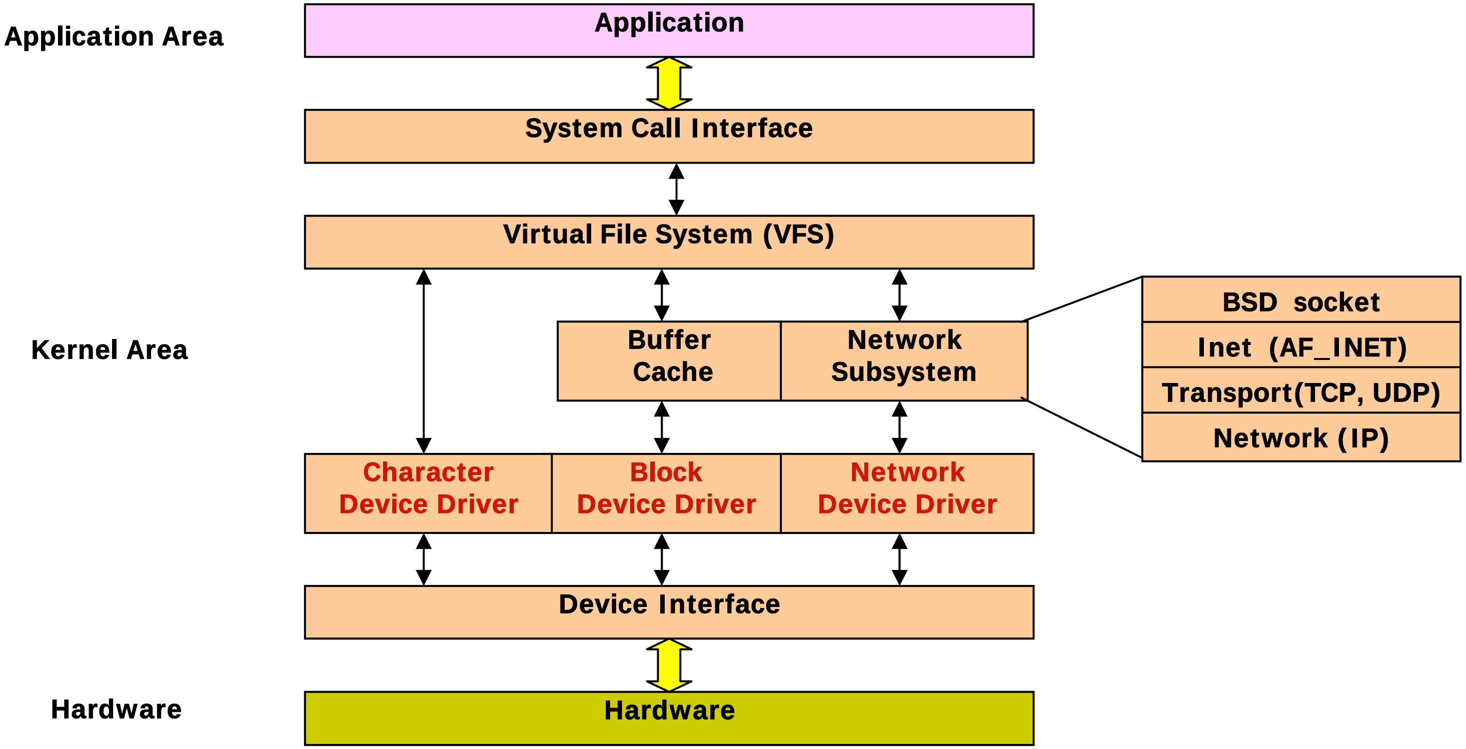
Embedded Systems

Prof. Myung-Eui Lee (F-102)

melee@kut.ac.kr

한국기술교육대학교
korea University of
Technology and Education

Device driver

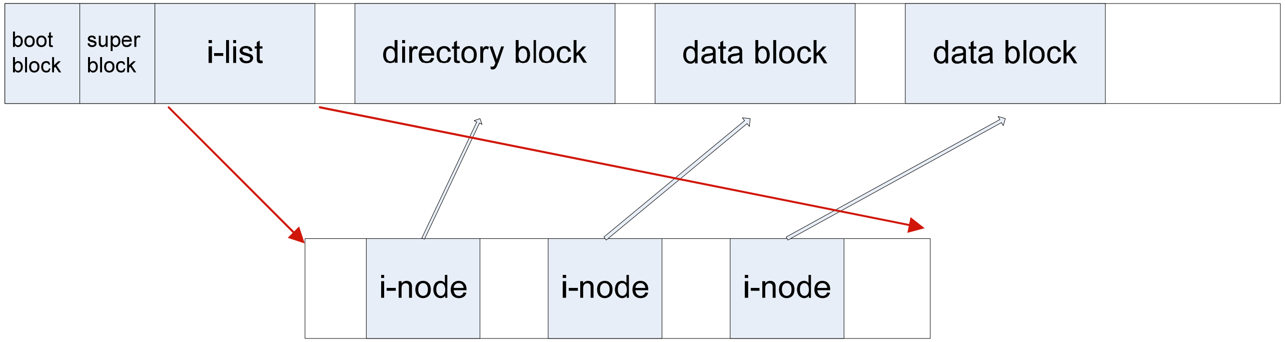


VFS

- **Virtual File System**
  - **Not a file system on its own but an interface**
  - **Kernel software layer that handles all system calls related to a stand Unix file system**
  - **Link between the operating system kernel and the different file systems**
  - **Supplies the applications with the system calls for file management**
  - **Passes tasks on to the appropriate actual file system**
- **File system supported by VFS**
  - **Disk-based file system**
    - Ext2/3, dos, vfat, ntfs, hpfs (IBM's OS/2), hfs (Apple's Macintosh)
  - **Network file system**
    - NFS, SMB, NCP(Novell's NetWare Core Protocol)
  - **Special file system (virtual file system)**
    - /proc, /dev/*, rootfs

VFS

- **VFS objects**
  - **superblock object : specific mounted filesystem**
    - struct super_block, struct super_operations (include/linux/fs.h)
  - **inode object : represents a specific file**
    - struct inode, struct inode_operations (include/linux/fs.h)
  - **dentry object : represents directory entry**
    - struct dentry, struct dentry_operations (include/linux/dcache.h)
  - **file object : represents a file opened by a process**
    - struct file, struct file_operations (include/linux/fs.h)



struct file

Type	Field	Description
struct file *	f_next	Pointer to next file object
struct file **	f_pprev	Pointer to previous file object
struct dentry *	f_dentry	Pointer to associated dentry object
struct file_operations *	f_op	Pointer to file operation table
mode_t	f_mode	Process access mode
loff_t	f_pos	Current file offset (file pionter)
unsigned int	f_count	File object's usage counter
unsigned int	f_flags	Flags specified when opening the file
unsigned long	f_reada	Read-ahead flag
unsigned long	f_ramax	Maximum number of pages to be read-ahead
unsigned long	f_raend	File pointer after last read-ahead
unsigned long	f_ralen	Number of read-ahead bytes
unsigned long	f_rawin	Number of read-ahead pages
struct fown_struct	f_owner	Data for asynchronous I/O via signals
unsigned int	f_uid	User's UID
unsigned int	f_gid	User's GID
int	f_error	Error code for networkwrite operation
unsigned long	f_version	Version number, automatically incremented after each use
void *	private_data	Needed for tty driver

"Understanding the Linux Kernel", O'reilly, 2001, Ver. 2.4

struct file_operations

```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*aio_read) (struct kiocb *, char __user *, size_t, loff_t);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*aio_write) (struct kiocb *, const char __user *, size_t, loff_t);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, struct dentry *, int datasync);
        int (*aio_fsync) (struct kiocb *, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*readv) (struct file *, const struct iovec *, unsigned long, loff_t *);
        ssize_t (*writev) (struct file *, const struct iovec *, unsigned long, loff_t *);
        ssize_t (*sendfile) (struct file *, loff_t *, size_t, read_actor_t, void *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
        int (*check_flags)(int);
        int (*dir_notify)(struct file *filp, unsigned long arg);
        int (*flock) (struct file *, int, struct file_lock *);
        ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
        ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
 };
```

