",xgeneral},			/* Test for General*/
	{"H",xhelp},		/* Help */
	{"I",xhelp_OS},			/* Help for HOS */
//	{"L",xtim_lap},			/* Lap data */
	{"M",xhelp_map},	/* Help for MAP */
	{"S",xswitch_test},	/* Check switch status */
	{"X",x7segtstxx},	/* Check 7 seg LED */
	{"?",xhelp},		/* Help */
	{"\0",xerrcmd},			/* Command Error */
};

/*  -----Command analysis and excute ----- */
void cmdsrch( void )
{
	char *c;
	TBLENTRY *p;

	SciPutS( mntrmsg1 );
	for(;;){
		PutCRLF();
		SciPutS("H8>");
		getline();
		if(line[0] == '\0'){
			SciPutS( mntrmsg0 );
			return;
		}
/*
		if(line[0] == '\0'){
			line[0] = 'D';
			line[1] = '\0';
		}
*/
		for(p = cmdtbl; *p -> cmd; p++){
			if((c = eqstrf(p -> cmd,lp)) != 0){
				break;
			}
		}
		lp = c;
//		PutCRLF();
		(*p -> func)();
	}
}

/*  ------Opening ------- */
void opning_msg( void )
{
	SciPutS( openmsg ); PutCRLF();
	SciPutS( opnmsg0 ); PutCRLF();
}

/*  ------Help ------- */
void xhelp_OS( void )
{
	opning_msg();
	SciPutS( hmsg0 );		PutCRLF();
	SciPutS( hmsg1 );		PutCRLF();
	SciPutS( hmsg2 );		PutCRLF();
	SciPutS( hmsg3 );		PutCRLF();
	SciPutS( hmsg4 );		PutCRLF();
	SciPutS( hmsg5 );		PutCRLF();
	SciPutS( hmsg6 );		PutCRLF();
	SciPutS( hmsg7 );		PutCRLF();

}

void xhelp( void )
{
	opning_msg();
	SciPutS( helpmsgH );  PutCRLF();
	// lap
	SciPutS( hmsgb );		PutCRLF();
//	SciPutS( helpmsgL );  PutCRLF();
	SciPutS( helpmsgL1);  PutCRLF();
	// memory control
	SciPutS( hmsg8 );		PutCRLF();
	SciPutS( helpmsgC );  PutCRLF();
	SciPutS( helpmsgD );  PutCRLF();
	SciPutS( helpmsgF );  PutCRLF();
	SciPutS( helpmsgE );  PutCRLF();
	// information
	SciPutS( hmsg9 );		PutCRLF();
	SciPutS( helpmsgI );  PutCRLF();
	// debug
	SciPutS( hmsga );		PutCRLF();
	SciPutS( helpmsgA );  PutCRLF();
	SciPutS( helpmsg7 );  PutCRLF();
	SciPutS( helpmsgEP);  PutCRLF();
	SciPutS( helpmsgG );  PutCRLF();
}

void xhelp_map( void )
{
	opning_msg();
	SciPutS( hmsgMROM );		PutCRLF();
	SciPutS( hmsgMRAM );		PutCRLF();
	SciPutS( hmsgMIO  );		PutCRLF();
}

//  -------Show analog data -------- //
void xanalog( void )
{
	ad_prnt();
}

//  -------Show Lap data -------- //
```

```
void xtim_lap( void )
{
        lap_cmd_srch();
}

//  -------Show analog data -------- //
void xswitch_test( void )
{
        SciPutS( "SW1(MODE) = " );
        if (SW1_MODE == SWON) {
                SciPutS( "ON " );
        } else {
                SciPutS( "OFF" );
        }
        SciPutS( "   SW2(ENTR) = " );
        if (SW2_ENTR == SWON) {
                SciPutS( "ON " );
        } else {
                SciPutS( "OFF" );
        }
        PutCRLF();
}

//  -------Test 7 segments LED -------- //
void x7segtst( void )
{
        test_7seg_led();
}
void x7segtstxx( void )
{
        test_7seg_ledxx();
}

