Step Motor Device Driver

경희대학교 컴퓨터공학과

조 진 성

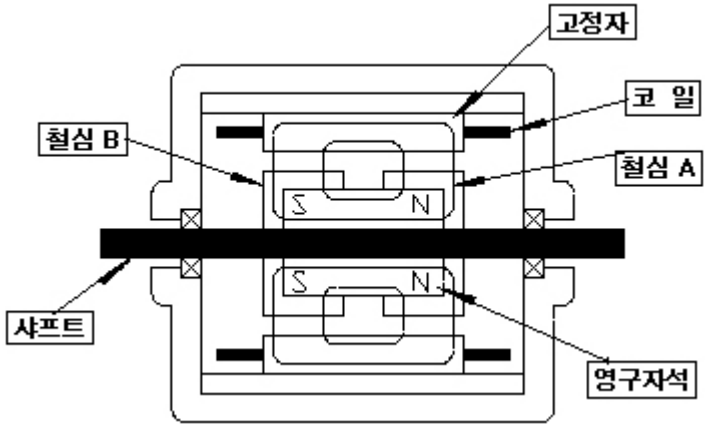
Step Motor의 개요

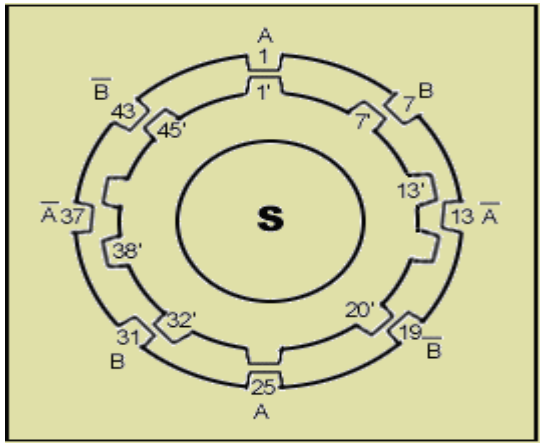
- Step Motor
  - ✓ Step Motor는 디지털 펄스를 기계적인 축 운동으로 변환시키는 변환기
  - ✓ 펄스는 디지털 source에 의해 가해짐
  - ✓ 정확한 각도제어에 유리하여 현재 주위에 많이 사용

Step Motor의 개요 (2)

- 장점
  - ✓ 디지털 신호로 직접 open loop 제어를 할 수 있고, 전반적으로 간단하게 구동
  - ✓ 회전 속도 : 펄스 신호의 주파수에 비례
  - ✓ 모터 회전각 : 입력 Pulse 수에 비례, 모터의 속도가 1초간의 입력 Pulse수에 비례
  - ✓ 기동, 정지, 정-역 회전, 변속이 용이하며 응답 특성도 양호
  - ✓ 1스텝 당 오차가 +5% 이내, 회전각의 오차는 step마다 누적
- 단점
  - ✓ 고속운전 시 탈조가 쉬움
  - ✓ 특정 주파수에서는 진공, 공진 현상이 발생하기 쉽고, 관성이 있는 부하에 약함

Step Motor의 구조

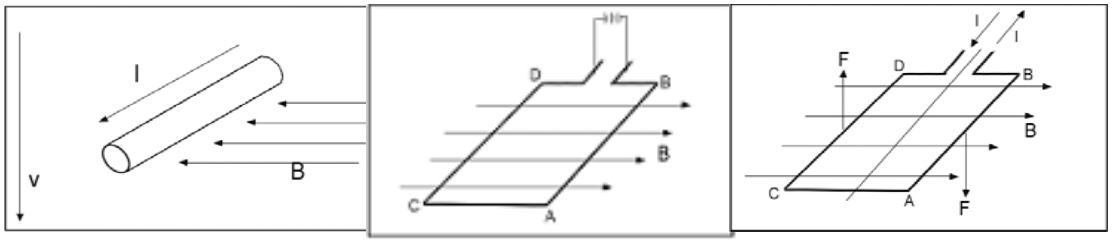




- Step Motor는 내부의 고정자의 극의 수에 따라 단상(1상), 2상~6상 등의 종류가 있음
- 상이랑 위상을 말하며 모터 내부에 몇 조의 코일을 가지는가를 뜻함