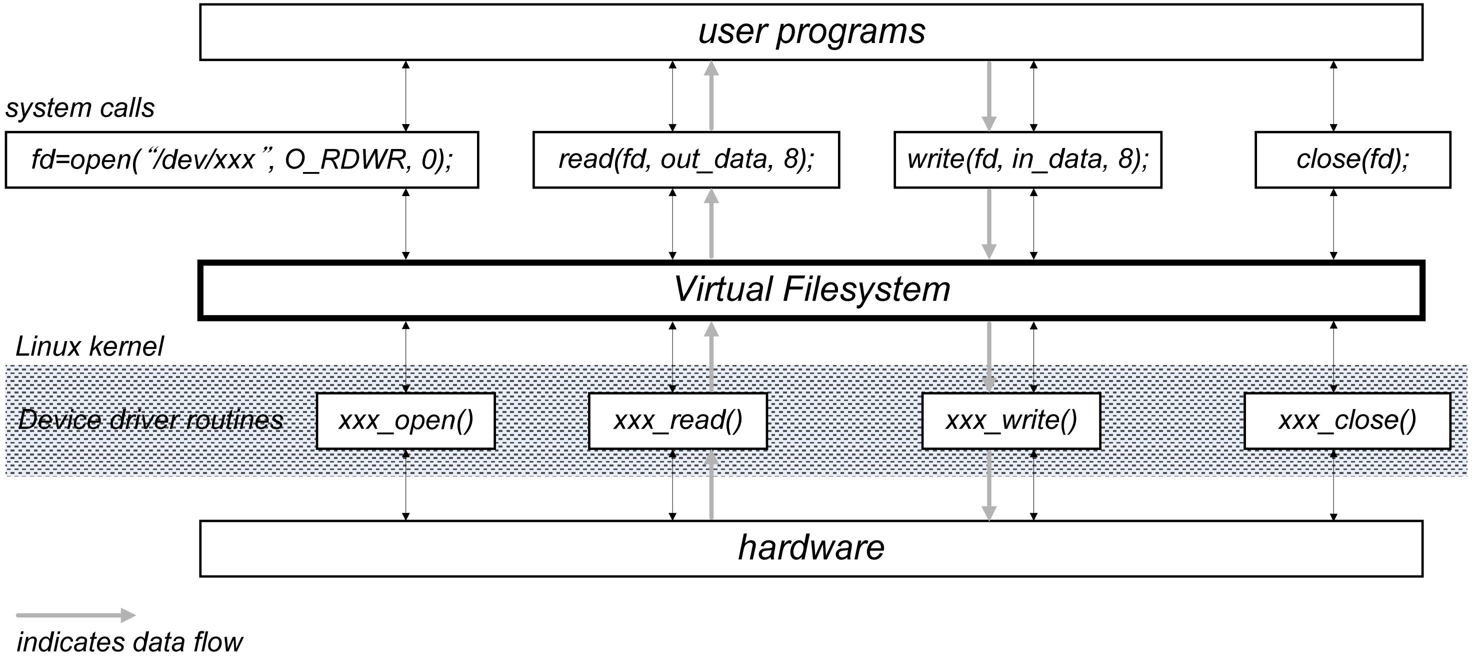
file operations

- llseek(file, offset, int) Updates the file pointer to given offset
- read(file, buf, count, offset) Reads count bytes from a file starting at position *offset into buf; the value *offset (which usually corresponds to the file pointer) is then incremented.
- write(file, buf, count, offset) Writes count bytes into a file starting at position *offset from buf; the value (which usually corresponds to the file pointer) is then incremented.
- readdir(dir, dirent, filldir) Returns the next directory entry of a directory in dirent; the filldir parameter contains the address of an auxiliary function that extracts the fields in a directory entry.
- poll(file, poll_table) Checks whether there is activity on a file and goes to sleep until something happens on it.
- ioctl(inode, file, cmd, arg) Sends a command to an underlying hardware device. This method applies only to device files.
- mmap(file, vma) Performs a memory mapping of the file into a process address space.
- open(inode, file) Opens a file by creating a new file object and linking it to the corresponding inode object.

file operations

- **flush(file)** Called when a reference to an open file is closed, that is, the data in buff is deleted.
- **release(inode, file)** = close Releases the file object
- **fsync(file, dentry)** Writes all cached data of the file to disk.
- **fasync(file, int)** Enables or disables asynchronous I/O notification by means of signals.
- **check_media_change(dev)** Checks whether there has been a change of media since the last operation on the device file (applicable to block devices that support removable media, such as floppies and CD-ROMs).
- **revalidate(dev)** Restores the consistency of a device (used by network filesystems after a media change has been recognized on a remote device).
- **lock(file, cmd, file_lock)** Applies a lock to the file

Device driver



Device driver

- **Device file types**

  cd /dev, ls -al

  - **1. character : c**
    - byte (sequential) process, no buffer cache
  - **2. block : b**
    - block (random) process, buffer cache
  - **3. network**
    - packet process, socket (protocol stack)

```
…
brw-r-----     1   root   disk  3  0   Oct   3    2006  hda
brw-r-----     1   root   disk  3  1   Oct   3    2006 hda1
brw-r-----     1   root   disk  3  2   Oct   3    2006  hda2
brw-r-----     1   root   disk  3  3   Oct   3    2006  hda3
…
crw--w--w--    1  card    tty    4  0  May  16   2006  tty0
crw--w--w--    1  card    tty    4  1  May  16   2006  tty1
crw--w--w--    1  card    tty    4  2  May  16   2006  tty2
…
```

Device driver

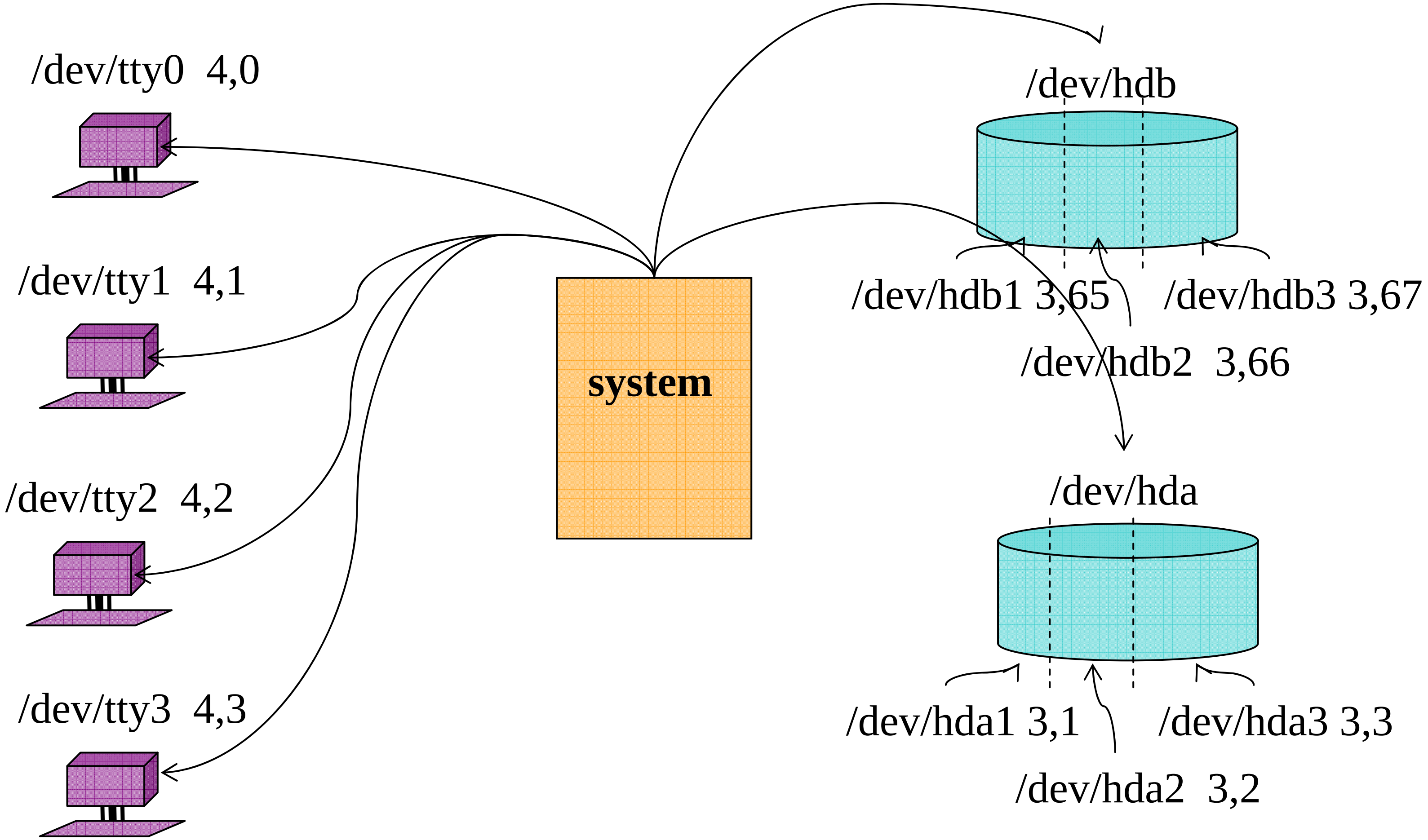
● Device file

- » **major number**
  - ◆ include/linux/major.h , Documentation/devices.txt , more /proc/devices
- » **minor number**

Major	Character devices	Block devices
0	unnamed	unnamed
1	physical memory access(/dev/mem)	First RAM disk(dev/ram0)
2	Kernel virtual memory(/dev/kmem)	floppy (fd*)
3		IDE hard disk (hd*)
4	terminal	
5	terminal & AUX	
6	Parallel Interface	
7	virtual console (vcs*)	
8		SCSI hard disk (sd*)
9	SCSI tapes (st*)	
10	Bus mice(bm, psaux)	
11		SCSI CD-ROM(scd*)
………		
23		Mitsumi CD-ROM (mcd*)
….		