//  -------Test for general purpuse -------- //
void test_eeprom( void)
{
        extern unsigned char   Device_id;
        unsigned short  Address;
        unsigned short  Address_start;
        unsigned short  Address_end;

        unsigned short  Rezult_code;
        unsigned char   Err_code;
        unsigned char   Data_w;
        unsigned char   Data_r;

        Address_start = 0x000;
        Address_end = 0x200;
        for (Address = Address_start; Address < Address_end; Address++){
                Data_w = (unsigned char) (Address & 0x00FF);

                Err_code = Master_byte_write (Device_id, Address, Data_w);
                if (Err_code != 0){
                        SciPutS("error in EEPROM acess");
                     return;
         }

                Rezult_code = Master_read_byte_random (Device_id, Address);
```

```c
		Err_code = Rezult_code >> 8;
		Data_r = (unsigned char) (Rezult_code & 0x00FF);
		if (Err_code != 0){
			SciPutS("error in EEPROM acess");
			return;
		}

		if (Data_w != Data_r){
			SciPutS("data not equal (Write and Read action");
	}
	}
}

void xgeneral( void )
{
//	SciPutS( "Nothing support at this time! " );
//	test_eeprom();
	test_led_bzr();
}


/*  ------Select ROM&RAM or EEPROM ------- */
void xrchangb (void)
{
	memory_flag = ROM_RAM;
	xchangb();
}

void xrchangw (void)
{
	memory_flag = ROM_RAM;
	xchangw();
}

void xrchang (void)
{
	memory_flag = ROM_RAM;
	xchang();
}

void xrdumpb (void)
{
	memory_flag = ROM_RAM;
	xdumpb();
}

void xrdumpw (void)
{
	memory_flag = ROM_RAM;
	xdumpw();
}

void xedumpb (void)
{
	memory_flag = EEPROM;
	xdumpb();
}
```

```
void xechangb (void)
{
        memory_flag = EEPROM;
        xchangb();
}

/*  -------Dump byte memory -------- */
void xdumpb( void )
{
        unsigned long p,q;
        unsigned long from,to;
        unsigned char ct;
        unsigned char c;

        skpspc();
        from = ogetdata( datap );
        if (memory_flag == EEPROM){
                from &= EEP_256K;
        } else {
                from &= MAP_64K;
        }
        if(chkterm()){
                to = from + 128;
        }
        else{
                chkcomma();
                to = ogetdata( from + 16 * 8 -1 );
        }
        chkend();
        SciPutS( addrno1 );
        for( p = from; from < to && p < to; p += 16 ){
                PutCRLF();
                SciPutC( ' ' );
                puthxl( p );
                SciPutS( "  " );
                q = p; ct= 16;
                do{
                        puthxb(peekb(q++));
                        SciPutC( ' ' );
                }while( --ct != 0);
                SciPutC( ' ' );
                SciPutC( ' ' );
                q = p; ct= 16;
                do{
                        c = peekb(q++);
                        if( c <= ' '  || c >= 0x7f  || c == '>'){
                                c = '.';
                        }
                        SciPutC((unsigned char)c);
                }while( --ct != 0);
        }
        datap = p;
}

/*  -------Dump word memory -------- */
void xdumpw( void )
{
        unsigned long p,q;
        unsigned long from,to;
        unsigned char ct;

        memory_flag = ROM_RAM;
        skpspc();
        from = ogetdata( datap );
        from &= MAP_64K;
        if(chkterm()){
                to = from + 128;
        }
        else{
                chkcomma();
                to = ogetdata( from + 16 * 8 -1 );
        }
        chkend();
        SciPutS( addrno2 );
        for( p = from; from < to && p < to; p += 16 ){
                PutCRLF();
                SciPutC( ' ' );
                puthxl( p );
                SciPutS( "  " );
                q = p; ct= 8;
                do{
                        puthxw(peekw(q));
                        SciPutC( ' ' );
                        q += 2;
                }while( --ct != 0);
        }
        datap = p;
}

