
mmap을 이용한 디바이스 제어

- Chapter 10 -

Contents

I. mmap()

II. Achro5250 확장 보드 디바이스

III. 디바이스 제어 실습

IV. 디바이스 제어 실습 - LED 제어

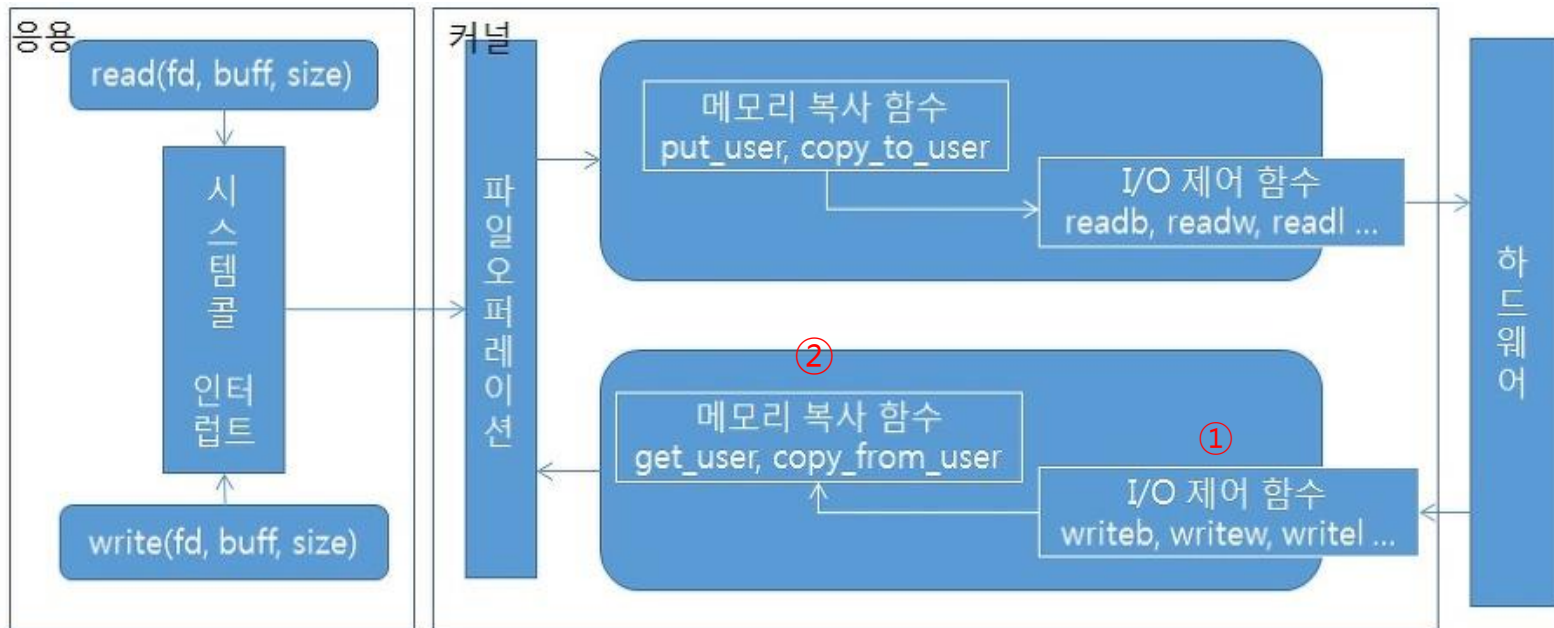
V. 디바이스 제어 실습 - FND 제어

mmap()

▶ mmap()의 배경 (Memory Map)

▶ 기존의 입출력 방식의 비효율성

- ▶ 한 프로세서가 많은 양의 데이터(디스크) 입출력(read(), write(), ioctl())을 할 경우 기존의 드라이버를 사용하면 ①디스크로 부터 커널 내부 버퍼로 복사하고 난 후 ②동일한 자료를 사용자 프로세스에 포함된 자료구조에 다시 복사를 할 경우 overhead ➔ 성능의 저하
- ▶ 참고: 리눅스는 모든 디바이스를 파일로 추상화 (/dev의 밑에 있는 파일(노드)로 간주)



mmap()

▶ mmap()의 개념

▶ 디바이스의 메모리 일부를 현재 프로세스의 메모리 영역으로 매핑(memory mapping)하고 직접 호출함으로써 데이터(파일) 접근속도를 증가시킴

▶ mmap()

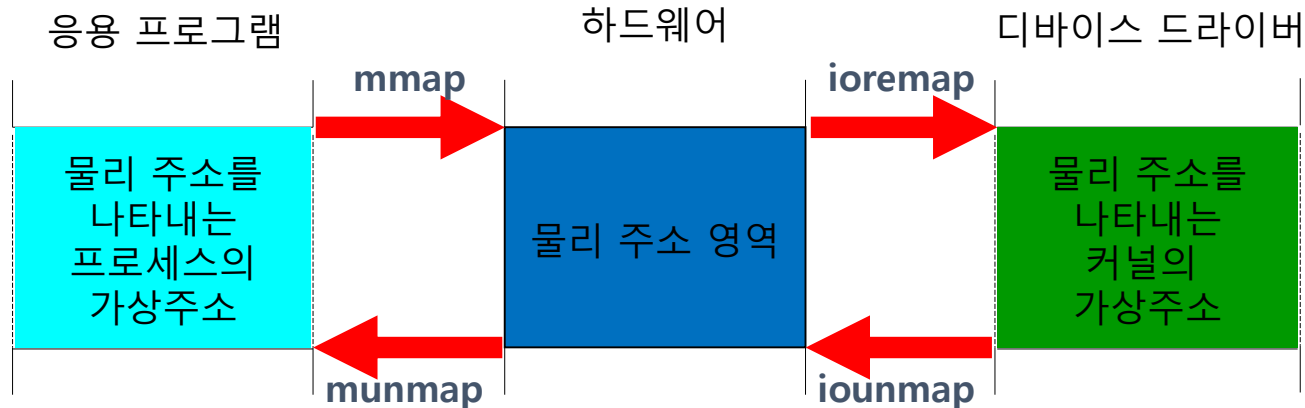
▶ 응용 프로그램 level에서 주소를 접근하고자 할 때 사용(응용프로그램의 가상 주소에 매핑)