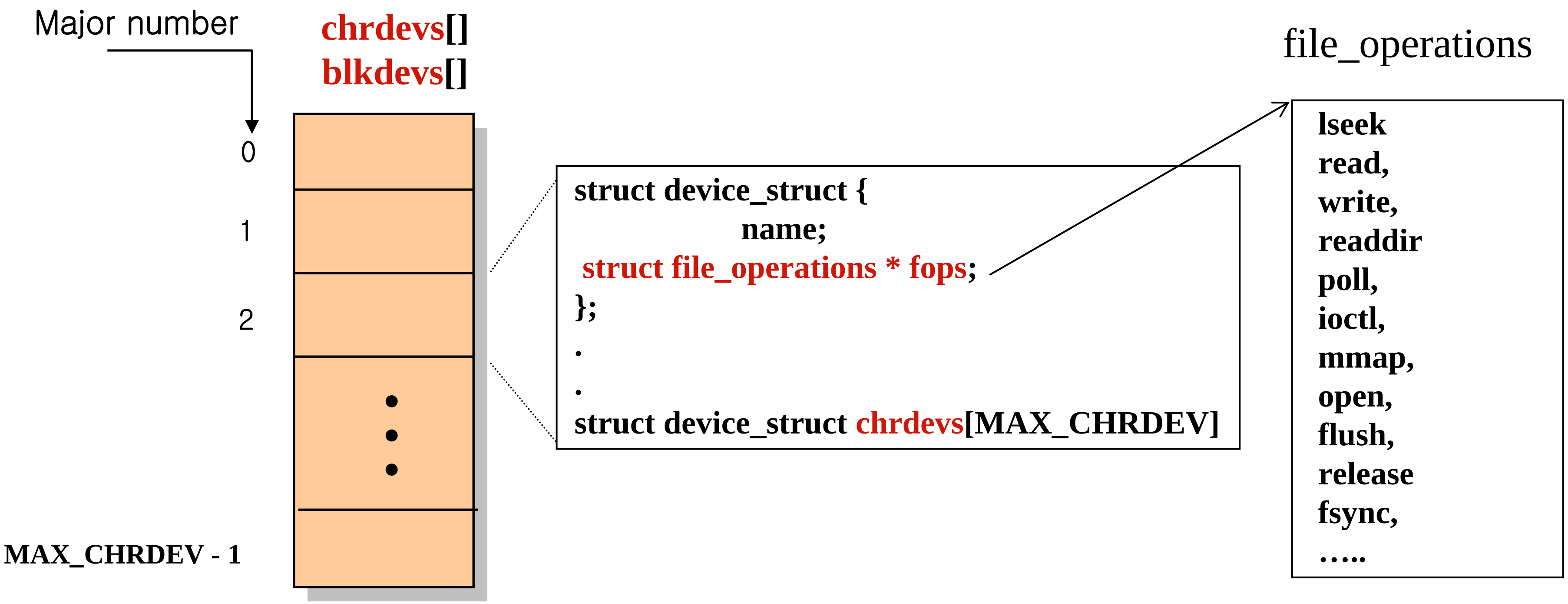
Device number

- » **Major : device type**
- » **Minor : device unit**



Device Number

- chrdevs[] : fs/devices.c
  - » insmod, open



```
struct device_struct {
                    name;
 struct file_operations * fops;
};
.
.
struct device_struct chrdevs[MAX_CHRDEV]
```

lseek
read,
write,
readdir
poll,
ioctl,
mmap,
open,
flush,
release
fsync,
…..

Memory Mapped I/O

Memory Mapped I/O

0xFFFF FFFF

Reserved(1024Mbytes)

0xB000 0000

SDRAM BANK 3(64Mbytes)

0xAC00 0000

SDRAM BANK 2(64Mbytes)

0xA800 0000

SDRAM BANK 1(64Mbytes)

0xA400 0000

SDRAM BANK 0(64Mbytes)

0xA000 0000

Reserved(1344Mbytes)
Internal SRAM
(5C00 0000 - 5C03 FFFF)

0x4C00 0000

Memory Mapped registers (Memory Ctl)
(64Mbyte)

0x4800 0000

Memory Mapped registers (LCD)
(64Mbyte)

0x4400 0000

Memory Mapped registers (Peripherals)
(64MBytes)

0x4000 0000

PCMCIA Socket 1 Space
(256Mbyte)

0x3000 0000

PCMCIA Socket 0 Space
(256Mbyte)

0x2000 0000

Reserved(128 Mbytes)

0x1800 0000

Static Bank Select 5(64Mbytes)

0x1400 0000

Static Bank Select 4(64Mbytes)

0x1000 0000

Static Bank Select 3(64Mbytes)

0x0C00 0000

Static Bank Select 2(64Mbytes)

0x0800 0000

Static Bank Select 1(64Mbytes)

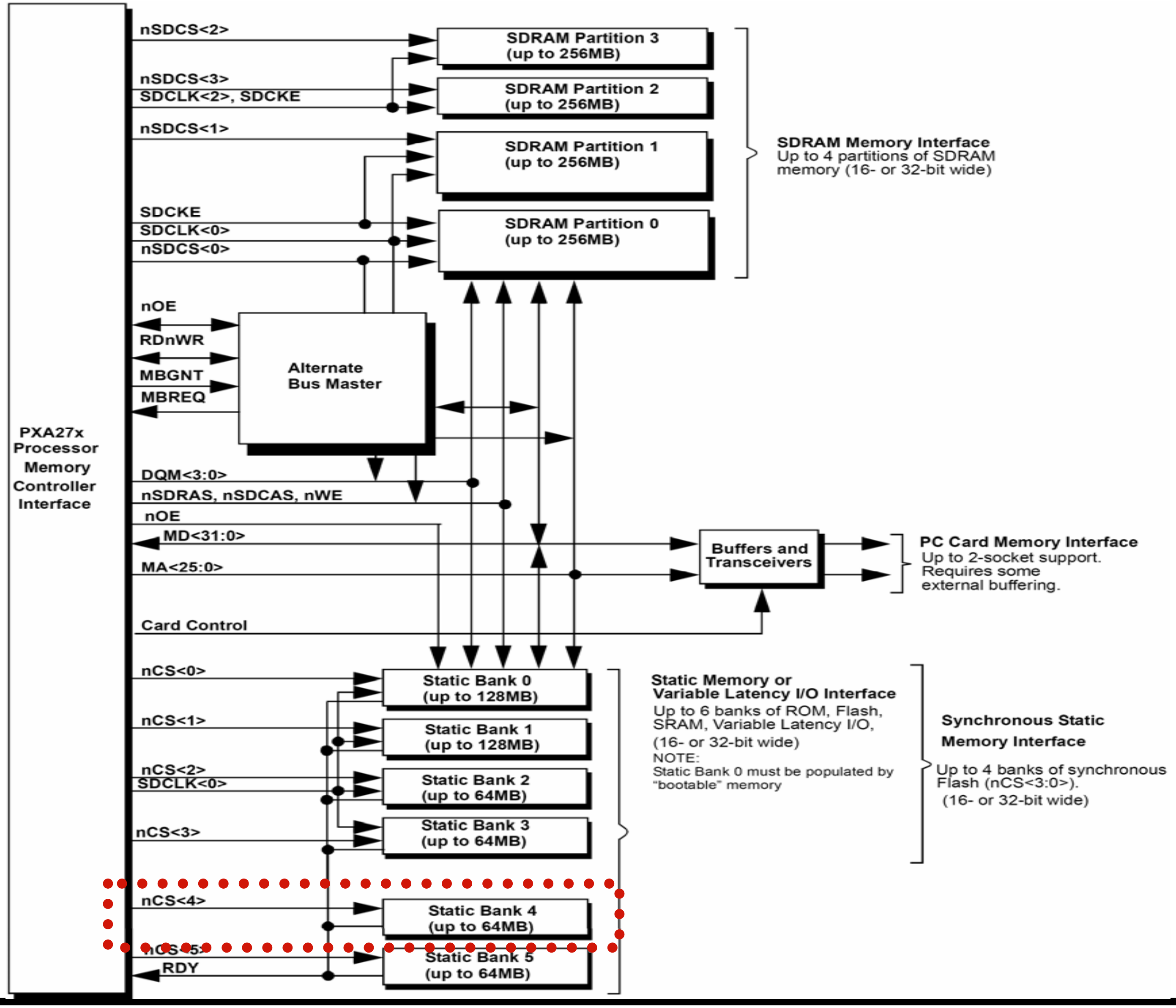
0x0400 0000

Static Bank Select 0(64Mbytes)

