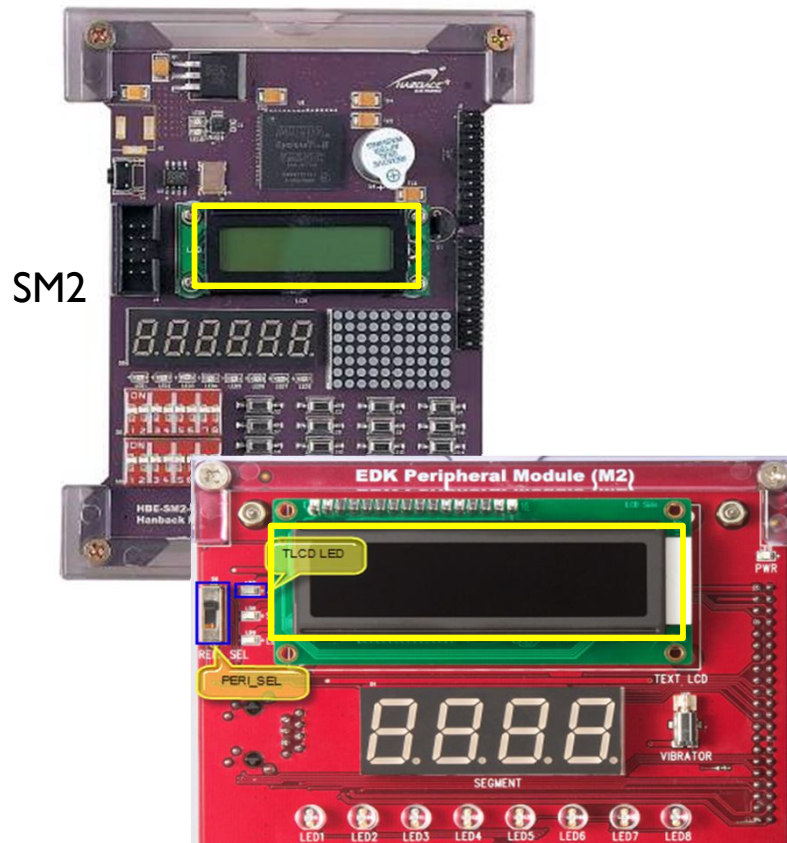


# Device driver example

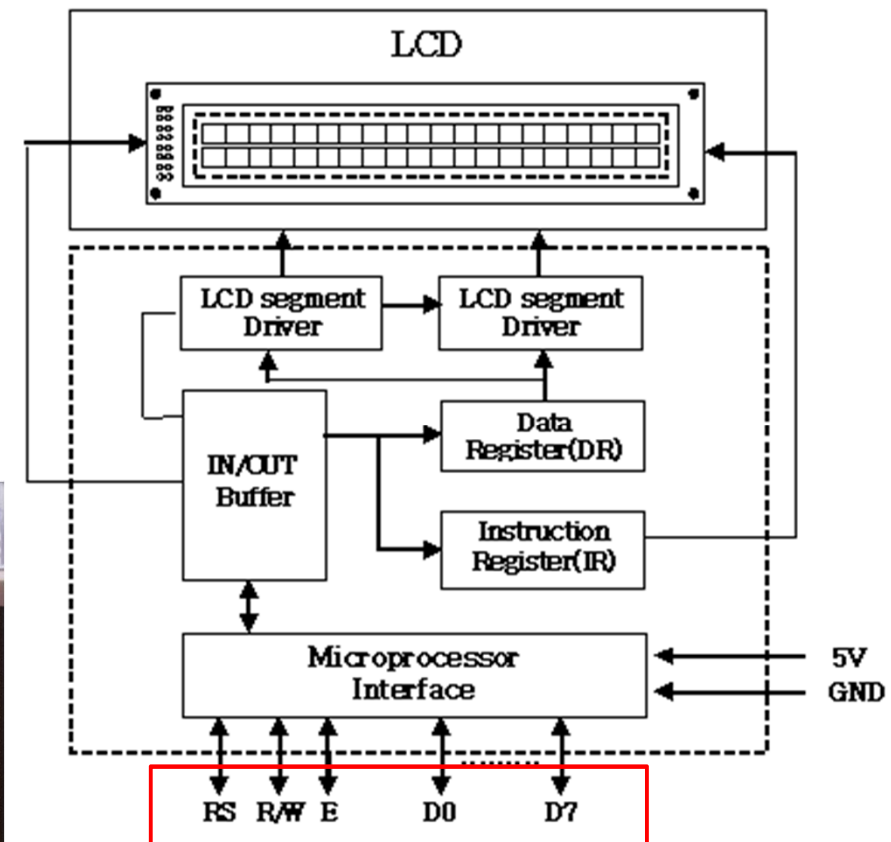
## - Text LCD -

# H/W user manual for Text LCD

## ▶ 16 \* 2의 Text LCD



EDK9



# H/W user manual for Text LCD

---

## ▶ LCD module pin

- ▶ RS(Register Selection)
  - ▶ RS핀으로 Text LCD module에 Data를 읽고 쓰기 할 레지스터를 선택.
    - 1: data register (Text LCD module에 글자를 표시하기 위해 데이터 값이 들어감)
    - 0: instruction register (Text LCD module의 환경설정)
- ▶ RW(Read/Write)
  - ▶ Text LCD 모듈에 데이터를 읽고 쓰기를 제어
  - ▶ 1: read, 0: Write
- ▶ E(Enable)
  - ▶ 1일 경우 Text LCD module에 instruction을 전달
- ▶ DB[7:0]: LCD 모듈의 레지스터에 읽고 쓰기 할 데이터 버스