▶ 사용자 레벨 ↔ 커널 레벨

▶ ioremap()

▶ mmap과 같은 역할을 하나 커널 내부에서 주소 접근시에 사용 (커널 레벨 ↔ 커널 레벨)

▶ device driver 작성시에는 mmap 대신 ioremap을 사용



mmap()

▶ mmap()의 용도

1) 프로세스간 데이터의 교환을 위한 용도

- ▶ 각각의 프로세스는 다른 프로세스와 중복되지 않는 각각의 주소공간(텍스트, 데이터, 스택)을 가지며 이는 기본적으로 다른 프로세스 메모리와 공유되지 않는데 메모리의 내용을 파일에 대응시킬 수 있다면 프로세스간 데이터의 교환을 위한 용도로 사용가능
 - 즉 프로세스간 공유하고자 하는 데이터를 파일에 대응시키고 이것을 쓰고 읽음

- ▶ mmap은 메모리의 특정영역(프로세스의 주소공간)을 파일(시스템 전역적인 객체)로 대응

2) 메모리의 내용을 파일에 직접 대응시켜서 성능을 향상

- ▶ 고전적인 방법은 파일지정자를 얻어서 직접 입출력 하는 방식으로 open, read, write, lseek와 같은 함수를 이용
- ▶ 이러한 함수의 사용은 당연하지만 번거로운 과정이 포함되는데, mmap를 이용하면 단순화
 - 메모리 복사 없이 하드웨어의 I/O 주소공간을 직접 사용
- ▶ 메모리의 페이지 크기의 n배로 구성해야 함
 - ▶ 4K(4096) 페이지에서 7바이트만 사용하면 나머지 4089바이트는 낭비됨

mmap()

▶ mmap() prototype

▶ `void *mmap(void *start, size_t length, int prot, int flags, int fd, off_t offset);`

▷ start: 프로세스의 주소공간에 매핑하기를 원하는 주소(NULL이면 빈공간에)

▷ length: 매핑 영역의 크기, PAGE_SIZE 단위

▷ prot : 파일에 대응되는 메모리 영역의 보호특성

- PRTO_EXEC(실행가능), PROT_READ(읽기가능), PROT_WRITE(쓰기가능), PROT_NONE(접근불가)

▷ flags:

- MAP_SHARE(해당 프로세스가 서로 공유)
- MAP_PRIVATE(하나의 프로세스만 접근 허용)

▷ offset:

