```
/*****************************************************************************
       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

       Copyright (C) 2001,'02,'04  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 "3664f.h"
#include "const.h"
#include "sci.h"
#include "iiceep.h"
#include <setjmp.h>

/***** Define *****/
#define ROM_RAM      0
#define EEPROM 1
#define EEP_256K 0x7ff0             // not use 0x7fff
#define EEP_64K  0x1ff0             // not use 0x1fff

/***** Function Prototype *****/
       void cmdsrch( void );
       void xhelp( void );

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

       void xchangw( void );
       void xdumpb( void );
       void xdumpw( void );
       void xchang( void );
       void xchangb( void );
       void xchangw( void );
       void xgo( void );
       void xerrcmd( void );
       void getline( 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 );
        SIZE chkwb( void );
        unsigned long gethex( void );

/***** Function Prototype ( Extern )  ********/
extern void PutCRLF( void );
//extern unsigned char Read_byte_EEPROM( unsigned short );
//extern unsigned char Write_byte_EEPROM( unsigned short , unsigned char );

/*****************************************************************************************/
/*                          ############################################################ */
/*                          0        10        20        30        40        50        60 */
/*                          01234567890123456789012345678901234567890123456789012345678901 */
static char *const mntrmsg = " --------- 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 ";
/*                          01234567890123456789012345678901234567890123456789012345678901 */
/*****************************************************************************************/

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

extern unsigned char error_status;  // buffer for error status

char memory_flag;                   // Flag for normal = 0 and EEPROM = 1

/*************************************************************************
*       Simple Monitor
*************************************************************************/
/*  -----Command Table -----  */
typedef const struct{
        char   *cmd;
        void   (*func)( void );
}TBLENTRY;

TBLENTRY cmdtbl[] = {
        {"CB",xrchangb},/* Memory Change (Byte) */
        {"CW",xrchangw},/* Memory Change (Word) */
        {"C",xrchang},/* Memory Change */
        {"DB",xrdumpb},       /* Memory Dump (Byte) */
```

```
        {"DW",xrdumpw},       /* Memory Dump (Word) */
        {"D",xrdumpb},/* Memory Dump (Word) */
        {"EC",xechangb},/* EEPROM Memory Change (Byte) */
        {"E",xedumpb},/* EEPROM Memory Dump (Byte) */
        {"G",xgo},            /* Go to User program */
        {"H",xhelp},  /* Help */
        {"\0",xerrcmd},       /* Command Error */
};

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

        PutCRLF();
        puts( mntrmsg );
        setjmp( jb_error );
        for(;;){
                PutCRLF();
                SciPutC('>');
                getline();
                if(line[0] == '\0'){
                        puts( exitmsg );
                        getline();
                        if((line[0] == '\0') || (line[0] == 'Y')){
                                PutCRLF();
                                opning_msg();
                                return;
                        }
                        else{
                                continue;
                        }
                }
                for(p = cmdtbl; *p -> cmd; p++){
                        if((c = eqstrf(p -> cmd,lp)) != 0){
                                break;
                        }
                }
                lp = c;
                (*p -> func)();
        }
}

/* ------Help ------- */
void xhelp( void )
{
        SciPutS(" CW,CB,C  MEMORY CHANGE in RAM");              PutCRLF();
        SciPutS(" EC       MEMORY CHANGE (byte) in EEPROM");PutCRLF();
        SciPutS(" DW,DB,D  MEMORY DUMP in ROM & RAM");          PutCRLF();
        SciPutS(" E        MEMORY DUMP (byte) in EEPROM");      PutCRLF();
        SciPutS(" G        GOTO USER SUBROUTINE");              PutCRLF();
        SciPutS(" [RET]    RETURN TO CALLED ROUTINE");          PutCRLF();
        SciPutS(" H        YOU KNOW THIS");                             PutCRLF();
}

/* ------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){
		from &= EEP_256K;
	} else {
		from &= 0x00ffffF0;
	}
	if(chkterm()){
```

```
		to = from + 128;
	}
	else{
		chkcomma();
		to = ogetdata( from + 16 * 8  -1 );
	}
	chkend();
	puts( addrno1 );
	for( p = from; from < to && p < to; p += 16 ){
		PutCRLF();
		SciPutC( ' ' );
		puthxl( p );
		puts( "  " );
		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;

	skpspc();
	from = ogetdata( datap );
	from &= 0x00fffff0;
	if(chkterm()){
		to = from + 128;
	}
	else{
		chkcomma();
		to = ogetdata( from + 16 * 8  -1 );
	}
	chkend();
	puts( addrno2 );
	for( p = from; from < to && p < to; p += 16 ){
		PutCRLF();
		SciPutC( ' ' );
		puthxl( p );
		puts( "  " );
		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;
        puts( chngmsg );
        chglp( n, p );
}

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

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

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

        skpspc();
        p = ogetdata(datap) & 0x00fffffe;
        chkend();
        puts( 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;
                }
        }
}

/* -------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( '?' );
        longjmp( jb_error,2 );

        /* Not reach below */
        SciPutS("ERROR ! then Restart");    PutCRLF();
        longjmp( jb_error,2 );
//      restart();

}

/*  -----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' );
        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){
		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){
		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);
}
```