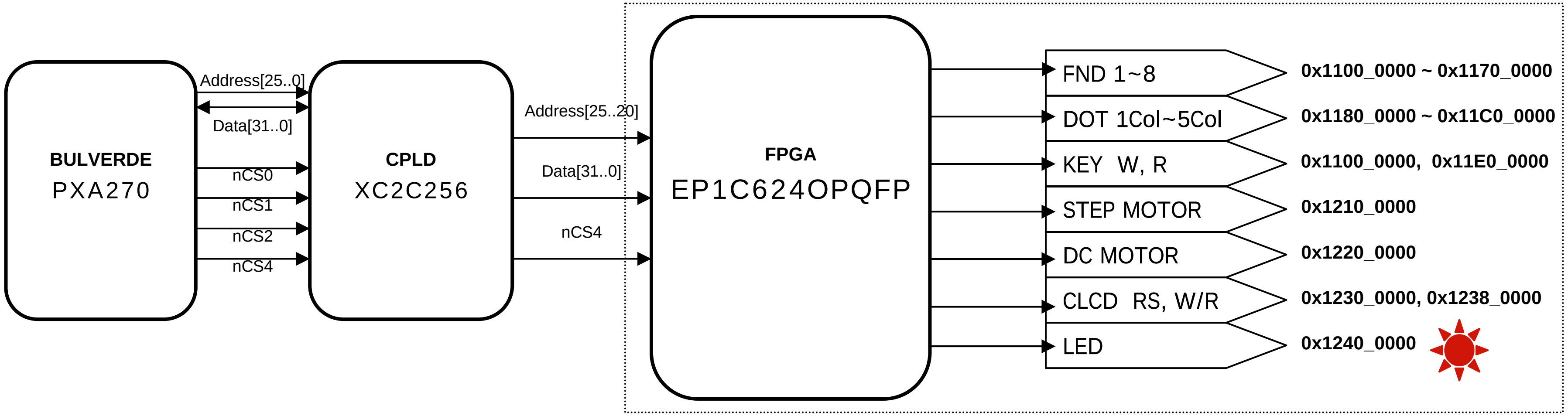
0x0000 0000

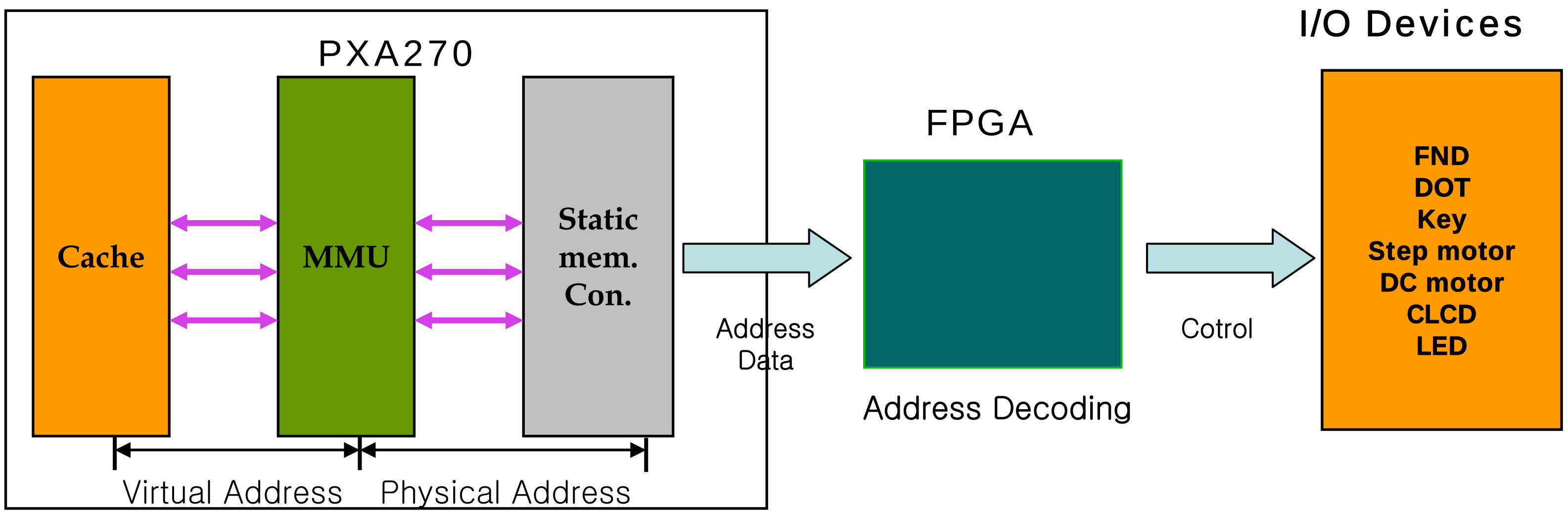
Name	Address	Size	Description
FND1	0x11000000	0x100000	7-segment 1
FND2	0x11100000	0x100000	7-segment 2
FND3	0x11200000	0x100000	7-segment 3
FND4	0x11300000	0x100000	7-segment 4
FND5	0x11400000	0x100000	7-segment 5
FND6	0x11500000	0x100000	7-segment 6
FND7	0x11600000	0x100000	7-segment 7
FND8	0x11700000	0x100000	7-segment 8
DOT_1 Col	0x11800000	0x100000	Dot column 1
DOT_2 Col	0x11900000	0x100000	Dot column 2
DOT_3 Col	0x11A00000	0x100000	Dot column 3
DOT_4 Col	0x11B00000	0x100000	Dot column 4
DOT_5 Col	0x11C00000	0x100000	Dot column 5
KEY_W	0x11D00000	0x100000	Key Write
KEY_R	0x11E00000	0x100000	Key Read
DAC	0x11F00000	0x100000	Digital to Analog Converter
ADC	0x12000000	0x100000	Analog to Digital Converter
STEP Motor	0x12100000	0x100000	Step Motor controller
DC Motor	0x12200000	0x100000	DC Motor controller
Character LCD	0x12300000	0x100000	Character LCD Control
LED	0x12400000	0x100000	LED

nCS<4>


nSDCS<2>
SDRAM Partition 3
(up to 256MB)
nSDCS<3>
SDCLK<2>, SDCKE
SDRAM Partition 2
(up to 256MB)
nSDCS<1>
SDRAM Partition 1
(up to 256MB)
SDCKE
SDCLK<0>
nSDCS<0>
SDRAM Partition 0
(up to 256MB)
SDRAM Memory Interface
Up to 4 partitions of SDRAM
memory (16- or 32-bit wide)
nOE
RDnWR
MBGNT
MBREQ
Alternate
Bus Master
PXA27x
Processor
Memory
Controller
Interface
DQM<3:0>
nSDRAS, nSDCAS, nWE
nOE
MD<31:0>
MA<25:0>
Buffers and
Transceivers
PC Card Memory Interface
Up to 2-socket support.
Requires some
external buffering.
Card Control
nCS<0>
Static Bank 0
(up to 128MB)
nCS<1>
Static Bank 1
(up to 128MB)
nCS<2>
SDCLK<0>
Static Bank 2
(up to 64MB)
nCS<3>
Static Bank 3
(up to 64MB)
nCS<4>
Static Bank 4
(up to 64MB)
nCS<5>
RDY
Static Bank 5
(up to 64MB)
Static Memory or
Variable Latency I/O Interface
Up to 6 banks of ROM, Flash,
SRAM, Variable Latency I/O,
(16- or 32-bit wide)
NOTE:
Static Bank 0 must be populated by
"bootable" memory
Synchronous Static
Memory Interface
Up to 4 banks of synchronous
Flash (nCS<3:0>).
(16- or 32-bit wide)

Address Mapping

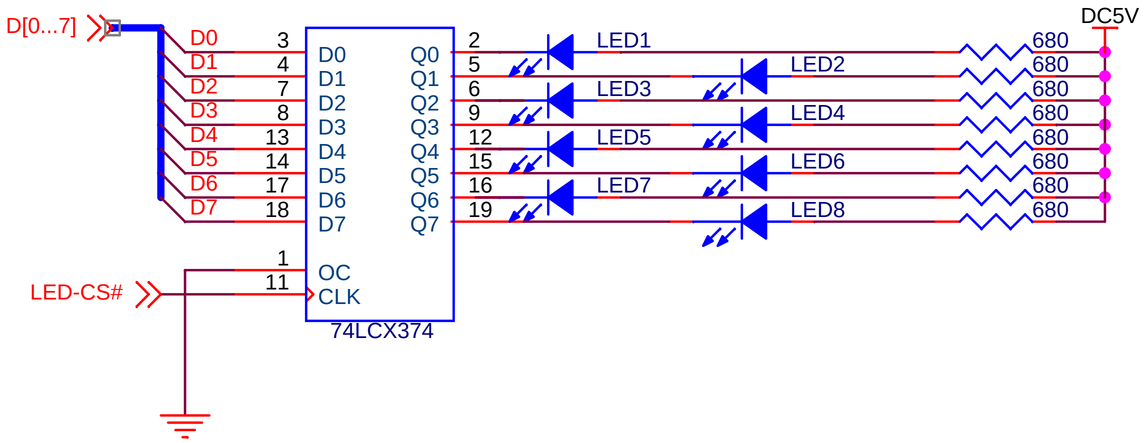




Address Decoder

- **LED**
  - 8Bit Write [D0~D7]
  - Base Address = 0x12400000

add	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
led	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16진수	1				2				4				0				0				0				0				0			