- 매핑을 요청하는 물리주소의 시작주소 또는 디바이스 드라이버가 제공하는 주소의 offset값

▶ `int munmap(void *start, size_t length);`

- 매핑된 영역 해제

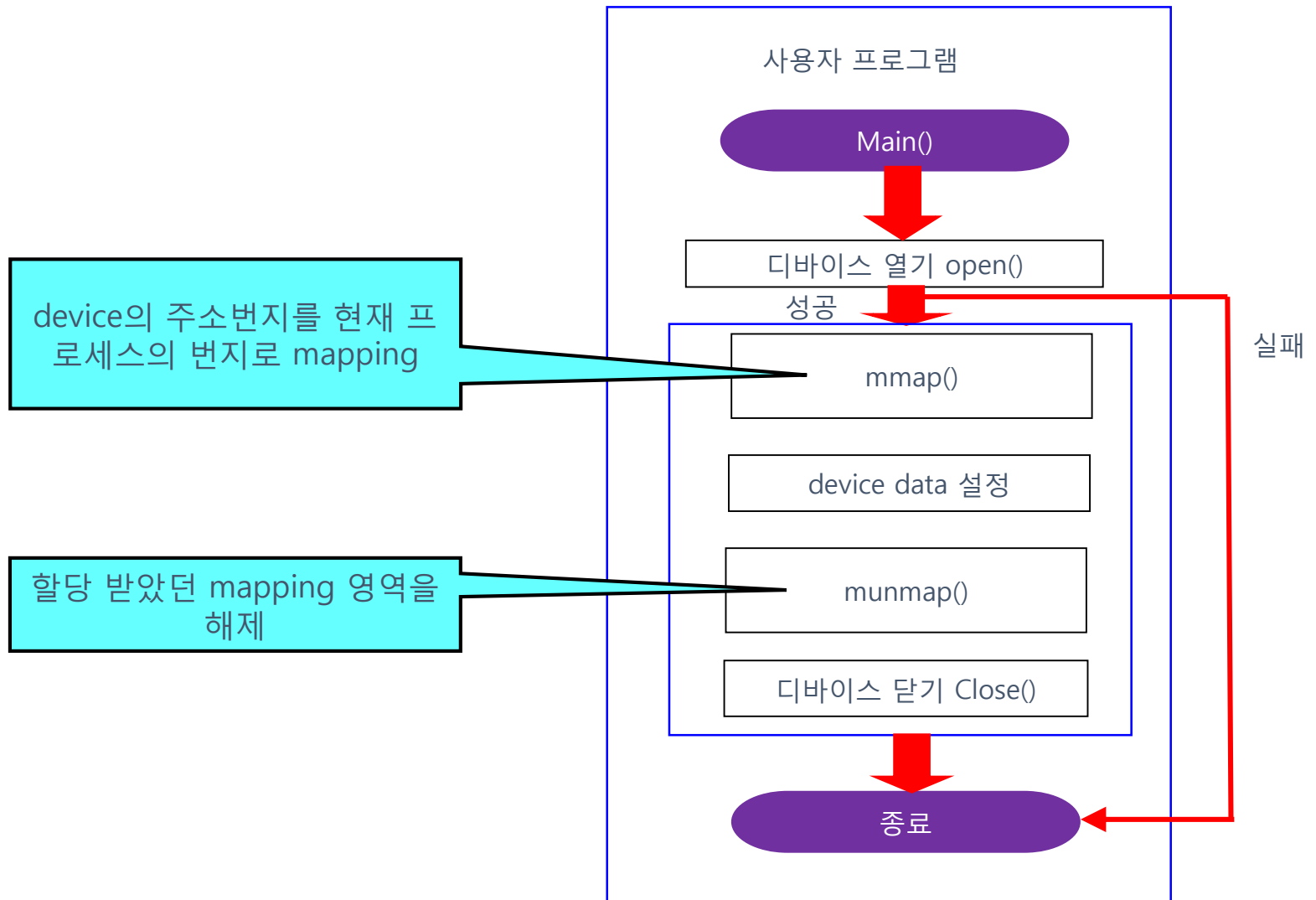
mmap()

▶ mmap() 사용 예제

```
int fd;  
  
char *addr;  
  
fd = open( "/dev/xxx", O_RDWR | O_SYNC );  
  
addr = (char *) mmap(0,                // 보통 0 사용  
                     4*4096,           // PAGE_SIZE(4096) 단위  
                     PROT_READ | PROT_WRITE,  
                     MAP_SHARED,  
                     fd,  
                     0x100000000 );    // 원하는 물리적 주소
```

mmap()

▶ mmap()을 이용한 디바이스 제어 과정



Achro5250 확장 보드 디바이스

▶ FPGA의 각 장치별 제어 어드레스 및 노드, 주변호 정보

- ▶ Achro-FPGA는 0x07000000에 매핑 되어있으므로, 베이스 어드레스로부터 해당 장치에 지정된 장치로 데이터가 전송

번호	장치	어드레스	node	Major
1	LED	0x0700_0016	/dev/fpga_led	260
2	Seven Segment (FND)	0x0700_0004	/dev/fpga_fnd	261
3	Dot Matrix	0x0700_0210	/dev/fpga_dot	262
4	Text LCD	0x0700_0100	/dev/fpga_text_lcd	263
5	Buzzer	0x0700_0020	/dev/fpga_buzzer	264
6	Push Switch	0x0700_0017	/dev/fpga_push_switch	265
7	3Dip Switch	0x0700_0000	/dev/fpga_dip_switch	266
8	Step Motor	0x0700_000C	/dev/fpga_step_motor	267
EN	Demo Register	0x0700_0300	N/A	N/A

디바이스 제어 실습

▶ 실습 종류

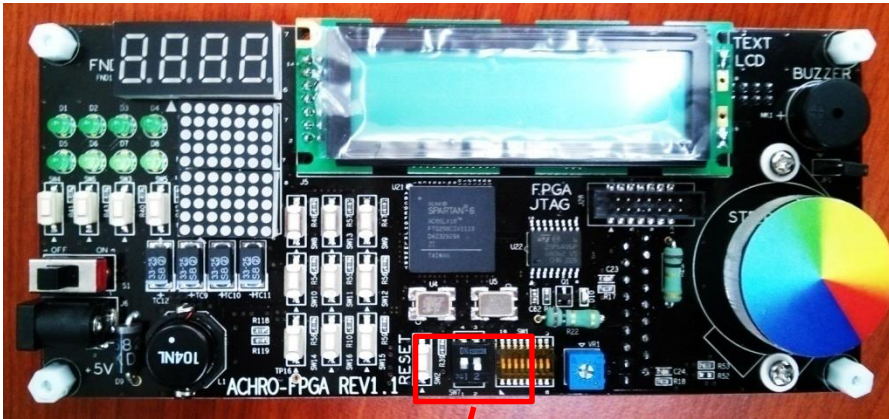
- 1) LED
- 2) Seven Segment (FND)