# H/W user manual for Text LCD

## ▶ Text LCD Control Register

- ▶ 8 data bits
- ▶ 2 control bits(RS, RW)
- ▶ 1 enable clock bit

Physical Address  
0x1480\_9000

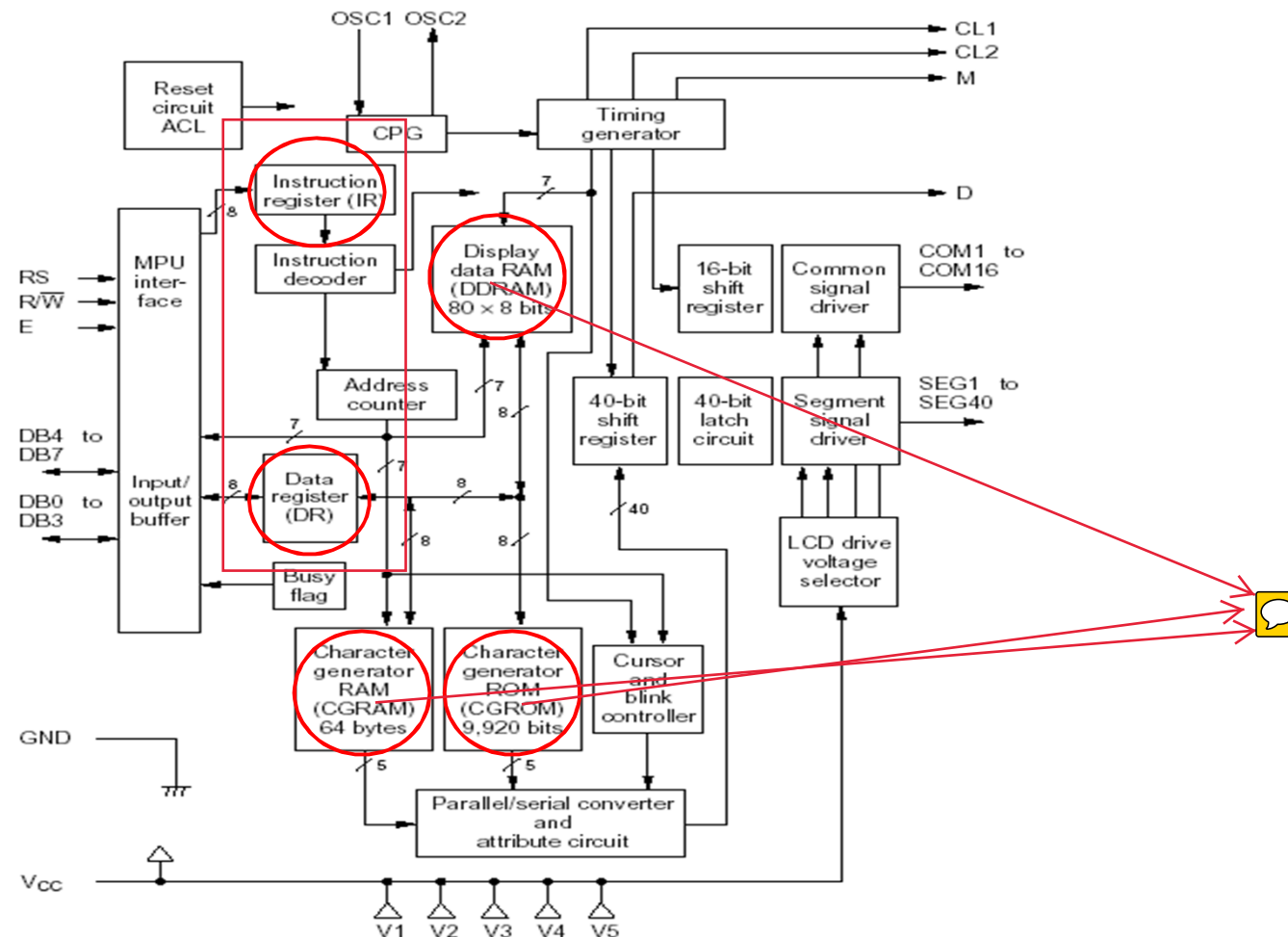
Text LCD Control  
Register

Peripheral Registers

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					RS	RW	E	D7	D6	D5	D4	D3	D2	D1	D0
Reset	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0
	Bits		Name		Description											
	D0 : D7		D0 : 7		LCD Module Data Bus [Write only]											
	D8		E		LCD Module Enable 신호 [Active High]											
	D9		RW		LCD Module Data R/W Control [Read High]											
	D10		RS		LCD Module Data Instruction Register Set											

# H/W user manual for Text LCD

- ▶ More detail block diagram of Text LCD controller



# H/W user manual for Text LCD

---

## ▶ 내부 메모리

- ▶ D.D.RAM(Display Data RAM)
  - ▶ 8비트 문자코드의 디스플레이 데이터를 저장하는 메모리
  - ▶ 최대용량은 80x8비트로 80문자를 저장 가능
  - ▶ 행별로 40 문자의 데이터를 저장
- ▶ C.G.ROM(Character Generator ROM)