Device driver

- **Device driver (not using OS)**
  - » **LED Example : boot-led (microcom)**
  - » **main.c**

```
#include <config.h>
#include <time.h>

#define IEB_LED_CS          0x12400000
#define LED_CS              (*((volatile unsigned char *)(IEB_LED_CS)))

int main(void)
{
            time_init();                        // Timer Initialization
            LED_CS = 0xFF;                      // LED all off

            /* 8-Line BUS LEDs Control */
            while(1)
            {
                        LED_CS = 0xEE;
                        mdelay(1000);           // 1sec
                        LED_CS = 0xDD;
                        mdelay(1000);
                        LED_CS = 0xBB;
                        mdelay(1000);
                        LED_CS = 0x77;
                        mdelay(1000);
            }
}
```

no cache, no optimization
Always Read/Write

*** I/O Write**

```
unsigned char *addr;
addr = (unsigned char *)(0x12400000);
*addr = 0xaa;
```

*** I/O Read**

```
unsigned char *addr, char ch;
addr = (unsigned char *)(0x12400000);
ch = *addr;
```

Test run

- **Flash fusing**
  - » **1. Jtag**
    - ./jflashmm pxa27x32.dat ledtest
  - » **2. boot monitor**
    - cp ledtest /tftpboot
    - tftp ledtest loader (target)
    - flash loader

mmap

- ## Dot matrix

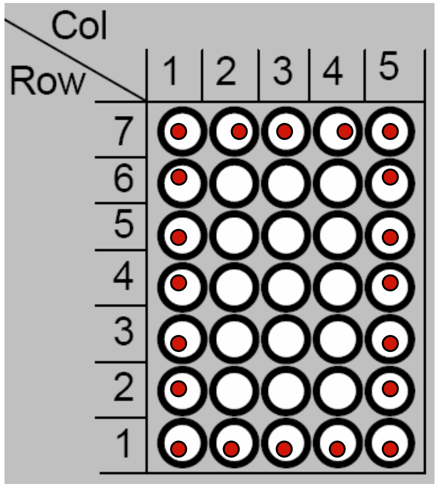
Physical Address

```
#define    ADDRESSDOT1      0x11800000
#define    ADDRESSDOT2      0x11900000
#define    ADDRESSDOT3      0x11A00000
#define    ADDRESSDOT4      0x11B00000
#define    ADDRESSDOT5      0x11C00000
```

Virtual Address

```
pled1 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT1);
pled2 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT2);
pled3 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT3);
pled4 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT4);
pled5 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT5);
```

Row7 : data[6]
Row6 : data[5]
Row5 : data[4]
Row4 : data[3]
Row3 : data[2]
Row2 : data[1]
Row1 : data[0]



	Number 0
Col1 : 0x11800000	7F
Col2 : 0x11900000	41
Col3 : 0x11A00000	41
Col4 : 0x11B00000	41
Col5 : 0x11C00000	7F

mmap

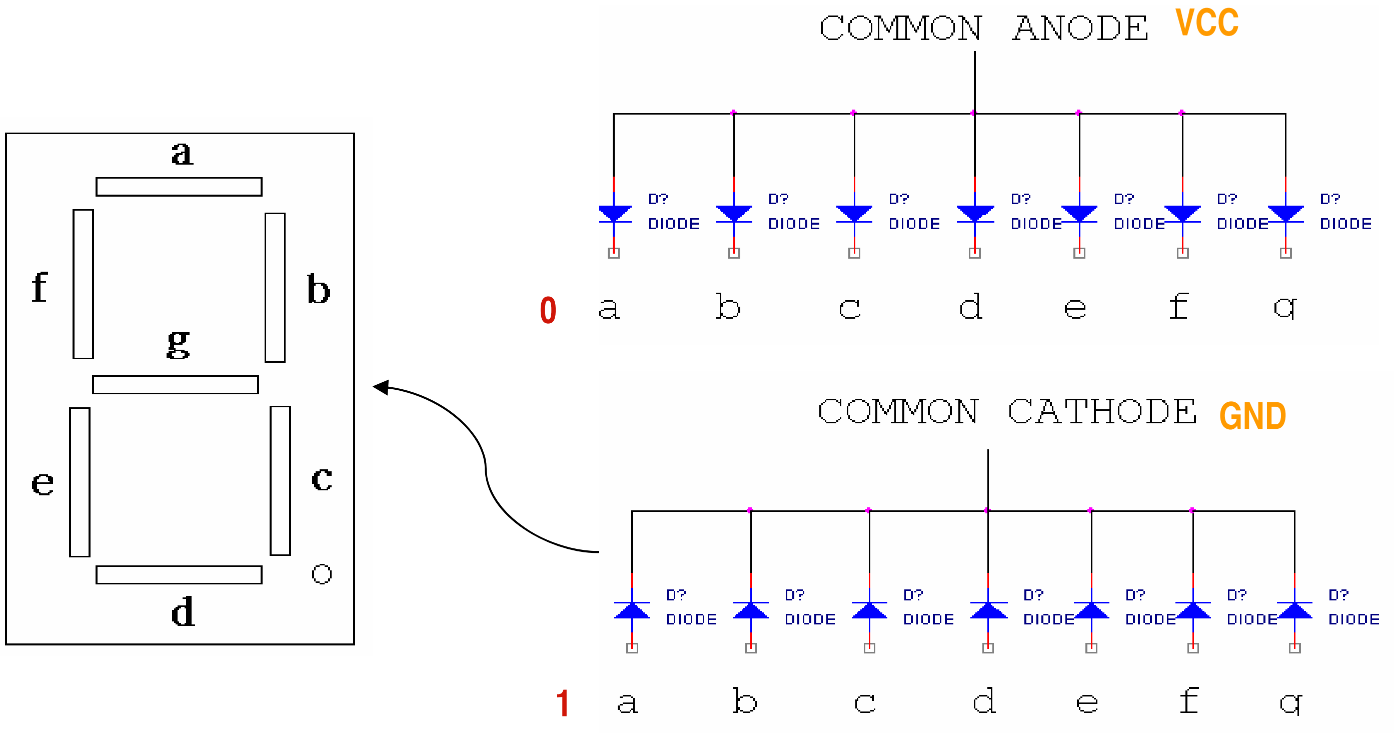
```
unsigned char dot_col[5] = {0};                              // Dot Data

#define    ADDRESSDOT1      0x11800000
#define    ADDRESSDOT2      0x11900000
#define    ADDRESSDOT3      0x11A00000
#define    ADDRESSDOT4      0x11B00000
#define    ADDRESSDOT5      0x11C00000

unsigned short *pled1;
unsigned short *pled2;
unsigned short *pled3;
unsigned short *pled4;
unsigned short *pled5;

void dot_init(void);
void dot_reset(void);
void asc_to_dot(int);

int main()
{                int count, unsigned int fd;
                 if ((fd=open("/dev/mem",O_RDWR|O_SYNC)) < 0){
                                 perror("mem open fail\n");
                                 exit(1);
                 }
                 dot_init();
                 dot_reset();
                 for(count =0 ; count <= 9; count++){   // number 0 – 9 display
                                 asc_to_dot(count);
                                 *pled1 = dot_col[0];
                                 *pled2 = dot_col[1];
                                 *pled3 = dot_col[2];
                                 *pled4 = dot_col[3];
                                 *pled5 = dot_col[4];
                                 sleep(1);
                 }
                 return 0;
}
```