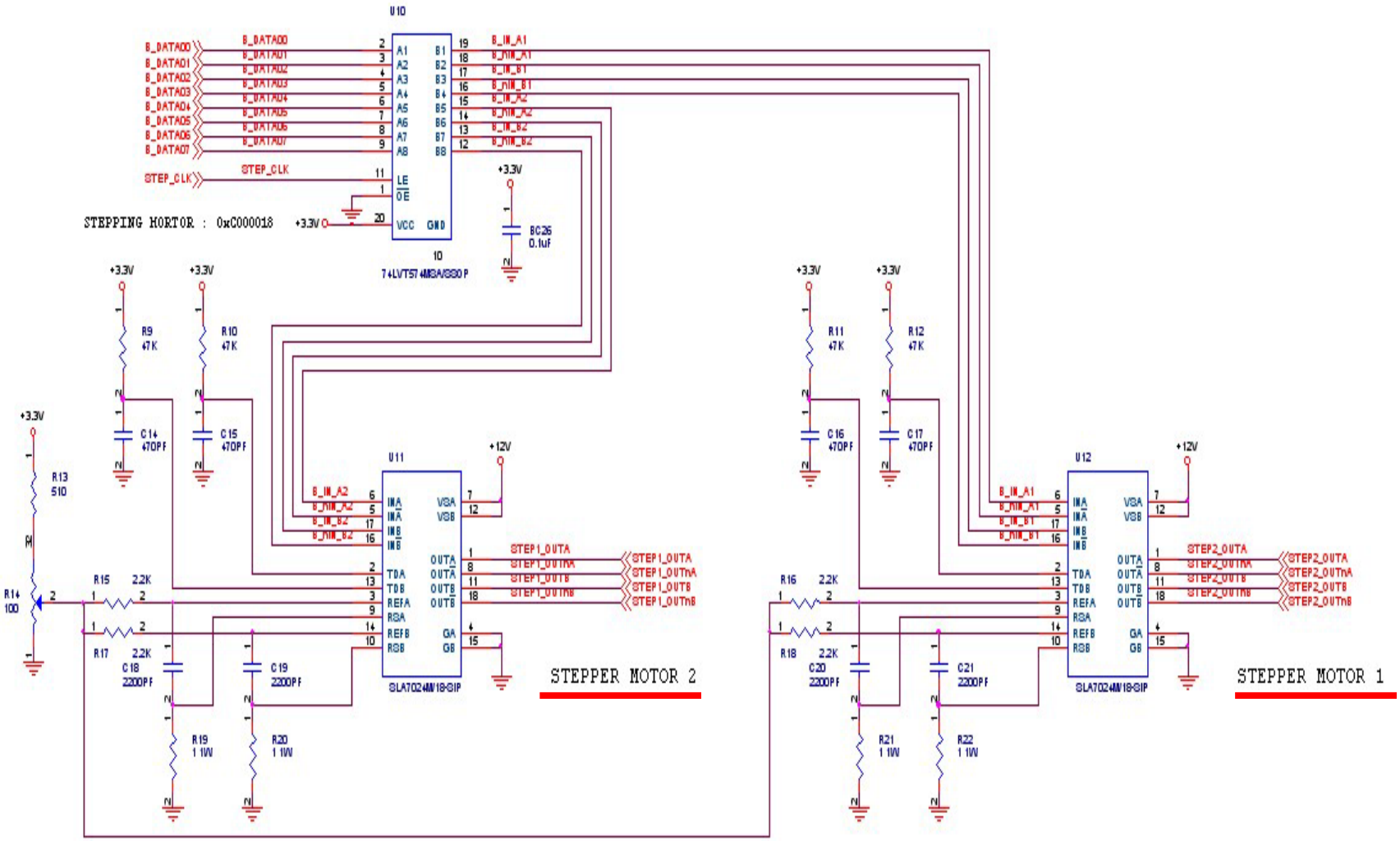
STEP MOTOR 구동원리

- Step Motor 구동 기본 법칙 들



- 고정자에 감긴 도선에 전류를 흐르게 하면, 회전자의 자계와 작용하여 회전자가 움직임
- 일단 회전자가 움직이면 이는 고정된 자계 속을 도선이 움직이는 것과 같은 효과가 발생하여 동일 도선에 회전속도에 비례하여 반대방향의 역기전압이 발생
- 따라서 외부에서 흘려주는 입력전류와 반대방향의 역기전압은 서로 상쇄되어 실제 도선 속을 흐르는 전류를 제한, 이로써 모터의 회전속도와 이 속도에서의 토르크를 제한, 즉 최대속도에서의 토르크는 0이 된다

PXA255-Pro Step Motor 회로도



Step Motor Data 입력 bit 설명

- 8bit로 두 개의 모터를 제어가능
- 현재 PXA255보드의 FPGA에는 1개의 모터를 사용

Bit	7	6	5	4	3	2	1	0
Step Motor	nB	B	nA	step2 A	nB	B	nA	step1 A

모터2 제어bit　　모터1 제어bit

제어순서

- Half Step Operation
  - ✓ 1상-2상-1상-2상 과 같이 교대적으로 펄스를 부여하는 방식이다
  - ✓ 이때의 스텝당 회전각도는 기본각의 1/2이며, 따라서 모터의 진동이 줄어들고 보다 고속운전이 가능하나 부하가 큰 경우 정지위치의 정밀도에 주의를 요한다

Sequence	Input				Output
	A	nA	B	nB	
1	1(H)	0(L)	0(L)	0(L)	A
2	0	0	1	0	B
3	0	1	0	0	nA
4	0	0	0	1	nB

제어순서 (2)

- Full Step Operation
  - ✓ 항시 4상중 2개의 상이 함께 입력전원을 받아들인 구동방식으로 일반적으로 많이 사용되는 방식이다. 출력 토르크가 좋다
  - ✓ 토르크(Torque)
    - 로터의 회전축 중심으로부터 1Cm 또는 1m 떨어진 부하(물체)를 단위시 (1초)에 1Cm 또는 1m를 움직일 수 있는 힘(단위 : gfcm 또는 Kgfcm 등)

Sequence	Input				Output
	A	nA	B	nB	
1	1(H)	0(L)	1(H)	0(L)	AB
2	0	1	1	0	nAB
3	0	1	0	1	nAnB
4	1	0	0	1	AnB

Device driver source code

```
#include <linux/module.h>      //모듈에 관한 정의 ex)MOD_INC_USR_COUNT
#include <asm/hardware.h>
#include <asm/uaccess.h>       //copy_from_user()
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <asm/ioctl.h>
#include <linux/ioport.h>
#include <linux/delay.h>       //udelay()
#include <asm/io.h>                    //GPIO controller에 관한 정의
#include <linux/init.h>                //init_module()과 cleanup_module()
#include <linux/version.h>

#define IOM_STEP_MAJOR 247             //define major number
#define IOM_STEP_NAME "STEP"           //define device name
#define IOM_STEP_ADDRESS 0xC00000C
```

Device driver source code (2)

```
#define A          (unsigned short)(0x1)
#define AB         (unsigned short)(0x5)
#define B          (unsigned short)(0x4)
#define _AB        (unsigned short)(0x6)
#define _A         (unsigned short)(0x2)
#define _A_B       (unsigned short)(0xa)
#define _B         (unsigned short)(0x8)
#define A_B        (unsigned short)(0x9)

#define iom_step_init        init_module

int iom_step_open(struct inode *, struct file *);
int iom_step_release(struct inode *, struct file *);
ssize_t iom_step_write(struct file *, const char *, size_t, loff_t *);
int __init iom_step_init(void);
void cleanup_module(void);
```