  - ▶ ASCII 문자의 글씨체 정보를 저장하고 있는 ROM
- ▶ C.G.RAM(Character Generator RAM)
  - ▶ 사용자가 정의한 글씨체를 다운로드하는 RAM
  - ▶ ASCII 문자 이외의 사용자 정의 문자를 출력할 때에 사용

# H/W user manual for Text LCD

## ▶ Text LCD module control command

기 능	제어신호		제어 명령								실행시간
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	
Clear Display	0	0	0	0	0	0	0	0	0	1	1.64ms
Return Home	0	0	0	0	0	0	0	0	1	0	1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	40us
Display On/Off control	0	0	0	0	0	0	1	D	C	B	40us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	0	0	40us
Function Set	0	0	0	0	1	D/L	N	F	0	0	40us
Set CG RAM Address	0	0	0	1	CG RAM Address						40us
Set DD RAM Address	0	0	1	DD RAM Address							40us
Read Busy Flag and Address	0	1	BF	Address Counter							0us
Data Write to CG RAM or DD RAM	1	0	Write Address								40us
Data Read to CG RAM or DD RAM	1	1	Read Address								40us

# H/W user manual for Text LCD

## ▶ Clear display

- ▶ LCD display를 clear하고 cursor를 첫 줄의 첫 칸에 위치시킴.