/*  -------Change word or byte memory -------- */
void xchang( void )
{
        unsigned long p;
        unsigned char n;

        skpspc();
        p = ogetdata(datap);
        skpspc();
        n = (unsigned char)chkwb() + 1;
        if(n == 2) p &= 0x00fffffe;
        SciPutS( chngmsg );
        chglp( n, p );
}

/*  -------Change byte memory -------- */
void xchangb( void )
{
        unsigned long p;

        skpspc();
        p = ogetdata(datap);
        chkend();
        SciPutS( chngmsg );
        chglp( 1, p );
}
```

```
/*  -------Change word memory -------- */
void xchangw( void )
{
        unsigned long p;

        skpspc();
        p = ogetdata(datap) & 0x00fffffe;
        chkend();
        SciPutS( chngmsg );
        chglp( 2, p );
}

/*  -------Change memory for ever -------- */
static void chglp( unsigned char wbflg, unsigned long addr )
{
        register unsigned int n;

        for(;;){
                PutCRLF();
                SciPutC(' ');
                puthxl(addr);
                SciPutC(' ');
                SciPutC(' ');
                if( wbflg == 2 ){
                        puthxw( peekw(addr) );
                }
                else{
                        puthxb( peekb(addr) );
                }
                SciPutC('=');
                getline();
                switch( line[0] ){
                        case '.':
                                return;
                        case '^':
                                addr  -= wbflg;
                                break;
                        case '\0':
                                addr += wbflg;
                                break;
                        default:
                                n = (unsigned int)getdata();
                                chkend();
                                if( wbflg == 2 ){
                                        pokew( addr,n );
                                }
                                else{
                                        poke( addr,n );
                                }
                                addr += wbflg;
                }
        }
}

//  -------Fill Memory ---------- //
// format f,xxxx,yyyy,dd
//
void xfillmem(void)
```

```
{
        unsigned long ps, pe, p;
        unsigned char n;

        memory_flag = ROM_RAM;
        skpspc();
        ps = ogetdata(datap); // xxxx
        skpspc();
        chkcomma();
        pe = ogetdata(datap); // yyyy
        skpspc();
        chkcomma();
        n = ogetdata(datap);  // dd
        if ((ps < RAM_B)  || (pe > RAM_E)  || (ps > pe)) {
                SciPutS(errfill);
                error();
                return;
        }
        p = ps;
        while (p <= pe){
                poke(p++,n);
        }
}

/*  -------Enable printf() -------- */
void xenable_pf( void )
{
        unsigned long p;

        skpspc();
        p = ogetdata(datap);
        flag_printf = (int) p;
        SciPutS( "Enable printf() for Debug" );
}

/*  -------Disable printf() -------- */
void xdisable_pf( void )
{
        flag_printf = 0xff;
        SciPutS( "Disable printf() for Debug" );
}

/*  -------Goto User program -------- */
void xgo( void )
{
        void   (*usrprg)();
        unsigned long p;

        skpspc();
        p = ogetdata(datap);
        chkend();
        usrprg = (void *)p;
        (*usrprg)();
}

/*  -------command error -------- */
void xerrcmd( void )
{
```

```
//	PutCRLF();
	SciPutC('?');
}

/* ************************************************
	data convert
************************************************ */
/*  -----4bit hex to ASCII ----- */
char ascii( unsigned char h )
{
	register unsigned char c;

	c = h;
	if( c <10 ){
		c += '0';
	}
	else{
		c += 'A' - 10;
	}
	return( (char)c );
}

/*  -----character strings out ----- */
/*
void puts( char *s )
{
	char c;

	while((c = *s++) != 0 ){
		SciPutC( c );
	}
}
*/

/*  -----error -----	*/
void error( void )
{
	PutCRLF();
	SciPutC( '?' );
	SciPutS("ERROR ! then Restart");	PutCRLF();
	main();
}

/*  -----strings compere ----- */
char *eqstrf( char *s , char *t )
{
	do{
		if( *t++ != *s++ )	return( 0 );
	}while( *s != 0 );
	return( t );
}

/*  -----colon end check ----- */
FLAG  chkcolon ( char *s )
{
	do{
		if( *s++ == ':' ) return( true );
	}while( *s >= '0' );
```