※ 각 장치의 디바이스 드라이버는 기존에 사용되던 것을 그대로 사용
특별히 `rmmod`를 실시하지 않았으면 이미 설치되어 있다

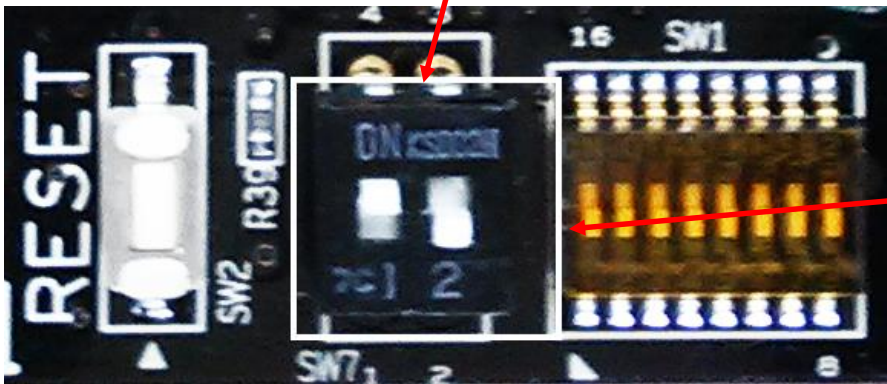
디바이스 제어 실습

▶ 보드 설정

▶ Achro-FPGA



▶ FPGA 사용을 위해서 FPGA의 점퍼 설정



SW7의 1번 ON, 2번 OFF

디바이스 제어 실습

▶ 실습 방법

1) 예제 가져오기 (학교 서버에서 ftp를 이용해 예제 및 실습환경을 /work/achro5250 폴더로 가져옴)

- ▶ # cd /work/achro5250 // 리눅스 호스트 작업 폴더
- ▶ # ftp computer.kpu.ac.kr // ftp를 이용
- ▶ Name (computer.kpu.ac.kr: root): anonymous
- ▶ Passwd: anonymous
- ▶ ftp> cd achro5250 // 서버 폴더 지정
- ▶ ftp> get mmap.tar.gz // 소스 및 환경 가져오기
- ▶ ftp> bye // tp 끝내기
- ▶ # tar xvf mmap.tar.gz // 리눅스 서버에서 압축 풀기
- ▶ # cd mmap

2) 실습용 소스 프로그램을 컴파일하고 타겟 보드로 옮길 준비 (모든 소스들을 동일한 방법)

- ▶ # arm-linux-gcc -o mmap_XXX mmap_XXX.c
 - arm-linux-gcc가 동작되지 않으면 심볼릭 링크 설정이 안된 것임
 - Arm-none-linux-gnueabi-gcc 명령을 사용할 것
- ▶ # cp mmap_XXX /nfsroot // nfs를 이용해 타겟으로 전송