기능	제어 신호		제어명령							
Clear Display	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	0	0	1

## ▶ Return home

- ▶ LCD display의 표시내용을 그대로 두고 cursor만 home으로 위치

기능	제어 신호		제어명령							
Return Home	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	0	1	0



# H/W user manual for Text LCD

## ▶ Entry Mode Set

- ▶ 데이터를 read/write할 경우에 cursor의 위치를 증가시킬 것인가(I/D=1) 감소시킬 것인가(I/D=0)를 결정하며, 또 이때 화면을 이동할 것인지(S=1) 아닌지(S=0)를 결정.

기능	제어 신호		제어명령							
Entry Mode Set	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	I	I/D	S



## ▶ Display On/Off Control


- ▶ 화면 표시를 On/Off 하거나(D), cursor를 On/Off하거나(C), cursor를 깜빡이게 할 것인지(B)의 여부 결정

기능	제어 신호		제어명령							
Display ON/OFF Control	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	I	D	C	B

# H/W user manual for Text LCD

## ▶ Cursor or Display Shift

- ▶ Display (S/C=1) 또는 cursor(S/C=0)를 오른쪽(R/L=1) 또는 왼쪽(R/L=0)으로 이동.

기능	제어 신호		제어명령							
Cursor or Display Shift	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	I	 C	R/L	0	0

## ▶ Function Set

- ▶ 인터페이스 데이터의 길이를 8bit(DL=1) 또는 4bit(DL=0)로 지정
  - ▶ 4비트로 인터페이스할 경우 DB4~DB7을 사용, 상위 4비트를 먼저 전송하고 하위 4비트를 전송해야 함.
- ▶ 화면 표시 행수를 2행(N=1) 또는 1행(N=0)으로 지정
- ▶ 문자의 폰트를 5\*10도트(F=1) 또는 5\*7도트(F=0)으로 지정

기능	제어 신호		제어명령							
Function Set	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	I	DL	N	F	0	0

# H/W user manual for Text LCD

## ▶ Set DD RAM Address

- ▶ Display Data RAM의 address를 지정. 이후에 송수신하는 데이터는 DD RAM의 데이터이다.
- ▶ DD RAM은 표시될 각 문자의 ASCII 코드 데이터가 저장되어 있는 메모리이며 모두 80개의 번지가 있는데, 각 행과 열의 위치에 고유한 주소 값이 부여되어 있다.

DD RAM address

	1	2	3	4	...	13	14	15	16
Line1	00	01	02	03		0C	0D	0E	0F
Line2	40	41	42	43		4C	4D	4E	4F

# H/W user manual for Text LCD

## ▶ Set CG RAM Address

- ▶ Character Generator RAM의 address를 지정. 이후에 송수신 하는 데이터는 CG RAM의 데이터이다.
- ▶ LCD 모듈에서 화면에 표시할 수 있는 문자의 종류는 대부분 ASCII 문자이다.

〈표 2-29〉 ASCII 도형문자 종류 및 코드 값

구분	00H	10H	20H	30H	40H	50H	60H	70H	80H	90H
0	사용자 정의 영역	미사용 영역		0	@	P	'	p	미사용 영역	
1			!	1	A	Q	a	q		
2			"	2	B	R	b	r		
3			#	3	C	S	c	s		
4			\$	4	D	T	d	t		
5			%	5	E	U	e	u		
6			&	6	F	V	f	v		
7			'	7	G	W	g	w		
8			(	8	H	X	h	x		
9			)	9	I	Y	i	y		
A			*	:	J	Z	j	z		
B			+	;	K	[	k	{		
C			,	<	L	₩	l			
D			-	=	M	}	m	]		
E			.	>	N	^	n	→		
F			/	?	O	_	o	←		

# Text LCD device driver

```
//filename : textlcd.c
```

```
#include <linux/init.h>
#include <linux/module.h>
#include <asm/hardware.h>
#include <asm/uaccess.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <asm-generic/ioctl.h>
#include <linux/ioport.h>
#include <asm/io.h>
#include <linux/delay.h>
#include "textlcd.h"
```

```
void setcommand(unsigned short command)
```

```
{
    command &= 0x00FF;

    *textlcd = command | 0x0000;
    *textlcd = command | 0x0100;
    *textlcd = command | 0x0000;
    udelay(50);
}
```