mmap

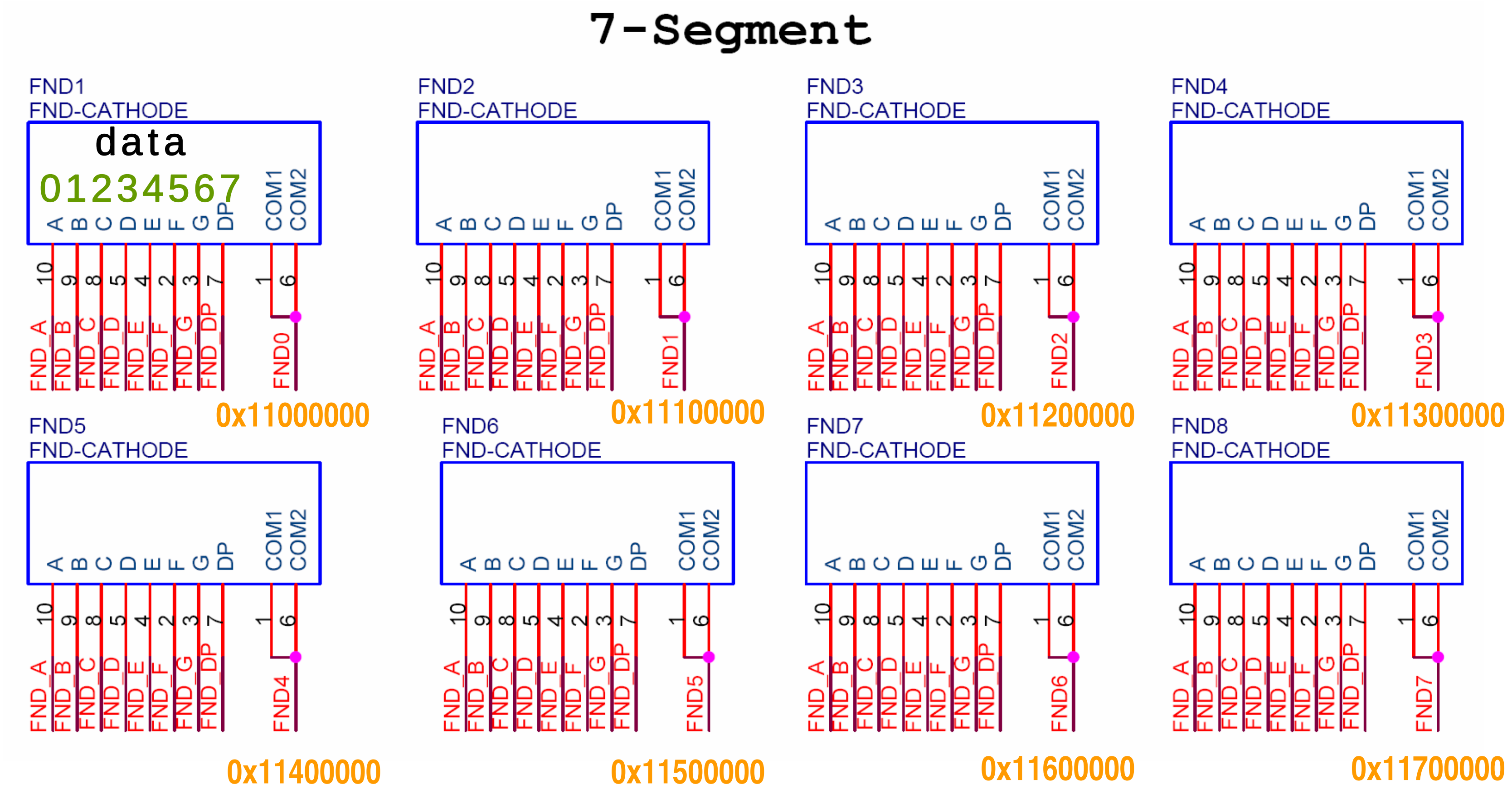
```
void dot_init(void)
{                     pled1 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT1);
                      pled2 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT2);
                      pled3 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT3);
                      pled4 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT4);
                      pled5 = mmap(NULL,1,PROT_WRITE,MAP_SHARED,fd,ADDRESSDOT5);
                      return;
}

void dot_reset(void){
// DOT Matrix Clear
   *pled1      = 0x00;
   *pled2      = 0x00;
   *pled3      = 0x00;
   *pled4      = 0x00;
   *pled5      = 0x00;
   return;
}

// Conversion from ASCII to DOT Data
void asc_to_dot(int asc){
   switch( asc){
      case 0 :  dot_col[0] = 0x7F; dot_col[1] = 0x41; dot_col[2] = 0x41; dot_col[3] = 0x41; dot_col[4] = 0x7F; break;
      case 1 :  dot_col[0] = 0x00; dot_col[1] = 0x00; dot_col[2] = 0x7F; dot_col[3] = 0x00; dot_col[4] = 0x00; break;
      case 2 :  dot_col[0] = 0x4F; dot_col[1] = 0x49; dot_col[2] = 0x49; dot_col[3] = 0x49; dot_col[4] = 0x79; break;
      case 3 :  dot_col[0] = 0x49; dot_col[1] = 0x49; dot_col[2] = 0x49; dot_col[3] = 0x49; dot_col[4] = 0x7F; break;
      case 4 :  dot_col[0] = 0x78; dot_col[1] = 0x08; dot_col[2] = 0x7F; dot_col[3] = 0x08; dot_col[4] = 0x08; break;
      case 5 :  dot_col[0] = 0x79; dot_col[1] = 0x49; dot_col[2] = 0x49; dot_col[3] = 0x49; dot_col[4] = 0x4F; break;
      case 6 :  dot_col[0] = 0x7F; dot_col[1] = 0x49; dot_col[2] = 0x49; dot_col[3] = 0x49; dot_col[4] = 0x4F; break;
      case 7 :  dot_col[0] = 0x40; dot_col[1] = 0x40; dot_col[2] = 0x40; dot_col[3] = 0x40; dot_col[4] = 0x7F; break;
      case 8 :  dot_col[0] = 0x7F; dot_col[1] = 0x49; dot_col[2] = 0x49; dot_col[3] = 0x49; dot_col[4] = 0x7F; break;
      case 9 :  dot_col[0] = 0x78; dot_col[1] = 0x48; dot_col[2] = 0x48; dot_col[3] = 0x48; dot_col[4] = 0x7F; break;
   }
   return;
}
```

Device driver

- **FND (7-segment LED)**



Address Mapping



Address Mapping

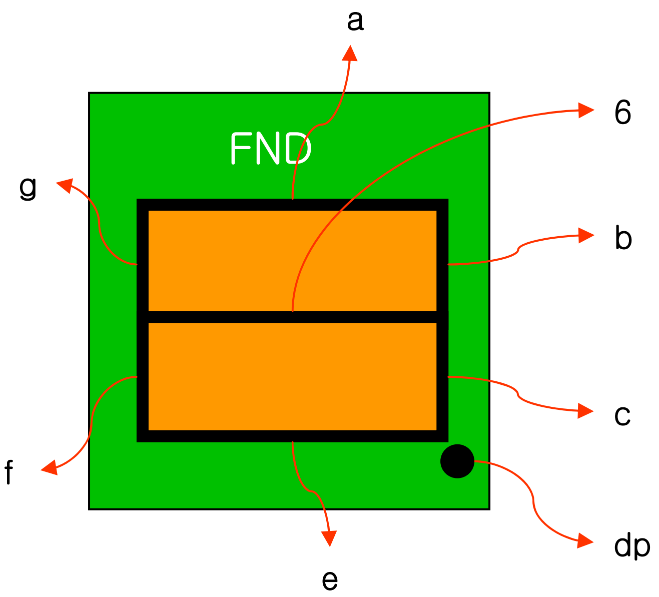
fnd.c

```
#define FND_MAJOR      231
#define FND_NAME       "fnd"
#define MAX_FND         8

#define FPGA_FND_CS0 (0x11000000)
#define FPGA_FND_CS1 (0x11100000)
#define FPGA_FND_CS2 (0x11200000)
#define FPGA_FND_CS3 (0x11300000)
#define FPGA_FND_CS4 (0x11400000)
#define FPGA_FND_CS5 (0x11500000)
#define FPGA_FND_CS6 (0x11600000)
#define FPGA_FND_CS7 (0x11700000)
```

```
mem_addr_fnd0 = FPGA_FND_CS0;
mem_addr_fnd1 = FPGA_FND_CS1;
mem_addr_fnd2 = FPGA_FND_CS2;
mem_addr_fnd3 = FPGA_FND_CS3;
mem_addr_fnd4 = FPGA_FND_CS4;
mem_addr_fnd5 = FPGA_FND_CS5;
mem_addr_fnd6 = FPGA_FND_CS6;
mem_addr_fnd7 = FPGA_FND_CS7;
mem_len  =  0x1000;
```

```
mem_fnd_cs0 = ioremap_nocache ( mem_addr_fnd0, mem_len);
        if( !mem_fnd_cs0) {
                printk("Error mapping fnd0 memory");
                return -EBUSY;
        }

#define FND_CS0     (*((volatile unsigned char *)(mem_fnd_cs0)))
#define FND_CS1     (*((volatile unsigned char *)(mem_fnd_cs1)))
#define FND_CS2     (*((volatile unsigned char *)(mem_fnd_cs2)))
#define FND_CS3     (*((volatile unsigned char *)(mem_fnd_cs3)))
#define FND_CS4     (*((volatile unsigned char *)(mem_fnd_cs4)))
#define FND_CS5     (*((volatile unsigned char *)(mem_fnd_cs5)))
#define FND_CS6     (*((volatile unsigned char *)(mem_fnd_cs6)))
#define FND_CS7     (*((volatile unsigned char *)(mem_fnd_cs7)))
```

Bit Position



a : data[0]
b : data[1]
c : data[2]
d : data[3]
e : data[4]
f : data[5]
g : data[6]
dp : data[7]

fnd-test.c

```
unsigned char asc_to_fnd(int n){
        unsigned char c;

                switch (n) {
                        case  0: c = 0x3f; break;
                        case  1: c = 0x06; break;
                        case  2: c = 0x5b; break;
                        case  3: c = 0x4f; break;
                        case  4: c = 0x66; break;
                        case  5: c = 0x6d; break;
                        case  6: c = 0x7d; break;
                        case  7: c = 0x07; break;
                        case  8: c = 0x7f; break;
                        case  9: c = 0x67; break;
                        default: c = 0x00; break;
                }
        return c;
}
```

FND Device Driver

- **Device Driver : fnd.c**

```
static struct file_operations device_fops = {
        .open =     fnd_open,
        .write =    fnd_write,
        .release =  fnd_release,
        .ioctl =    fnd_ioctl,
};
```

- **fnd_open**
  - **fnd_clear() -> all off (FND_CS0 = 0x00;)**
- **fnd_release : close**
  - **return 0 : release file object**
- **fnd_write**
  - **copy_from_user(disp, buf, count); buf(user), disp(kernel)**
  - **number write (case 1: FND_CS7 = disp[0]; break;)**