디바이스 제어 실습

▶ 실습 방법 - 계속

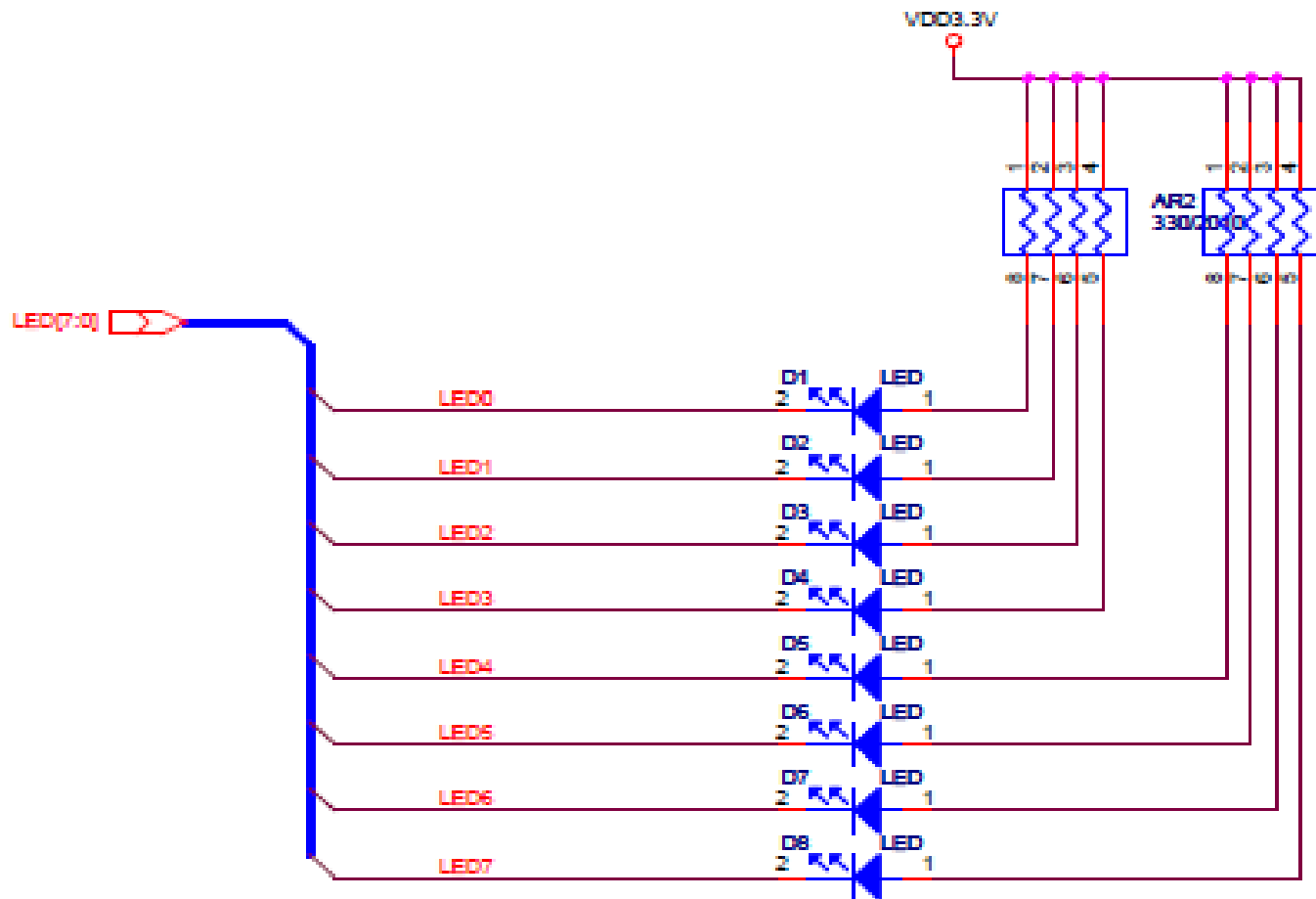
3) 타겟 보드에서 NFS 마운트 후 진행

- ▶ # mount -t nfs 10.40.1.51:/nfsroot /mnt/nfs -o tcp,nolock // 마운트
- ▶ # cp -a /mnt/nfs/mmap_XXX /root // 타겟의 루트폴더로 실행
파일 복사
- ▶ # cd /root
- ▶ # ./mmap_XXX // 실행

디바이스 제어 실습 - LED 제어

▶ FPGA LED Driver

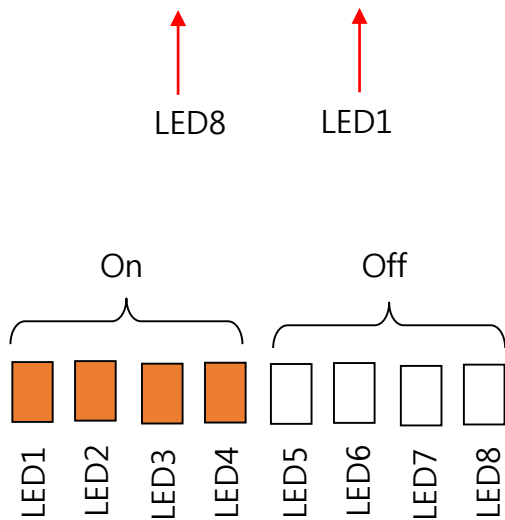
- ▶ VDD로부터 전원을 공급 받고 있기 때문에 LED I/O측 값에 따라서 LED가 점등



디바이스 제어 실습 - LED 제어

▶ LED 제어 예시

- ▶ LED1~8 중에 LED1~4를 'On' 시키기
- ▶ 비트 값이 0이면 ON이 되고, 1이면 OFF됨
- ▶ 0xf0 -> b'11110000' 이다.



디바이스 제어 실습 - LED 제어

▶ mmap을 이용한 LED 제어 소스

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <ctype.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <signal.h>
#define MAP_SIZE    0x1000                /* 페이지 크기(4096)의 정수배 */
#define MAP_PHYS    0x07000000
#define LED_OFFSET  0x16
#define LED (*(volatile unsigned short*)(led_addr + LED_OFFSET))
unsigned int val[] = { 0xff, 0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f };
void *led_addr;
int quit = 0;
void quit_signal(int sig)
{
    quit = 1;
}
```

LED1~8을 순차적으로 점멸하기
위한 값이다.

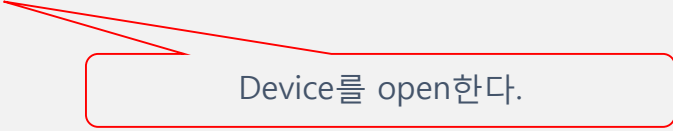
디바이스 제어 실습 - LED 제어

▶ mmap을 이용한 LED 제어 소스 - 계속

```
int main (int argc, char *argv[])
{
    int fd;
    static unsigned char led;

    /*
     * 주의: O_SYNC 플래그를 줘야지만 캐시되지 않는다.
     */
    fd = open( "/dev/mem", O_RDWR | O_SYNC );
    if (fd == -1) {
        perror("open(₩"/dev/mem₩")");
        exit(1);
    }

    /*
     * MAP_SIZE에 해당하는 페이지 만큼의 영역을 매핑한다.
     */
    led_addr = mmap( NULL, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, MAP_PHYS );
    if (led_addr == MAP_FAILED) {
        perror("mmap()");
        exit(2);
    }
}
```



Device를 open한다.

디바이스 제어 실습 - LED 제어

▶ mmap을 이용한 LED 제어 소스 - 계속

```
/*
 * <ctrl+c> 를 누르면 종료되게 signal을 등록한다.
 */
signal(SIGINT, quit_signal);
printf("\nPress <ctrl+c> to quit.\n\n");