// enable bit: low(0) → high(1) → low(0)

제어 신호			제어명령							
RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0/1	0	0	0	0	0	0	0	0

# Text LCD device driver

```
void writebyte(char ch)
{
    unsigned short data;
    data = ch & 0x00FF;

    *textlcd = data | 0x400;
    *textlcd = data | 0x500;
    *textlcd = data | 0x400;
    udelay(50);
}
```

제어 신호			제어명령							
RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
I	0	I/O	0	0	0	0	0	0	0	0

Data  
write

```
void initialize_textlcd(void)
{
    function_set(2,0);           // Function Set:8 bit,display 2 lines,5x7 mod
    display_control(1,0,0);      // Display on, Cursor off
    clear_display();             // Display clear
    return_home();               // go home
    entry_mode_set(1,0);         // Entry Mode Set: shift right cursor
    udelay(2000);
}
```

# Text LCD device driver

```
int function_set(int rows, int nfonts){
    unsigned short command = 0x30;

    if(rows == 2) command |= 0x08;
    else if(rows == 1) command &= 0xf7;
    else return -1;

    command = nfonts ? (command | 0x04) : command;
    setcommand(command);

    return 1;
}
```

기능	제어 신호			제어명령							
Function Set	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	1	DL	N	F	0	0

8-bits 2-lines 5\*7

```
int display_control(int display_enable, int cursor_enable, int nblink){
    unsigned short command = 0x08;
    command = display_enable ? (command | 0x04) : command;
    command = cursor_enable ? (command | 0x02) : command;
    command = nblink ? (command | 0x01) : command;
    setcommand(command);

    return 1;
}
```

기능	제어 신호			제어명령							
Display On/Off Control	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	1	D	C	B

Display Cursor Blink  
on off off

# Text LCD device driver

```
int cursor_shift(int set_screen, int set_rightshit){
    unsigned short command = 0x10;                // 0001 0000
    command = set_screen ? (command | 0x08) : command;
    command = set_rightshit ? (command | 0x04) : command;
    setcommand(command);

    return 1;
}
```

기능	제어 신호			제어명령							
	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Cursor or display shift	0	0	0	0	0	0	I	S/C	R/L	0	0

```
int entry_mode_set(int increase, int nshift){
    unsigned short command = 0x04;                // (1,0)
    command = increase ? (command | 0x2) : command; // 0000 0100
    command = nshift ? (command | 0x1) : command;  // 0000 0110
    setcommand(command);
    return 1;
}
```

기능	제어 신호			제어명령							
	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Entry mode set	0	0	0	0	0	0	0	0	I	I	0

Cursor  
increase



# Text LCD device driver

```
int return_home(){
    unsigned short command = 0x02;           // 0000 0010
    setcommand(command);

    udelay(2000);
    return 1;
}
```

기능	제어 신호			제어명령							
Return home	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	0	0	1	0

```
int clear_display(){
    unsigned short command = 0x01;           // 0000 0001
    setcommand(command);

    udelay(2000);
    return 1;
}
```

기능	제어 신호			제어명령							
Clear display	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	0	0	0	1

# Text LCD device driver

```
int set_ddram_address(int pos){
    unsigned short command = 0x80;                // 1000 0000
    command += pos;
    setcommand(command);
    return 1;
}
```

기능	제어 신호			제어명령							
Set DD RAM address	RS	RW	E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	1							

DD RAM address

```
static int textlcd_open(struct inode *minode, struct file *mfile)
{
    if(textlcd_usage != 0) return -EBUSY;

    textlcd_ioremap= (unsigned int)ioremap(TEXTLCD_ADDRESS,TEXTLCD_ADDRESS_RANGE);
    if(!check_mem_region(textlcd_ioremap,TEXTLCD_ADDRESS_RANGE))
        request_region(textlcd_ioremap,TEXTLCD_ADDRESS_RANGE,TEXTLCD_NAME);
    else
        printk(KERN_WARNING"Can't get IO Region 0x%x\n",TEXTLCD_ADDRESS);
    textlcd_usage = 1;

    textlcd = (unsigned short *)textlcd_ioremap;
    initialize_textlcd();
    return 0;
}
```