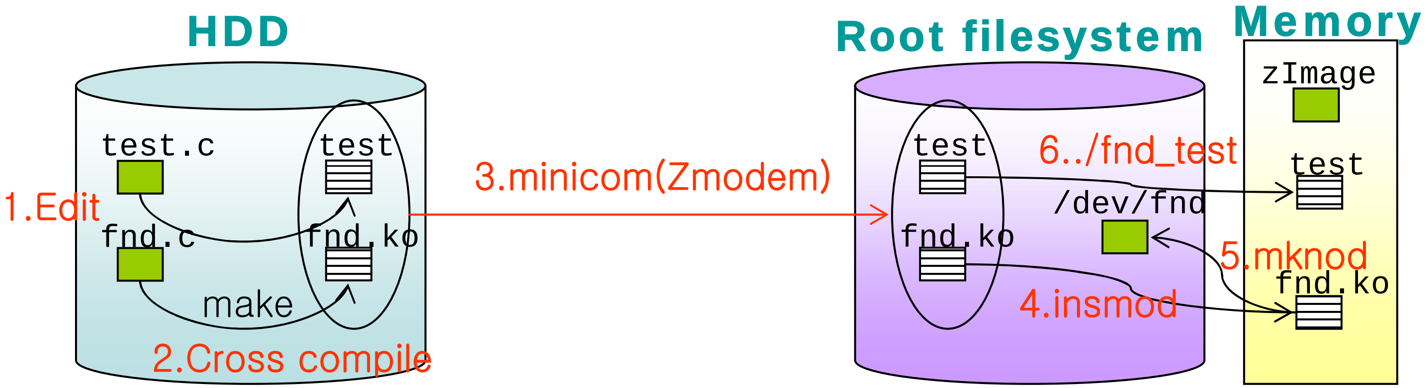
FND Device Driver

- **fnd_ioctl**
  - » command process, but do nothing in this case
- **module_init : fnd_init**
  - » register_chrdev
  - » memory mapping : ioremap
- **module_exit : fnd_exit**
  - » unregister_chrdev
  - » memory unmapping : iounmap

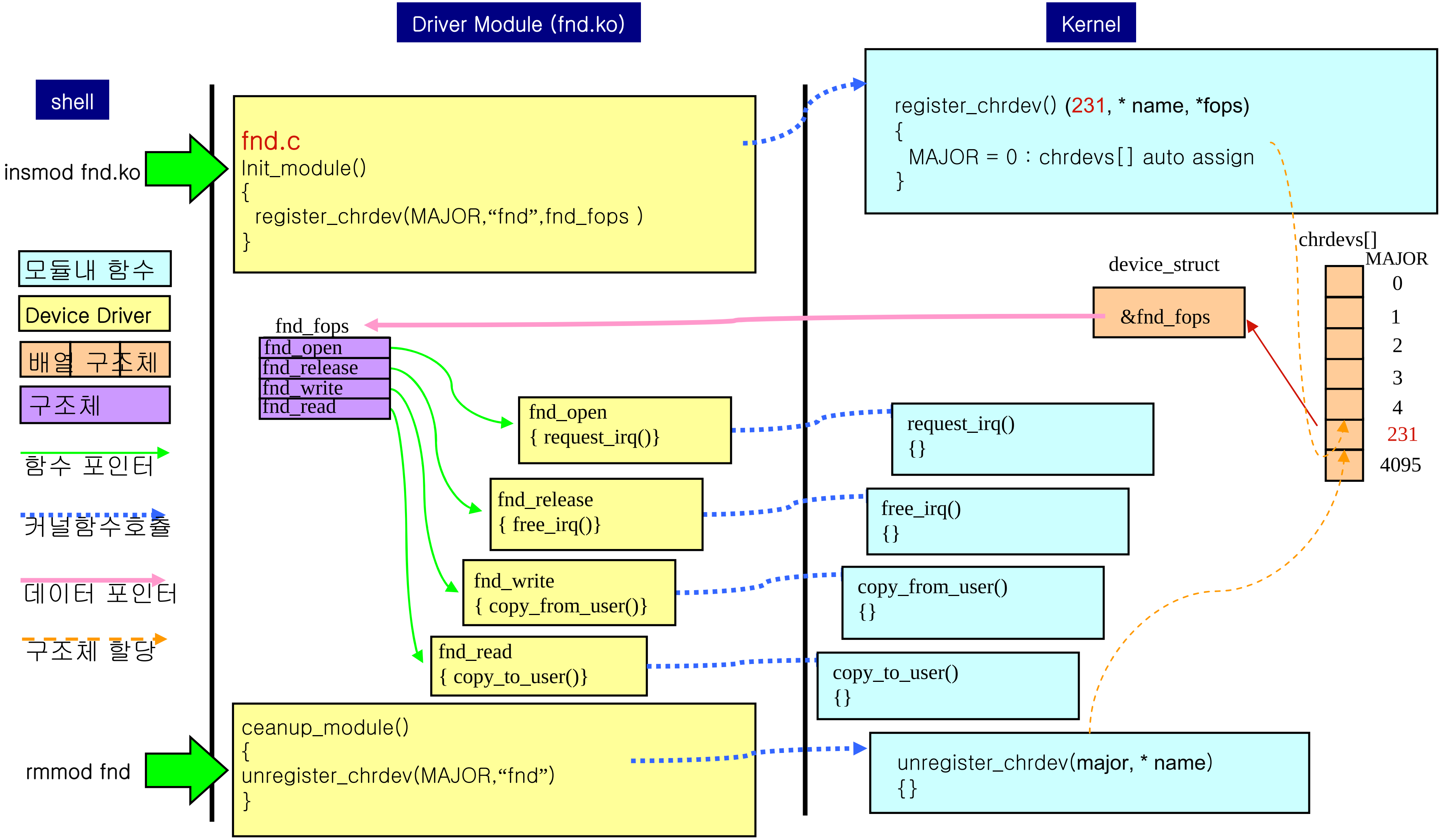
FND Test Program

- **fnd-test.c**

```
main(int ac, char *av[])
{
        int n, count, dev;
        unsigned char                   buf[MAXFND+1];

        dev = open( fnd_dev, O_RDWR);
        if (dev < 0) {
                fprintf(stderr, "cannot open FND (%d)", dev);
                exit(2);
        }
        memset(buf, 0, sizeof(buf));

        for (n = 0 ; n <= 9; n++)
        {
                for( count = 0 ; count < MAXFND; count++){
                buf[count]= asc_to_fnd(n);
                }
                write(dev, buf,MAXFND);
                usleep(500000);
        }
}
```



Module Install/Remove



Driver Module (fnd.ko)

Kernel

shell

insmod fnd.ko

```
fnd.c
Init_module()
{
  register_chrdev(MAJOR,“fnd”,fnd_fops )
}
```

```
register_chrdev() (231, * name, *fops)
{
  MAJOR = 0 : chrdevs[] auto assign
}
```

```
fnd_open
{ request_irq()}
```

```
request_irq()
{}
```

```
fnd_release
{ free_irq()}
```

```
free_irq()
{}
```

```
fnd_write
{ copy_from_user()}
```

```
copy_from_user()
{}
```

```
fnd_read
{ copy_to_user()}
```

```
copy_to_user()
{}
```

rmmod fnd

```
ceanup_module()
{
unregister_chrdev(MAJOR,“fnd”)
}
```

```
unregister_chrdev(major, * name)
{}
```

- 모듈내 함수
- Device Driver
- 배열 구조체
- 구조체
- 함수 포인터
- 커널함수호출
- 데이터 포인터
- 구조체 할당

Test run

- **Compile module** (o: 2.4, ko: 2.6) : make
- **Download to target** : minicom / nfs
- **Install the module**

  #insmod fnd.ko : /etc/rc.sysinit

  - » **Adding a new driver at module initialization**
    - int register_chrdev(unsigned int major, const char *name, struct file_operations *fops);
  - » **Remove module**
    - int unregister_chrdev(unsigned int major, const char *name);
- **List the module**

  #lsmod
- **If you let the system pick Major number, you can find the major number (for special creation) by**

  #more /proc/devices
- **Make a special file**

  #mknod /dev/fnd c 231 0

- **Application program**

  #./fnd-test

- **Target booting :**

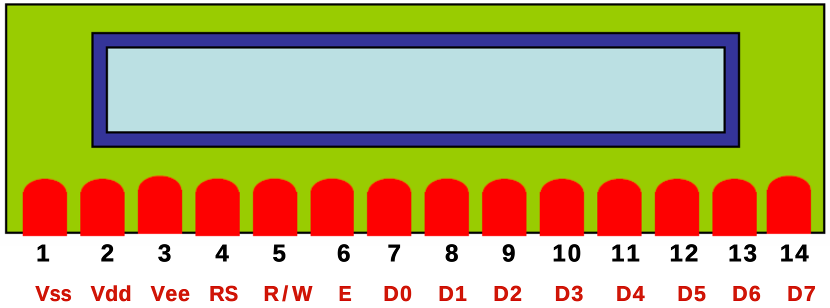
  - » /etc/rc.sysinit : module load

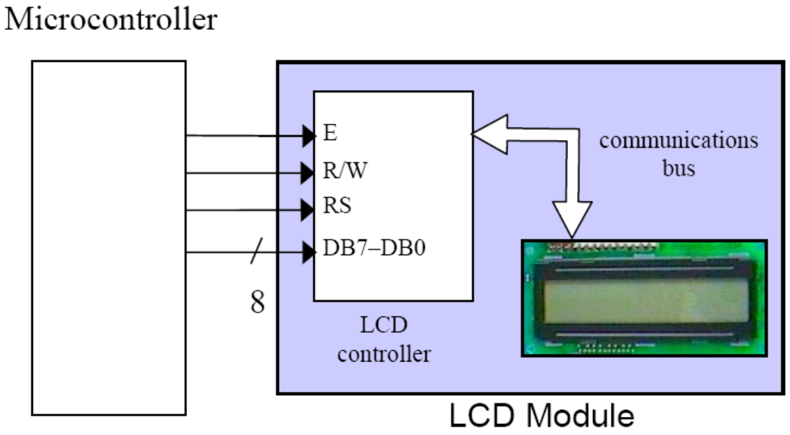
```
# ieb module
insmod /data/clcd.ko
insmod /data/dc_motor.ko
insmod /data/dot.ko
insmod /data/fnd.ko
insmod /data/key.ko
insmod /data/led.ko
insmod /data/st_motor.ko
```