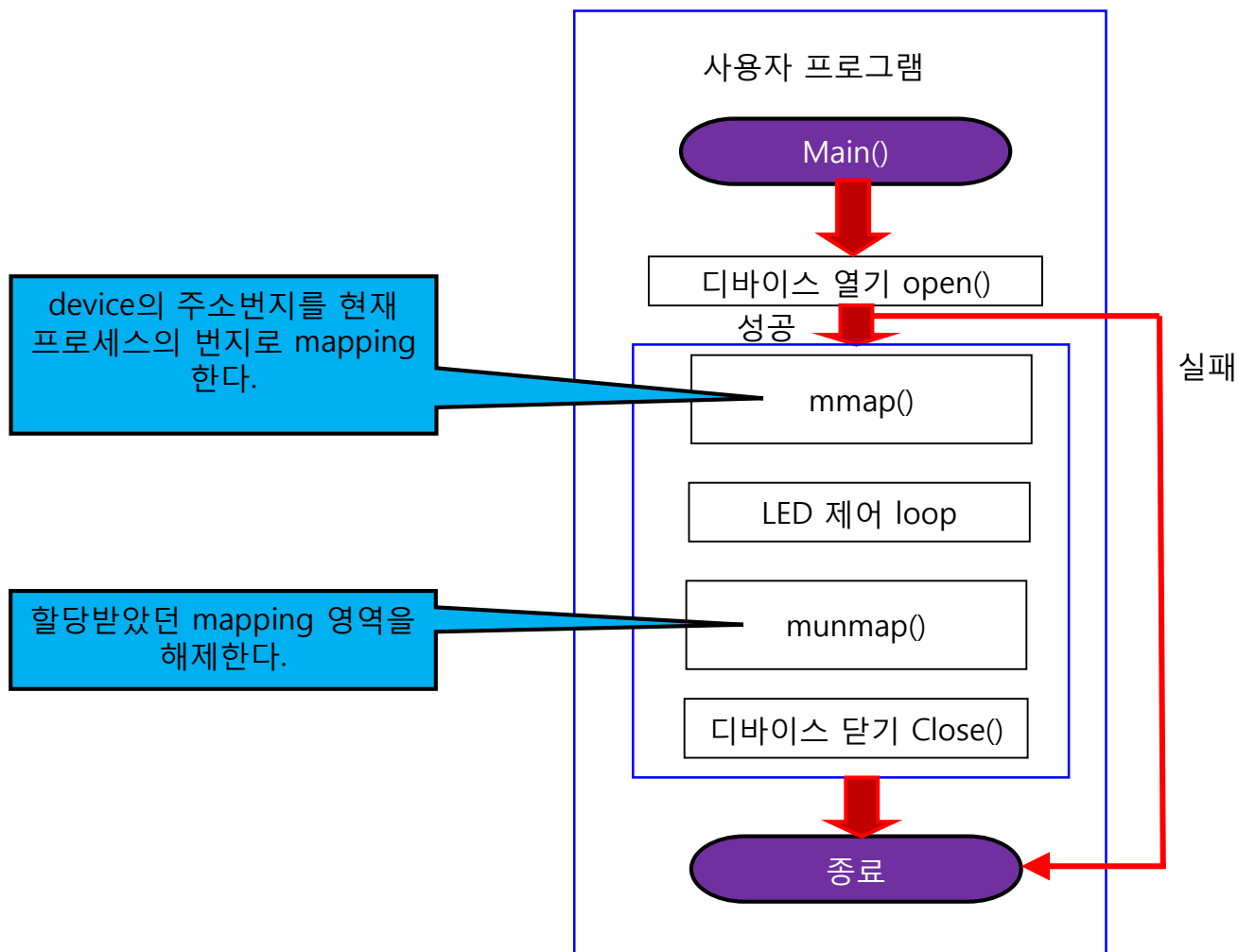
while(!quit)
{
    LED = ~(val[led++ % 9]);
    sleep(1);
}
LED = 0xffff;

/*
 * 할당받았던 매핑 영역을 해제한다.
 */
if (munmap(led_addr, MAP_SIZE) == -1) {
    perror("munmap()");
    exit(3);
}
close(fd);
return 0;
}
```

순차적으로 LED1~8을 점멸한다.

디바이스 제어 실습 - LED 제어

▶ mmap()을 이용한 LED 디바이스 제어 과정



디바이스 제어 실습 - LED 제어

▶ LED 제어 프로그램 컴파일 방법 및 타겟 보드로 전송

```
# arm-none-linux-gnueabi-gcc -o mmap_led mmap_led.c  
# cp mmap_led /nfsroot
```

▶ LED 제어 프로그램 실행 방법(타겟 보드)

▶ 생성된 'led' 실행 파일을 타겟 보드로 전송 후 타겟 보드에서 실행한다