Device driver source code (3)

```
//Global variable
static int step_usage = 0;
static int step_major = 0;
static char mode;
static int check = 0;

static unsigned short *iom_step_addr;

//file operation structure
static struct file_operations iom_step_fops =
{
	open:		iom_step_open,
	write:		iom_step_write,
	release:	iom_step_release,
};
```

Device driver source code (4)

```
into iom_step_open(struct inode *minode, struct file *mfile)
{
        if(step_usage != 0) return -EBUSY;

        MOD_INC_USE_COUNT;
        step_usage = 1;
        check = 0;
        return 0;
}

int iom_step_release(struct inode *minode, struct file *mfile)
{
        MOD_DEC_USE_COUNT;
        step_usage = 0;
        outw(0,iom_step_addr);
        return 0;
}
```

Device driver source code (5)

```
size iom_step_write(struct file *inode, const char *gdata, size_t length,
loff_t *off_what)
{
        if(check == 0)
        {//모드를 입력받았는지 check
                const char *temp = gdata;
                copy_from_user(&mode, temp, 1);
                check = 1;
        }
        else if(check ==1)
        {//모드를 입력받았다면 Step Motor동작
                const char *tmp = gdata;
                unsigned short speed;

                copy_from_user(&speed, tmp , 2);
#if 0
                //Full Step Operation
                outw(AB,iom_step_addr);
                udelay(speed);
```

Device driver source code (6)

```
        outw(_AB,iom_step_addr);
        udelay(speed);
        outw(_A_B,iom_step_addr);
        udelay(speed);
        outw(A_B,iom_step_addr);
        udelay(speed);
#else
        //Half Step Operation
        if(mode == 'a'){//정방향 모드
                outw(A,iom_step_addr);
                udelay(speed);
                outw(B,iom_step_addr);
                udelay(speed);
                outw(_A,iom_step_addr);
                udelay(speed);
                outw(_B,iom_step_addr);
                udelay(speed);
        }
```

Device driver source code (7)

```
        else if(mode == 'b'){//역방향 모드
                outw(_B,iom_step_addr);
                udelay(speed);
                outw(_A,iom_step_addr);
                udelay(speed);
                outw(B,iom_step_addr);
                udelay(speed);
                outw(A,iom_step_addr);
                udelay(speed);
        }
#endif
    }
    return length;
}
```

Device driver source code (8)

```
into __init iom_step_init(void)
{
        int result;
        result = register_chrdev(IOM_STEP_MAJOR,IOM_STEP_NAME,&iom_step_fops);
        if(result < 0) {
                printk(KERN_WARNING"Can't get any major\n");
                return result;
        }
        step_major = IOM_STEP_MAJOR;
        iom_step_addr = ioremap(IOM_STEP_ADDRESS,0x02);
        printk("init module, %s major number : %d\n",IOM_STEP_NAME,step_major);
        return 0;
}
void cleanup_module(void)
{
        iounmap(iom_step_addr);
        if(unregister_chrdev(step_major,IOM_STEP_NAME))
                printk(KERN_WARNING"%s DRIVER CLEANUP FALLED\n",IOM_STEP_NAME);
}
```

Application source code

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

static char mode;
into main(void)
{
	int fd;
	unsigned short c = 0xffff;
	printf("a. spin\n");//menu
	printf("b. back spin\n");
	printf("select->");
	scanf("%c", &mode);
	printf("\n");
```

Application source code (2)

