**Embedded System Software 과제 2**

**(과제 수행 결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20171640, 박수진**

**개발기간: 2021. 05. 03. - 2021. 05. 16.**

**최 종 보 고 서**

**I. 개발 목표**

- ioctl을 이용하여 유저 프로그램과 커널의 통신을 구현하며, 디바이스 드라이버에서 1) 전달된 디바이스 구동 옵션 저장 2) 디바이스 구동 2가지를 수행한다. 또한 타이머 기능을 바탕으로 fnd, led, dot matrix, text lcd 디바이스를 구동하는 모듈을 구현한다.

**II. 개발 범위 및 내용**

**가. 개발 범위**

1) user application (app.c)

- user로부터 옵션 입력 받기 및 예외처리

- ioctl system call호출

2) module (dev\_driver.c, dev\_driver.h)

- file operation function 구현: open, release, ioctl

- data processing

- kernel timer handling

- fpga device control

**나. 개발 내용**

- ioctl

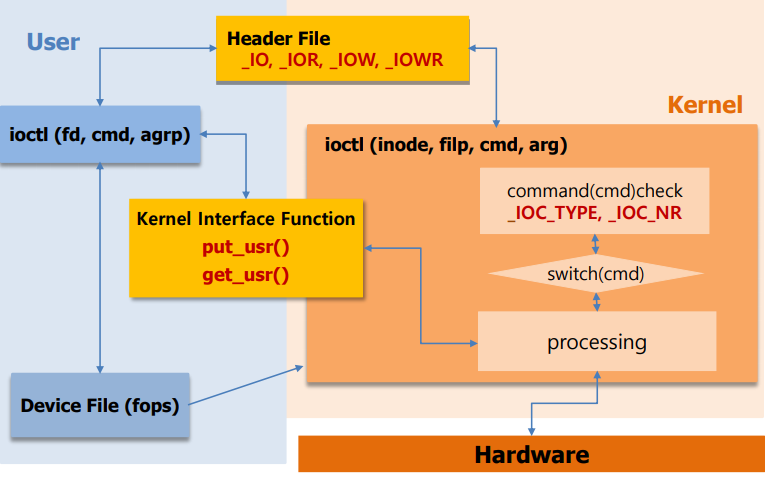
ioctl은 file operation의 하나로 입출력 제어 (I/O control)의 준말이다. Open, read, write와 마찬가지로 file operation struct에 ioctl 함수를 저장한 후 사용한다. User program에서 ioctl을 호출할 때 형식은 다음과 같다.

**int ioctl(int fd, int cmd, …)**

여기서 fd는 device file이며 cmd 는 사용자가 임의로 정할 수 있는 integer이다. 모듈의 ioctl함수에서는 cmd에 따라 다른 동작을 하게 구현할 수 있다. 3번째 argument는 optional하며 모듈의 ioctl로 전달할 parameter이다. Struct를 이용하면 많은 정보를 모듈의 ioctl로 넘길 수 있다. 이번 과제의 첫 번쨰 ioctl은 옵션 정보를 넘겨야 하므로 3번째 argument가 필요하지만, 두 번쨰 ioctl은 timer를 구동하는 것이 목적이므로 3번쨰 argument는 없어도 된다. Ioctl의 cmd를 define하는 형식은 다음과 같다.

**\_IOW(type, number, datatype)**

IOW는 IO direction을 나타내며 write만 하겠다는 의미이다. 이번 과제에서는 Read가 불필요하므로 IOW만 사용하였다. 그러나 이 부분은 크게 중요하지 않다. Type은 device의 major number를 의미하며, number는 command의 번호를 의미한다. 서로 다른 command는 서로 다른 번호를 가지며 임의로 지정 가능하다. Datatype은 전달되는 data type을 의미한다.

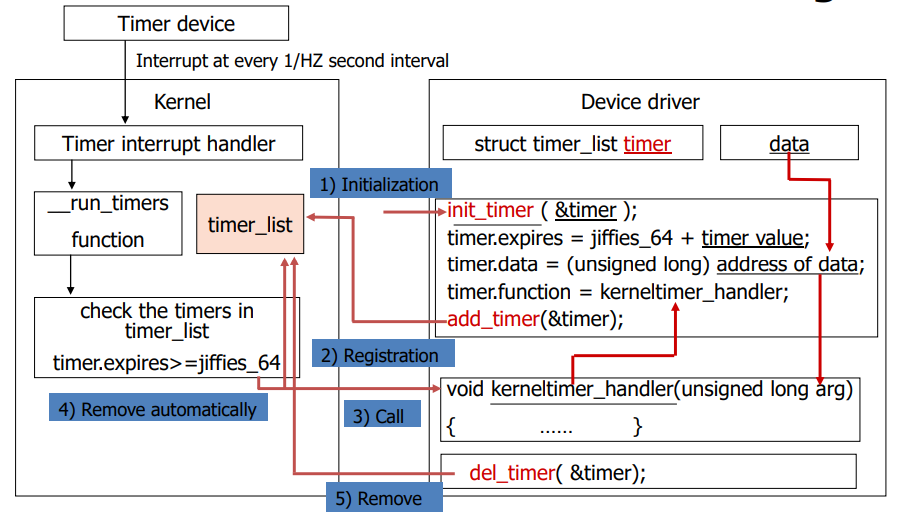


현재 커널 버전에서 ioctl function prototype은 다음과 같다.