```
# mount -t nfs 10.40.1.51:/nfsroot /mnt/nfs -o tcp,nolock  
# cp -a /mnt/nfs/mmap_led /root  
# cd /root  
# ./mmap_led
```

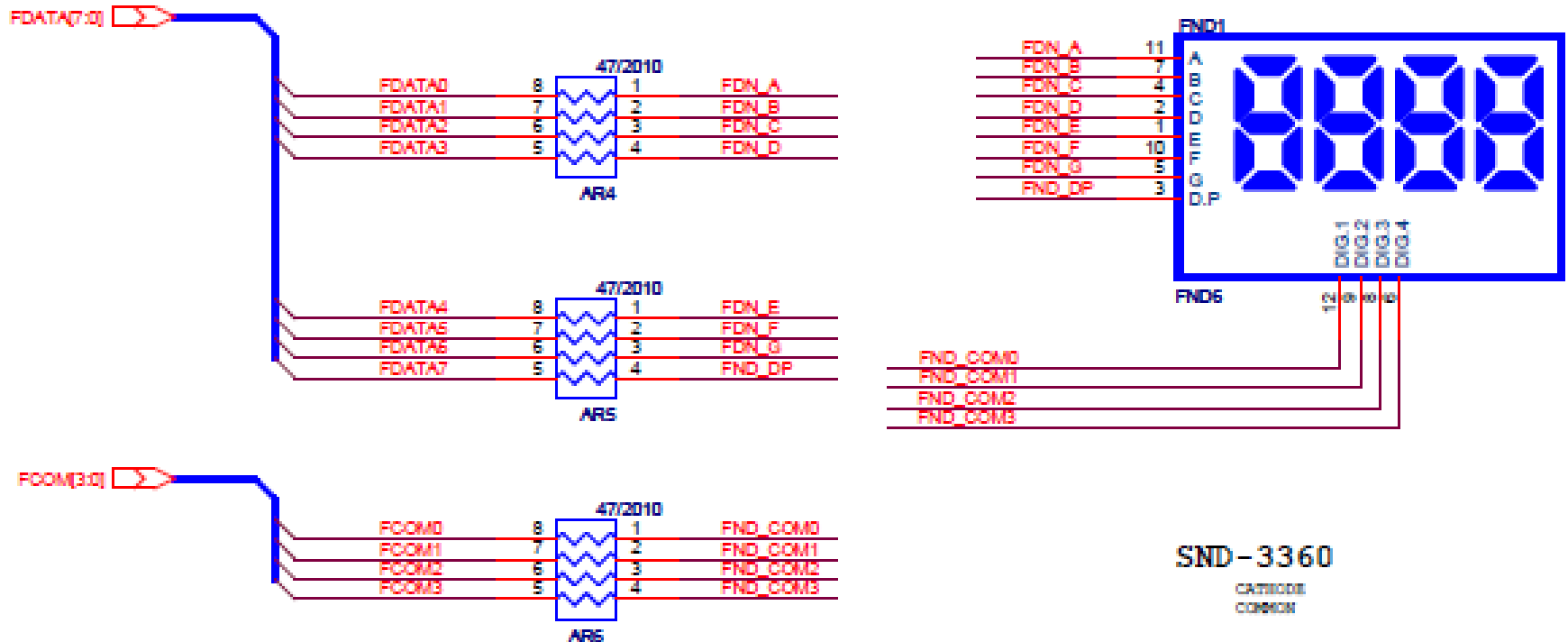
▶ LED가 순차적으로 점등



디바이스 제어 실습 - FND 제어

▶ FPGA FND Driver

▶ 회로



디바이스 제어 실습 - FND 제어

▶ FPGA FND Driver

▶ 주소

Digit			Address					
1			0x0700_0004					
2			0x0700_0005					
3			0x0700_0006					
4			0x0700_0007					
Bit	7	6	5	4	3	2	1	0
FND	D.P	G	F	E	D	C	B	A

디바이스 제어 실습 - FND 제어

▶ FND 제어 예시

▶ FND1, FND2 에 '12' 출력하기

▶ #define MAP_PHYS 0x70000000

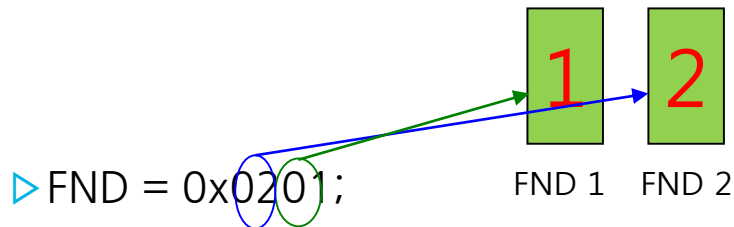
- FND가 연결된 물리주소이다.

▶ #define FND (*(volatile unsigned short *)(fnd_addr + FND_OFFSET))

▶ void *fnd_addr;

▶ fnd_addr = mmap(NULL, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, MAP_PHYS);

- MAP_PHYS를 mmap()하여 fnd_addr 로 변환된 가상주소를 받는다.



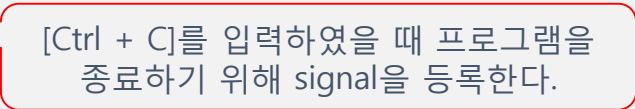
디바이스 제어 실습 - FND 제어

▶ mmap을 이용한 FND 제어 소스

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <ctype.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <signal.h>

#define MAP_SIZE      0x4000
#define MAP_PHYS      0x70000000          /* FND의 physical address */
#define FND_OFFSET    0x4
#define FND            (*((volatile unsigned short *) (fnd_addr + FND_OFFSET)))
void *fnd_addr;

int quit = 0;
void quit_signal(int sig)
{
    quit = 1;
}
```



디바이스 제어 실습 - FND 제어

▶ mmap을 이용한 FND 제어 소스

```
int main (int argc, char *argv[])
{
    int fd;
    static unsigned char fnd;

    /*
     * 주의: O_SYNC 플래그를 줘야지만 캐쉬되지 않는다.
     */
    fd = open( "/dev/mem", O_RDWR | O_SYNC );
    if (fd == -1) {
        perror("open(W\"/dev/mem\")");
        exit(1);
    }

    /*
     * MAP_SIZE에 해당하는 페이지 만큼의 영역을 매핑한다.
     */
    fnd_addr = mmap( NULL, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, MAP_PHYS );
    if (fnd_addr == MAP_FAILED) {
        perror("mmap()");
        exit(2);
    }
}
```

Device를 open한다.