```
        while(1){
            fd = open("/dev/STEP",O_WRONLY);//divice open
            if (fd < 0){
                printf("Device Open Error\n");
                exit(1);}
            write(fd, &mode, 1);//write mode
            for(;;){
                if(c < 0x1000){
                        c = 0x1000;//일정속도가 되면 유지
                }
                else c -=0x100;//increase speed
                write(fd,&c,2);//write speed
            }
            close(fd);
        }
    }
```

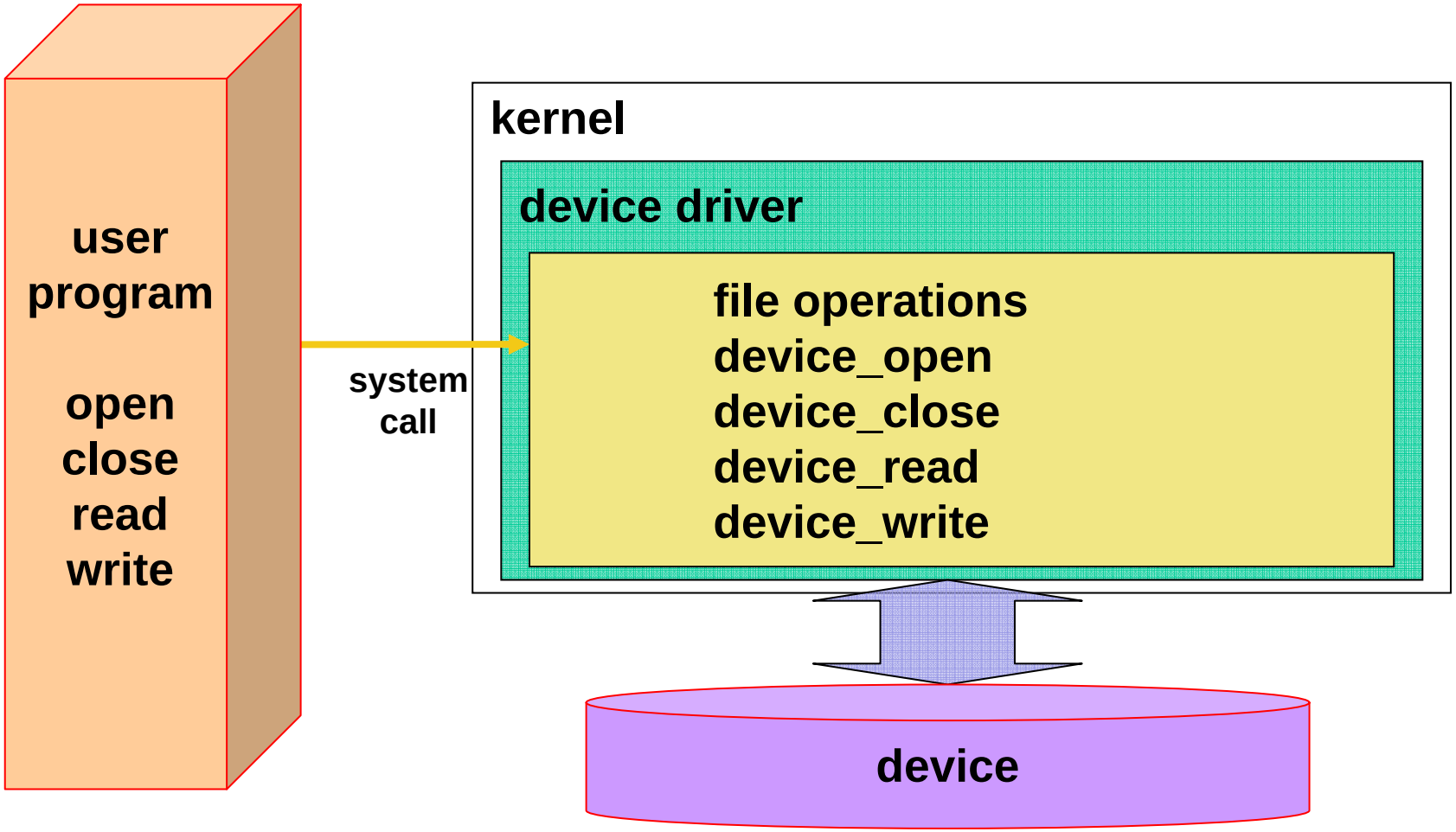
Makefile