**long device\_ioctl(struct file \*file, unsigned int ioctl\_num, unsigned long ioctl\_param)**

ioctl\_num으로 cmd에 따라 서로 다른 명령을 수행할 수 있다. ioctl\_param은 user application으로부터 전달된 데이터이다.

- kernel timer



Kernel timer는 다음과 같은 과정을 거친다.

1. Timer initialization’

init\_timer(&timer)

1. Timer list setting

Timer.expires, timer.data, timer.function

1. Timer registration

Add\_timer(&timer)

1. Timer removal

Del\_timer\_sync(&timer)

본 프로그램에서는 init\_timer는 module init 시에, del\_timer는 module exit 시에 수행하였다. Timer expire는 타이머가 만료되는 시간을 지정한다. 만약 10초 후에 타이머를 만료하고자 할 때 timer.expires 값은 get\_jiffies\_64() + (10 \* HZ)가 된다. Timer data는 timer에 저장할 data이며 본 프로그램에서는 timer list와 cnt를 저장할 count 변수를 저장하였다. 마지막으로 timer function은 timer가 expire될 때 불려지는 함수이다.

- device control

본 프로그램에서는 mapping방식을 사용하여 device를 control하였다. Mapping에 사용되는각 fpga의 address는 다음과 같다.

TEXT LCD 0x08000090

FND 0x08000004

LED 0x08000016

DOT MATRIX 0x08000210

Ioremap을 이용하여 mapping 하고 기능이 끝난 후 iounmap을 사용하여 unmapping하였다. 각각은 다음으로 mapping 된다.

TEXT LCD 0x32

FND 0x4

LED 0x1

DOT MATRIX 0x10

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 기간 | 내용 |
| 05/03 ~ 05/05 | Application program 완성 |
| 05/06 ~ 05/09 | 모듈 open, release, ioctl 기능 구현 |
| 05/10 ~ 05/12 | Data processing, Kernel timer handling |
| 05/13 ~ 05/15 | Fpga device control 구현 |
| 05/16 | 테스트 및 오류 수정 |

**나. 개발 방법**

1) user application (app.c)

- user로부터 옵션 입력 받기 및 예외처리

Argc, argv를 이용하고 argc가 4가 아닐 경우, usage에 표시된 range를 벗어나는 경우 또는 init의 형식에 어긋날 경우 에러 출력한다.

- ioctl 호출

Ioctl은 총 두 번 호출한다. 타이머 디바이스 옵션을 모듈로 전달할 때, 그리고 타이머 디바이스를 구동할 때이다.

각각의 cmd 번호를 0과 1로 설정하였으며 IOW direction으로 설정하였다.

Ioctl로 data를 전달할 structure가 별도로 필요하다.

2) module (dev\_driver.c, dev\_driver.h)

- file operation function 구현: open, release, ioctl

Fops에 해당 system call들을 저장한 후 각각에 맞는 함수를 구현해야 한다.

Open 함수에서는 timer init이 필요하고 release 함수에서는 timer removal이 필요하다.

ioctl에서는 user program에서 전달된 두가지 cmd를 모두 수행할 수 있도록 switch문을 사용하여 구현한다.

- data processing

타이머 옵션에 따라 time interval마다 data를 변경해주어야 한다.

초기 data에서 같은 자리에서 숫자를 1씩 올리다가 data가 8이 되면 다음 data는 1로 설정하고 자리를 오른쪽으로 옮긴다.

제일 오른쪽에 있으면 제일 왼쪽으로 옮긴다.

Text data는 time interval마다 좌우로 이동하도록 구현한다.

- kernel timer handling

Timer의 function에서는 data processing과 data 출력이 수행되도록 한다.

Timer data는 cnt값과 timer list를 가진다.

Timer expires는 get\_jiffies\_64() + ((time\_interval) \* HZ / 10)으로 설정하면 time interval입력이 1~100 일 때 시간 간격이 0.1~10초로 나타난다.

- fpga device control

Device control은 mmaping 방식을 사용한다.

개발 내용에 적힌 address를 사용하여 module init 시에 mapping하고, module exit 시에 unmapping 한다.