# Text LCD device driver

---

```
static int textlcd_release(struct inode *minode, struct file *mfile)
{
    release_region(textlcd_ioremap, TEXTLCD_ADDRESS_RANGE);
    iounmap((unsigned short *)textlcd_ioremap);
    textlcd_usage = 0;
    return 0;
}

static ssize_t textlcd_write(struct file *inode, const char *gdata, size_t length, loff_t *off_what)
{
    int i, ret;
    char buf[100];

    ret = copy from user(buf, gdata, length);
    if (ret < 0) return -1;

    for (i=0; i<length; i++)
        writebyte(buf[i]);

    return length;
}
```

# Text LCD device driver

---

```
static int textlcd_ioctl(struct inode *inode, struct file *file, unsigned int cmd, unsigned long gdata)
{
    struct strcommand_variable strcommand;
    int ret;

    ret = copy_from_user(&strcommand, (char *)gdata, 32);
    if(ret < 0) return -1;

    switch(cmd)
    {
        case TEXTLCD_COMMAND_SET:
            setcommand(strcommand.command);
            break;

        case TEXTLCD_FUNCTION_SET:
            function_set((int)(strcommand.rows+1), (int)(strcommand.nfonts));
            break;

        case TEXTLCD_DISPLAY_CONTROL:
            display_control((int)strcommand.display_enable,
                (int)strcommand.cursor_enable, (int)strcommand.nblink);
            break;

        case TEXTLCD_CURSOR_SHIFT:
            cursor_shit((int)strcommand.set_screen, (int)strcommand.set_rightshit);
            break;
    }
}
```

# Text LCD device driver

---

```
case TEXTLCD_ENTRY_MODE_SET:
    entry_mode_set((int)strcommand.increase, (int)strcommand.nshift);
    break;

case TEXTLCD_RETURN_HOME:
    return_home();
    break;

case TEXTLCD_CLEAR:
    clear_display();
    break;

case TEXTLCD_DD_ADDRESS:
    set_ddram_address((int)strcommand.pos);
    break;

case TEXTLCD_WRITE_BYTE:
    writebyte(strcommand.buf[0]);
    break;

default:
    printk("driver : no such command!\n");
    return -ENOTTY;
}
return 0;
}
```

# Text LCD device driver

---

```
static struct file_operations textlcd_fops = {
    .owner          = THIS_MODULE,
    .write          = textlcd_write,
    .ioctl          = textlcd_ioctl,
    .open           = textlcd_open,
    .release        = textlcd_release,
};

int textlcd_init(void)
{
    int result;
    result = register_chrdev(TEXTLCD_MAJOR, TEXTLCD_NAME, &textlcd_fops);

    if(result < 0) {
        printk(KERN_WARNING "Can't get any major\n");
        return result;
    }
    textlcd_major = result;
    printk("init module, textlcd major number : %d\n", result);
    return 0;
}
```

# Text LCD device driver

---

```
void textlcd_exit(void)
{
    unregister_chrdev(textlcd_major, TEXTLCD_NAME);
}

module_init(textlcd_init);
module_exit(textlcd_exit);

MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
MODULE_LICENSE("GPL");
```