디바이스 제어 실습 - FND 제어

▶ mmap을 이용한 FND 제어 소스 - 계속

```
/*
 *      <ctrl+c> 를 누르면 종료되게 signal을 등록한다.
 */
signal(SIGINT, quit_signal );
printf("WnPress <ctrl+c> to quit.WnWn");
```

signal을 등록한다.

```
while(!quit)
{
```

```
    FND = fnd % 10; FND |= (fnd % 10) << 8;
    FND |= (fnd % 10) << 16; FND |= (fnd % 10) << 24;
    fnd++;
    sleep(1);
```

1초 간격으로 '1~9'의 숫자를 FND에 출력한다.

```
}
FND = 0x00000000;
```

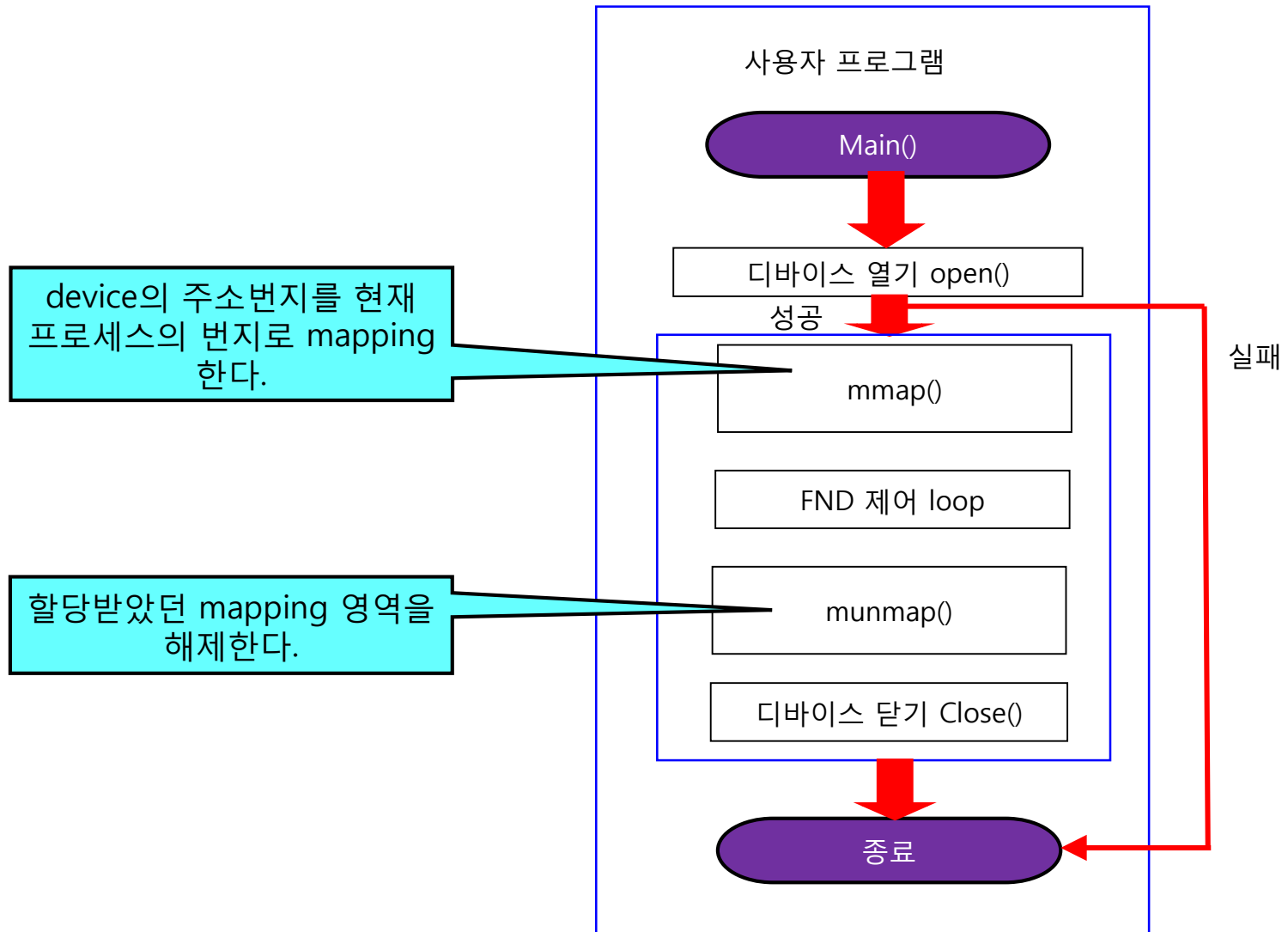
```
/*
 * 할당받았던 매핑 영역을 해제한다.
 */
```

```
if (munmap(fnd_addr, MAP_SIZE) == -1) {
    perror("munmap()");
    exit(3);
}
```

```
close(fd);
return 0;
```

```
}
```

디바이스 제어 실습 - FND 제어



디바이스 제어 실습 - FND 제어

▶ FND 제어 프로그램 컴파일 방법 및 타겟 보드로 전송

```
# arm-none-linux-gnueabi-gcc -o mmap_fnd mmap_fnd.c  
# cp mmap_fnd /nfsroot
```

▶ FND 제어 프로그램 실행 방법(타겟 보드)

▶ 생성된 'fnd' 실행 파일을 타겟 보드로 전송 후 타겟 보드에서 실행한다

```
# mount -t nfs 10.40.1.51:/nfsroot /mnt/nfs -o tcp,nolock  
# cp -a /mnt/nfs/mmap_fnd /root  
# cd /root  
# ./mmap_fnd
```



Q & A