**IV. 연구 결과**

- app.c

텍스트이(가) 표시된 사진

자동 생성된 설명

Ioctl을 이용하여 전달할 data는 위와 같다. 프로그램 시작 시에 유저로부터 입력받은 timer option들이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

ioctl에서 device를 구동하기 위해 device file을 open한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ioctl에 사용될 cmd들은 위와 같이 정의하였다. IOW direction을 사용하며 SET\_OPTIONS은 0번, COMMAND는 1번 cmd로 설정하였다. 전달되는 data는 SET\_OPTION의 경우 USER\_DATA인데 COMMAND는 data 전달이 불필요하므로 임의로 int로 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자로부터 받은 타이머 옵션을 udata에 저장하여 ioctl을 호출한다. 타이머 옵션을 저장하는 ioctl으로 cmd는 SET\_OPTION이다. Ioctl에 실패할 시 에러 메시지를 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

타이머 디바이스를 구동하기 위한 ioctl이다. Ioctl의 3번째 argument는 불필요하므로 사용하지 않았다. COMMAND cmd를 사용하며 ioctl에 실패할 시 에러 메시지를 출력한다.

- dev\_driver.c

텍스트이(가) 표시된 사진

자동 생성된 설명

Module(device driver)을 init할 때는 먼저 device를 등록해야한다. Register\_chrdev 함수를 이용하여 dev name “dev\_driver”, major number 242로 설정하여 등록한다. Fops는 아래 사진과 같이 open, release, ioctl을 포함한다. 또한 device를 control하기 위해 각 fpga device를 ioremap으로 mapping한다. Timer를 사용하는 프로그램이므로 Timer를 initialization 해야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

File operation은 위와 같이 설정하였다. 본 프로그램에서 사용되는 system call은 open, release, 그리고 ioctl이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Open과 release에 사용되는 function은 위와 같다. Device를 open할 때 usage를 1로 올려주고 device를 release할 때 usage를 0으로 설정하였다. 해당 디바이스를 동시에 여러 군데에서 사용되지 않도록 하기 위함이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Module(device driver)를 exit 할 때는 mapping 했던 fpga device들을 unmap한다. 또한 init했던 timer도 removal해준다. 마지막으로 등록했던 device를 unregister한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 사진은 ioctl에서 SET\_OPTION cmd의 경우 수행되는 코드이다. 우선 user data를 kernel로 저장한 후 타이머의 interval, cnt, init data를 모두 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ioctl의 COMMAND cmd의 경우 타이머 디바이스를 직접적으로 구동해야 하므로 필요한 데이터 값들(fnd와 text lcd에 출력될 초기 data)을 세팅하고 타이머를 세팅한다. Timer의 expire는 입력된 interval / 10초 간격으로 설정한다. 따라서 디바이스가 구동된 후 입력된 interval 만큼의 시간이 지나야 디바이스가 동작하기 시작한다. 만약 입력된 interval이 10이면 app을 실행한 지 1초 후부터 시작되며 입력이 100이면 10초 후부터 시작된다. timer의 data는 아래와 같은 struct를 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Timer list와 timer가 총 작동될 횟수인 cnt를 저장하고 있다.

Timer의 function은 아래와 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Timer가 수행되는 횟수를 조절하기 위해 timer의 count를 1 decrease한다. 그리고 fpga device에 time에 해당하는 data를 출력하고 타이머의 세팅 값은 이전과 똑같이 한 후 timer를 add한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Led와 dot matrix를 출력할 때는 fnd에 출력되는 숫자를 이용한다. 만약 0040이면 led D4를 켜고 Dot matrix에는 4가 출력되도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

fnd는 저장된 fnd data를 출력한 후 다음 fnd 값으로 update한다. rot\_cnt에서 현재 몇 번째 rotation인지 확인하고 rotation을 다 돌았으면 0이 아닌 숫자가 출력되는 자리를 오른쪽으로 한 칸 이동한다. 가장 오른쪽에 있을 경우 가장 왼쪽으로 이동한다. 숫자도 1을 올려주고 8이 넘어가는 경우 다시 1로 세팅한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Text lcd도 우선 저장된 data를 먼저 lcd에 출력하고 다음 data를 update한다. 위쪽 text는 학번 20171640을 출력하고 아래 text는 이름 Sujin Park을 출력한다. 첫 문자가 출력되는 위치를 pos 변수들에 저장하고 만약 up\_pos가 8이면 방향을 왼쪽으로, 0이면 방향을 오른쪽으로 바꾼다. (pos가 8일 때 가장 마지막 문자가 제일 오른쪽에 도착하기 때문이다.) down\_pos도 마찬가지로 6이면 방향을 왼쪽으로, 0이면 방향을 오른쪽으로 바꾼다. 그 다음 위 아래 문자열을 합쳐서 다음 text lcd에 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

cnt만큼의 동작이 끝나면 모든 device를 clear해준다. Led는 모두 꺼주고 fnd는 0000으로 설정한다. Dot matrix는 모두 꺼주고 Text lcd는 모두 ‘ ‘로 설정한다.

**V. 기타**

- kernel timer를 직접 사용해 봄으로써 timer의 동작 방식을 더욱 잘 이해할 수 있게 되었습니다. Ioctl도 이론만 봤을 때는 어떻게 동작하는지 이해가 잘 안되었는데 직접 코드를 작성해보니 어떤 방식으로 작동되는지 flow를 이해할 수 있었습니다.