```
INCLUDEDIR := /home/max233/linux-2.4.19-cd/include

CFLAGS := -D__KERNEL__ -I$(INCLUDEDIR) -Wall -O2 -DMODULE

CROSS_COMPILE := arm-linux-

CC=$(CROSS_COMPILE)gcc
LD=$(CROSS_COMPILE)ld

all: step_driver test_step
step_driver:
	$(CC) $(CFLAGS) -c step_driver.c -o step_driver.o
test_step:
	$(CC) -I$(INCLUDEDIR) -o test_step test_step.c
clean:
	rm -f *.o
	rm -f test_step
```

Driver를 커널에 모듈로 등록 및 해제하기

- **Step Motor Device Driver 등록 방법**
  - ✓ 외부와 device driver는 file interface (node를 의미)를 통해 연결
  - ✓ Device driver는 자신을 구별하기 위해 고유의 major number를 사용
- **장치의 등록과 해제**
  - ✓ 등록 : int register_chrdev(unsigned int major, const char *name, struct file_operations *fops)
    - Major : 등록할 major number. 0이면 사용하지 않는 번호 중 자동으로 할당
    - Name : device의 이름
    - Fops : device에 대한 file 연산 함수들
  - ✓ 해제 : int unregister_chrdev(unsigned int major, const char *name)

Driver를 커널에 모듈로 등록 및 해제하기 (2)



Driver를 커널에 모듈로 등록 및 해제하기 (3)

- **Major number와 minor number**
  - ✓ 장치를 구분하는 방법으로 둘을 이용하여 특정 장치를 구별
  - ✓ Major number : 커널에서 디바이스 드라이버를 구분하는데 사용
  - ✓ Minor number : 디바이스 드라이버 내에서 필요한 경우 장치를 구분하기 위해 사용
  - ✓ 새로운 디바이스는 새로운 major number를 가져야 함
  - ✓ register_chrdev()로 장치를 등록할 때 major number를 지정
  - ✓ 같은 major number가 등록되어 있으면 등록 실패
  - ✓ Major와 minor 번호는 파일의 정보를 담고 있는 inode의 i_rdev에 16bit로 저장된다. 상위 8bit는 major, 하위 8bit는 minor이다.

Driver를 커널에 모듈로 등록 및 해제하기 (4)