```c
        return( false );
}

/*  -----ASCII to hex -----  */
unsigned char hex( char c )
{
        if( c<= '/' ) return( ERROR );
        if( ( c  -= '0' ) <= 9  || 10 <= ( c  -= 'A'  - '0'  - 10 ) && c <= 15 ){
                return( (unsigned char)c );
        }
        return( ERROR );
}

/*  -----byte data out to console -------- */
void puthxb( unsigned char h)
{
        SciPutC( ascii( (char)(h >>4)) );
        SciPutC( ascii( h & 0x0f ) );
}

/*  -----word data out to console -------- */
void puthxw( unsigned int n)
{
        puthxb((unsigned char)(n >> 8));
        puthxb((unsigned char)(n & 0x000000ff));
}

/*  -----long data out to console -------- */
void puthxl( unsigned long n)
{
        puthxw((unsigned int)(n >> 8));
        puthxb((unsigned char)(n & 0x000000ff));
}

/*  -----read hex from input line buffer ----- */
unsigned long gethex( void )
{
        unsigned char c;
        unsigned long v;

        for(v = 0 ; ( c = hex( *lp ) ) != ERROR ; lp++ ){
                v = ( v << 4 ) + (unsigned long)c;
        }
        return( v );
}

/*  -----check S format header ----- */
unsigned char chksfmt( void )
{
        if( *lp == 'S'){
                *lp ++;
                return( hex( *lp ++) );
        }
        else{
                return 0;
        }
}
```

```
/*  -----number then read ----- */
unsigned long getdata( void )
{
        if( hex( *lp ) == ERROR ){
                error();
        }
        return( gethex() );
}

/*  -----data read ----- */
unsigned long ogetdata( unsigned long p )
{
        switch( *lp ){
        case ',':
        case '\0':
                return( p );
        default:
                return( getdata() );
        }
}

/*  -----skip space ----- */
void skpspc( void )
{
        while( *lp == ' ' ){
                lp++;
        }
}

/*  -----check comma ----- */
void chkcomma( void )
{
        if( *lp++ != ',' ){
                error();
        }
}

/*  -----check comma or line end ----- */
void chkecomma( void )
{
        if( *lp++ != '\0' && *lp++ != ',' ){
                error();
        }
}

/*  -----check word or byte  ----- */
SIZE chkwb( void )
{
        char c;

        switch( *lp++ ){
        case '\0':
                return( words );
        case ':':
                c = *lp;
                if( c == 'W'  || c == 'w' ){
                        return( words );
                }
                else if( c == 'B'  || c == 'b' ){
                        return( bytes );
                }
                break;
        default:
                error();
        }
        return( words );
}

/*  -----check end(if not line end then error) ------ */
void chkend( void)
{
        if( *lp != '\0') error();
}

/*  -----check terminate(if line end then true) ------ */
unsigned char chkterm( void)
{
        if( *lp == '\0'){
                return(1);
        }
        else{
                return(0);
        }
}

/*  -----peek memory(byte) ------ */
unsigned char peekb( unsigned long addr )
{
        unsigned char i;

        if (memory_flag == EEPROM){
                i = Read_byte_EEPROM((unsigned short)addr);
                if (error_status == ERROR){
                        SciPutC( '?' );
                }
                return(i);
        } else {
                return(*(unsigned char *)addr);
        }
}

/*  -----peek memory(word) ------ */
unsigned int peekw( unsigned long addr )
{
        return(*(unsigned int *)addr);
}

/*  -----peek memory(byte) ------ */
void poke( unsigned long addr, unsigned char data )
{
        unsigned char i;

        if (memory_flag == EEPROM){
                i = Write_byte_EEPROM((unsigned short)addr, data );
                if (error_status == ERROR){
                        SciPutC( '?' );
                }
        } else {
                *(unsigned char *)addr = data;
        }
}

/*  -----peek memory(word) ------ */
void pokew( unsigned long addr, unsigned int data )
{
        *(unsigned int *)addr = data;
}

/*  -----save to line buffer ------ */
void getline( void )
{
        lp = line;
        SciGetS(line,1);
}
```