- **http://lxr.linux.no**

LCD

LCD Interface





1. Direct Control or Address decoder Control
2. 4bit or 8bit Data Length

Pin No	Name	Function	Description
1	Vss	Power	GND
2	Vdd	Power	+ 5 V
3	Vee	Contrast Adj.	0 - 5 V
4	RS	Command	Register Select 0 control 1 data
5	R/W	Command	Read / Write 0 write 1 read
6	E	Command	Enable (Strobe)
7	D0	I/O	Data LSB
8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data MSB

LCD

Instructions

Instruction	RS	R W	D7	D6	D5	D4	D3	D2	D1	D0	Description	time
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0
Clear Display	0	0	0	0	0	0	0	0	0	1	Clear display & set address counter to zero	1.52ms
Cursor Home	0	0	0	0	0	0	0	0	1	x	Set adress counter to zero, return shifted display to original position. DD RAM contents remains unchanged.	1.52ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direction (I/D) and specify automatic display shift (S).	37us
Display Control	0	0	0	0	0	0	1	D	C	B	Turn display (D), cursor on/off (C), and cursor blinking (B).	37us
Cursor / Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Shift display or move cursor (S/C) and specify direction (R/L).	37us
Function Set	0	0	0	0	1	DL	N	F	x	x	Set interface data width (DL), number of display lines (N) and character font (F).	37us
Set CGRAM Address	0	0	0	1	CGRAM Address						Set CGRAM address. CGRAM data is sent afterwards.	37us
Set DDRAM Address	0	0	1	DDRAM Address							Set DDRAM address. DDRAM data is sent afterwards.	37us
Busy Flag & Address	0	1	BF	Address Counter							Read busy flag (BF) and address counter	0
Write Data	1	0	Data								Write data into DDRAM or CGRAM	37us
Read Data	1	1	Data								Read data from DDRAM or CGRAM	37us

LCD

Bit Settings

Bit name	Settings	
I/D	0 = Decrement cursor position	1 = Increment cursor position
S	0 = No display shift	1 = Display shift
D	0 = Display off	1 = Display on
C	0 = Cursor off	1 = Cursor on
B	0 = Cursor blink off	1 = Cursor blink on
S/C	0 = Move cursor	1 = Shift display
R/L	0 = Shift left	1 = Shift right
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = 1 line	1 = 2 lines
F	0 = 5x7 dots	1 = 5x10 dots
BF	0 = Can accept instruction	1 = Internal operation in progress
x	Don’t care	DDRAM : Display Data RAM CGRAM : Character Generator RAM

LCD

- **DDRAM**

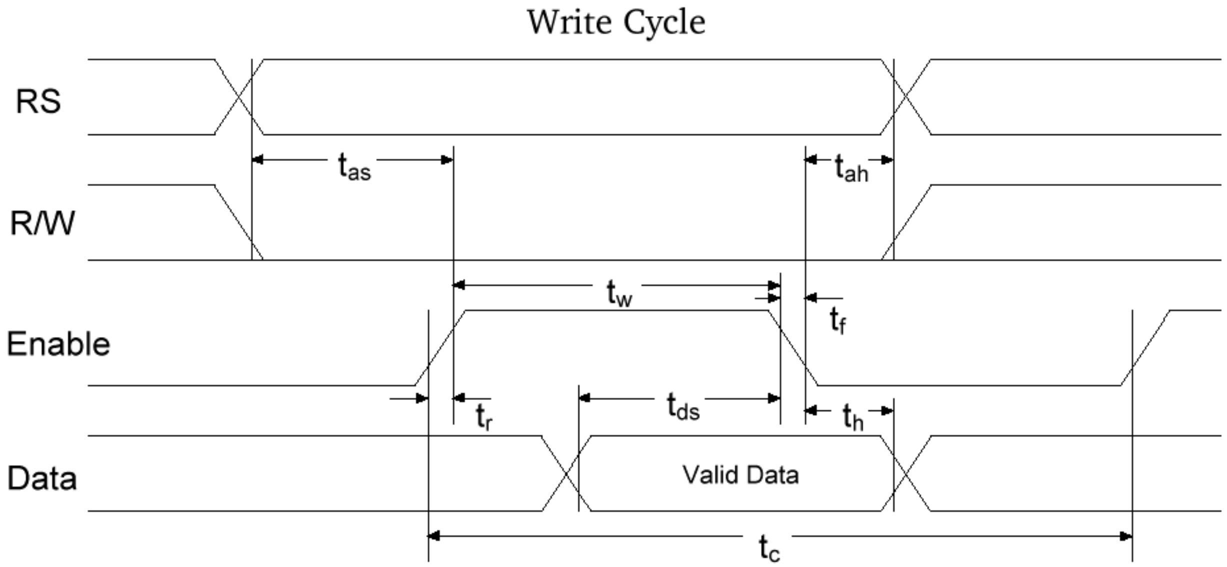
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

1. Character position (dec)
2. Row0(Line 1) DDRAM address (hex)
3. Row1(Line 2) DDRAM address (hex)

DDRAM address usage for a 2-line LCD		
Display size	Visible	
	Character positions	DDRAM addresses
2 x 16	00..15	00h..0Fh + 40h..4Fh
2 x 20	00..19	00h..13h + 40h..53h
2 x 24	00..23	00h..17h + 40h..57h
2 x 32	00..31	00h..1Fh + 40h..5Fh
2 x 40	00..39	00h..27h + 40h..67h

LCD

- ## Time cycle



Write-Cycle	V_{DD}	2.7-4.5V	4.5-5.5V	2.7-4.5V	4.5-5.5V	
Parameter	Symbol	Min		Max		Unit
Enable Cycle Time	t_c	**1000**	**500**	**-**	**-**	**ns**
Enable Pulse Width (High)	t_w	**450**	**230**	**-**	**-**	**ns**
Enable Rise/Fall Time	t_r, t_f	**-**	**-**	**25**	**20**	**ns**
Address Setup Time	t_{as}	**60**	**40**	**-**	**-**	**ns**
Address Hold Time	t_{ah}	**20**	**10**	**-**	**-**	**ns**
Data Setup Time	t_{ds}	**195**	**80**	**-**	**-**	**ns**
Data Hold Time	t_h	**10**	**10**	**-**	**-**	**ns**

Device Driver

- **LCD device driver : clcd.c**
  - » **8 Bit Write [D0~D7]**
  - » **Control Address = 0x12300000, Data Address = 0x12380000**
- **clcd_write**
  - » **copy_from_user(disp, buf, count);** buf(user), disp(kernel)
  - » **string_out(data);**

```
static void string_out(char *str)
{
            char *s;
            int i=0;

            printk("clcd.o: %s\n", str);

            lcd_init();

            for (s=str; *s; s++){
                        *((volatile unsigned char *)(mem_base_rs)) = *s; //            CLCD_DATA_ADDR = *s;

                        if(i == 15) {
                                    udelay(100);
                                    *((volatile unsigned char *)(mem_base_wr)) = 0xC0;
                        }

                        udelay(100);
                        i++;
            }
}
```

Device Driver

- clcd_open
  - » return 0
- clcd_release
  - » return 0 : release file object
- clcd_ioctl
  - » command process

```
static int clcd_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)
{
                switch(cmd)
                {
                                case 1:
                                                {
                                                                string_out("ioctl() cmd 1 test ");
                                                                break;
                                                }
                                default:
                                                return 0;
                }
                return 0;
}
```

Device Driver

- **module_init : clcd_init**
  - » register_chrdev
  - » memory mapping : ioremap
- **module_exit : clcd_exit**
  - » unregister_chrdev
  - » memory unmapping : iounmap

Device Driver

Clcd-test.c Test program

```
/* LCD test program */
#include    <stdio.h>
#include    <fcntl.h>

static char lcdDev[] = "/dev/clcd";
static int  lcdFd = (-1);

#define     MAXCHR  32

main(int ac, char *av[])
{
    int n;
    char        buf[MAXCHR];

    lcdFd = open( lcdDev, O_RDWR);
    if (lcdFd < 0) {
        fprintf(stderr, "cannot open LCD (%d)", lcdFd);
        exit(2);
    }

//  ioctl(lcdFd, 1, 0);
// sleep(1);

    memset(buf, 0, sizeof(buf));
    if (ac > 1) {
        n = strlen(av[1]);              // av[0] = clcd-test

        if (n > MAXCHR)
               n = MAXCHR; //plus the newline
    memcpy(buf, av[1], n);
    }
   write(lcdFd, buf, MAXCHR);
}
```