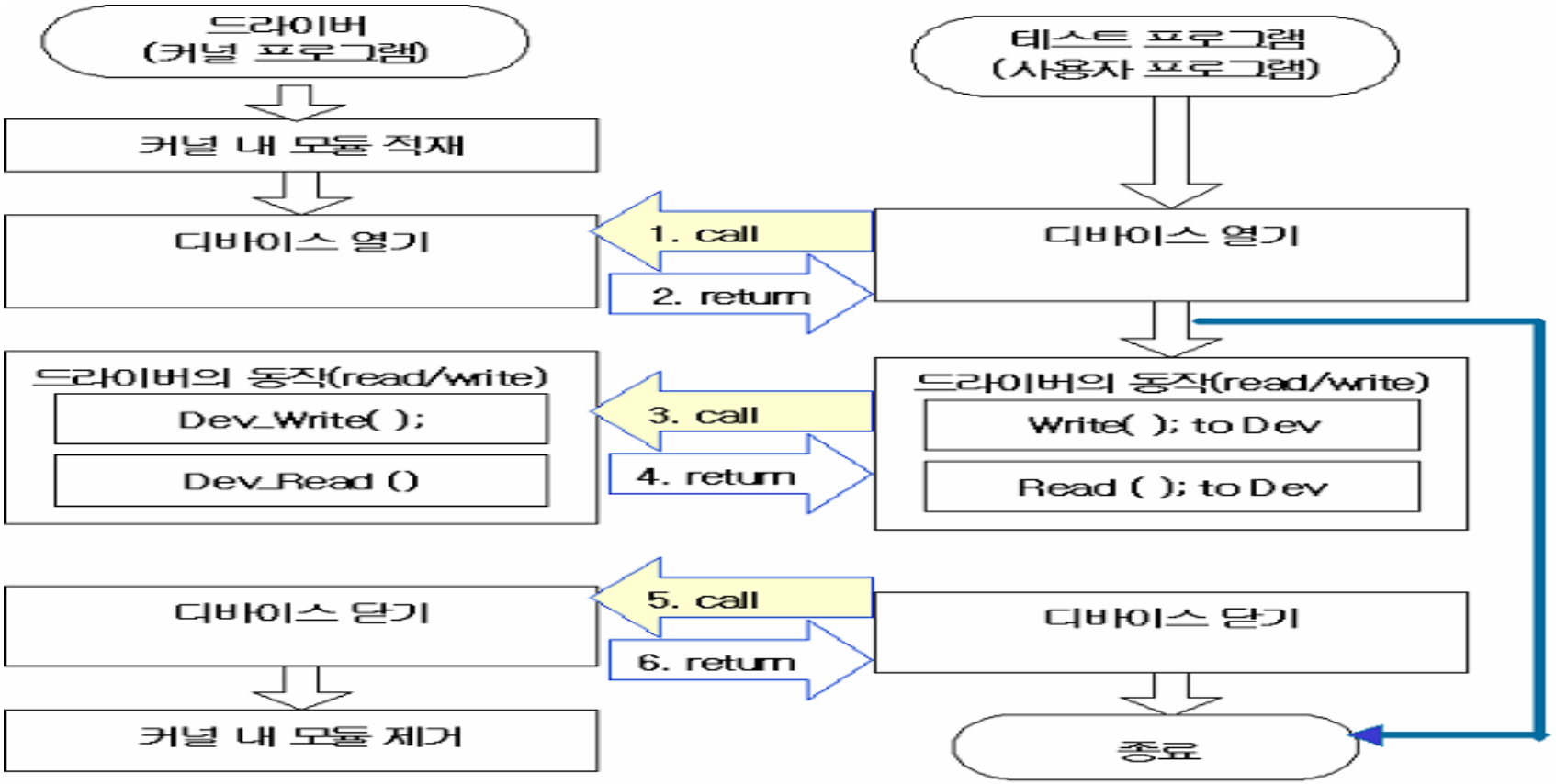
- **mknod 명령으로 디바이스 드라이버에 접근할 수 있는 장치 파일 생성**
  - ✓ mknod [device file name] [type] [major] [minor]
    - Ex] %mknod /dev/STEP c 247 0
- **mdev_t : 장치의 major, minor number를 표현하는 자료구조**
  - ✓ MAJOR() : kdev_t에서 major number를 얻어내는 매크로
    - Ex] MAJOR(inode->i_rdev);
  - ✓ MINOR() : kdev_t에서 minor number를 얻어내는 매크로
  - ✓ cat /proc/devices 명령으로 현재 로드된 디바이스 드라이버 확인

Driver를 커널에 모듈로 등록 및 해제하기 (5)

이 름	용 도
insmod	module을 설치(install)
rmmod	실행중인 modules을 제거(unload)
lsmod	Load된 module들의 정보를 표시
depmod	Module들의 symbol들을 이용하여 Makefile과 유사한 dependency file을 생성
modprobe	depmod명령으로 생성된 dependency를 이용하여 지정된 디렉토리의 module들과 연관된 module들을 자동으로 load
modinfo	목적화일을 검사해서 관련된 정보를 표시

Driver를 커널에 모듈로 등록 및 해제하기 (6)

- **Device Driver의 동작 과정**



Driver를 커널에 모듈로 등록 및 해제하기 (7)

- 앞에서 만든 드라이버 소스를 step_driver.c, test application은 test_step.c, 그리고 makefile은 Makefile로 작성 후 저장한다
- make 명령어를 이용하여 위의 2 files을 컴파일한다

  % make
- 생성된 step_driver.o 와 test_step를 target으로 전송한다. 전송방식은 앞에서 배웠던 minicom 또는 nfs방식을 이용한다. 밑에서는 nfs 방식을 통하여 파일을 전송하였다

Driver를 커널에 모듈로 등록 및 해제하기 (8)

```
root@localhost:~/test2
파일(F)  편집(E)  보기(V)  터미널(T)  가기(G)  도움말(H)
# insmod step_driver.o
Using step_driver.o
init module, STEP major number : 247
# mknod /dev/STEP c 247 0
# ./test_step
a. spin
b. back spin
select->a
```

- 위와 같이 실행 후 Step Motor의 동작을 확인한다