# Text LCD device driver

```
/* textlcd.h */
```

```
#define DRIVER_AUTHOR  
#define DRIVER_DESC  
#define TEXTLCD_MAJOR  
#define TEXTLCD_NAME  
#define TEXTLCD_MODULE_VERSION  
#define TEXTLCD_ADDRESS  
#define TEXTLCD_ADDRESS_RANGE  
  
#define TEXTLCD_MAGIC  
#define TEXTLCD_COMMAND_SET  
#define TEXTLCD_FUNCTION_SET  
#define TEXTLCD_DISPLAY_CONTROL  
#define TEXTLCD_CURSOR_SHIFT  
#define TEXTLCD_ENTRY_MODE_SET  
#define TEXTLCD_RETURN_HOME  
#define TEXTLCD_CLEAR  
#define TEXTLCD_DD_ADDRESS  
#define TEXTLCD_WRITE_BYTE
```

```
//Global variable  
static int textlcd_usage = 0;  
static int textlcd_major = 0;  
static unsigned int textlcd_ioremap;  
static unsigned short *textlcd;
```

```
"Hanback Electronics"  
"textlcd test program"  
0  
"TEXT LCD PORT"  
"TEXT LCD PORT V0.1"  
0x14809000  
0x1000
```

```
'b'  
_IOW(TEXTLCD_MAGIC,0,int)  
_IOW(TEXTLCD_MAGIC,1,int)  
_IOW(TEXTLCD_MAGIC,2,int)  
_IOW(TEXTLCD_MAGIC,3,int)  
_IOW(TEXTLCD_MAGIC,4,int)  
_IOW(TEXTLCD_MAGIC,5,int)  
_IOW(TEXTLCD_MAGIC,6,int)  
_IOW(TEXTLCD_MAGIC,7,int)  
_IOW(TEXTLCD_MAGIC,8,int)
```

\_IOW(type, number, datatype)  
- One of macros that help set up the command numbers  
type: magic number(eight bits)  
number: sequential number(eight bits)  
datatype: user data type



# Text LCD device driver

---

```
struct strcommand_variable {
    char rows;
    char nfonts;
    char display_enable;
    char cursor_enable;
    char nblink;
    char set_screen;
    char set_rightshit;
    char increase;
    char nshift;
    char pos;
    char command;
    char strlength;
    char buf[16];
};

void setcommand(unsigned short command);
void usr_wait(unsigned long delay_factor);
void writebyte(char ch);
void initialize_textlcd(void);
int function_set(int rows, int nfonts);
int display_control(int display_enable, int cursor_enable, int nblink);
int cursor_shit(int set_screen, int set_rightshit);
int entry_mode_set(int increase, int nshift);
int return_home(void);
int clear_display(void);
int set_ddram_address(int pos);
```

# Application for Text LCD driver test

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include "textlcd.h"

int main(int argc, char **argv)
{
    int i,dev;
    char buf[13] = "Hanback.co.kr";
    char wellcom[14] = "Welcome to the";
    char wellcom2[16] = "embedded World!!";
    struct strcommand_variable strcommand;

    strcommand.rows = 0;
    strcommand.nfonts = 0;
    strcommand.display_enable = 1;
    strcommand.cursor_enable = 0;
    strcommand.nblink = 0;
    strcommand.set_screen = 0;
    strcommand.set_rightshit = 1;
    strcommand.increase = 1;
    strcommand.nshift = 0;
    strcommand.pos = 0;
    strcommand.command = 1;
    strcommand.strlength = 16;
```

# Application for Text LCD driver test

```
dev = open("/dev/textlcd", O_WRONLY|O_NDELAY );

if (dev != -1) {
    write(dev,buf,13);

    sleep(2);
    ioctl(dev,TEXTLCD_CLEAR,&strcommand,32);
    strcommand.pos = 0;
    ioctl(dev,TEXTLCD_DD_ADDRESS,&strcommand,32);
    for(i=0;i<14;i++) {
        memcpy(&strcommand.buf[0],&wellcom[i],sizeof(wellcom));
        ioctl(dev,TEXTLCD_WRITE_BYTE,&strcommand,32);
    }
    sleep(2);
    strcommand.pos = 40;
    ioctl(dev,TEXTLCD_DD_ADDRESS,&strcommand,32);
    for(i=0;i<16;i++) {
        memcpy(&strcommand.buf[0],&wellcom2[i],sizeof(wellcom2));
        ioctl(dev,TEXTLCD_WRITE_BYTE,&strcommand,32);
    }
    close(dev);
} else {
    printf( "application : Device Open ERROR!\n");
    exit(1);
}
return 0;
}
```

```
int ioctl(int fd, unsigned long cmd, ...);
- the third argument depends on the second argument.
e.g.) no arguments, an integer value, and a pointer to other data
```