디바이스드라이버 프로그래밍(3)

Push Button 디바이스 드라이버

- blocked 입력
  - 입력이 공급될 때까지 기다려서, 입력이 공급되면 입력을 반환함
- 인터럽트를 사용한 blocked 입력
  - 버튼이 눌려질 때 까지 wait 상태가 됨
  - 버튼이 눌려지면 interrupt handler가 실행되어 wait 상태인 프로세스를 깨우고 눌려진 버튼 정보를 반환함
- Push Button Key Parallel Port
  - 주소: 0xFF200050
  - 레지스터:

Address	31 30 ... 4	3	2	1	0	
0xFF200050	Unused			KEY_{3-0}		Data register
Unused	Unused					
0xFF200058	Unused			Mask bits		Interruptmask register
0xFF20005C	Unused			Edge bits		Edgecapture register

Push Button 디바이스 드라이버(계속)

- 디바이스 드라이버의 예
  - push button을 1개의 device로 다룸 (/dev/key)
  - read() 함수 – 현재 눌려진 버튼 값을 읽음 (4-bit)
    - 인터럽트 구동 입력 – 버튼이 눌려질 때에 입력하여 반환
- 인터럽트 ID

I/O Peripheral	Interrupt ID #
FPGA Pushbutton KEYs	73

- interrupt mask register
  - 해당 bit가 1이면 해당 입력이 바뀔 때에 interrupt 요청
- edge capture register
  - 0에서 1로 변화된 입력 위치의 비트를 1로 설정
  - write 동작은 이 레지스터 값과 관련된 인터럽트를 clear 시킴

header 등

- 파일 "key.c"

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/interrupt.h>
#include <asm/io.h>

#include <linux/fs.h>
#include <linux/uaccess.h>
#include <linux/types.h>
#include <linux/ioport.h>

#include <linux/sched.h>

#define base_lwFPGA 0xFF200000
#define len_lwFPGA  0x200000

#define addr_LED    0
#define addr_HEX0   0x20
#define addr_HEX1   0x30
#define addr_SW     0x40
#define addr_KEY    0x50

#define offset_INTMASK  0x08
#define offset_EDGE 0x0C

#define KEY_DEVMAJOR    242
#define KEY_DEVNAME "keys"

MODULE_LICENSE("GPL");
MODULE_AUTHOR("SangKyun Yun");
MODULE_DESCRIPTION("DE1SoC Pushbutton Device Driver");

static void  *mem_base; // base virtual address of FPGA peripherals
static void  *key_addr; // virtual address of KEY Button
```

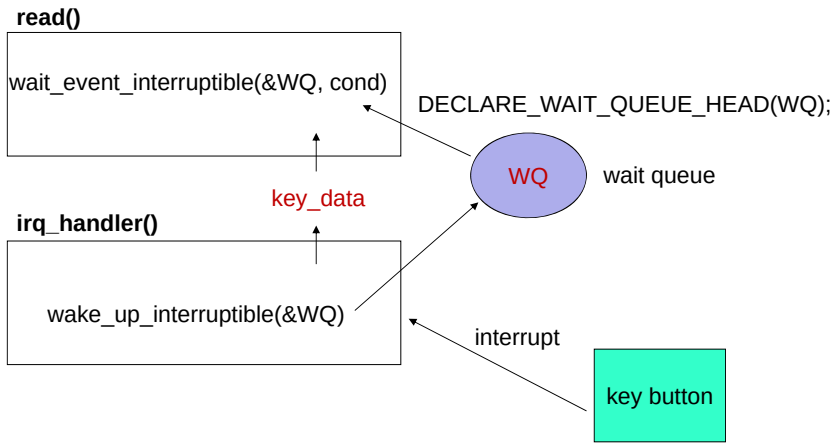
file operation 함수 작성

```
static int key_open(struct inode *minode, struct file *mfile)
{
    return 0;
}

static int key_release(struct inode *minode, struct file *mfile)
{
    return 0;
}

static ssize_t key_write (struct file *file, const char __user *buf, size_t
count, loff_t *f_pos)
{
    return 0;
}
```

- read() 함수는 다음 page

```
static struct file_operations key_fops = {
    .read       =   key_read,
    .write      =   key_write,
    .open       =   key_open,
    .release    =   key_release,
};
```


read - 인터럽트를 사용한 입력



read 함수/인터럽트 핸들러

```
static DECLARE_WAIT_QUEUE_HEAD(key_queue);
static int flag = 0;
unsigned int key_data;

static ssize_t key_read (struct file *file, char __user *buf, size_t count,
loff_t *f_pos){
    wait_event_interruptible(key_queue, flag != 0);
    flag = 0;
    put_user(key_data, buf);
    return 4;
}

irq_handler_t irq_handler(int irq, void *dev_id, struct pt_regs *regs)
{
    key_data = ioread32(key_addr + offset_EDGE);
    iowrite32(0xf, key_addr + offset_EDGE);

    flag = 1;
    wake_up_interruptible(&key_queue);

    return (irq_handler_t) IRQ_HANDLED;
}
```


모듈 init/exit 함수

```
static int __init keys_init(void)
{
    int res;

    res=register_chrdev(KEY_DEVMAJOR, KEY_DEVNAME, &key_fops);
    if(res < 0) {
        printk(KERN_ERR " push buttons : failed to register device.\n");
        return res;
    }

    mem_base = ioremap_nocache(base_lwFPGA, len_lwFPGA);
    if( !mem_base) {
        printk("Error mapping memory\n");
        release_mem_region(base_lwFPGA, len_lwFPGA);
        return -EBUSY;
    }
    key_addr = mem_base + addr_KEY;

    // Clear the PIO edgecapture register (clear any pending interrupt)
    iowrite32(0xf, key_addr+offset_EDGE);
    // Enable IRQ generation for the 4 buttons
    iowrite32(0xf, key_addr+offset_INTMASK);
```

```
    // Register the interrupt handler.
    res = request_irq(73, (irq_handler_t)irq_handler,
          IRQF_SHARED, "key_irq_handler", (void *)(irq_handler));
    if (res) {
        printk("failed to register key interrupt handler\n");
        return res;
    }
    printk("Device: %s MAJOR: %d\n", KEY_DEVNAME, KEY_DEVMAJOR);
    return 0;
}

static void __exit keys_exit(void){
    free_irq(73, (void*) irq_handler);
    iounmap(mem_base);
    unregister_chrdev(KEY_DEVMAJOR, KEY_DEVNAME);
    printk(" %s unregisterd.\n",KEY_DEVNAME);
}

module_init(keys_init);
module_exit(keys_exit);
```

"Makefile"

```
obj-m += key.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```


응용 프로그램

- 파일 "app_key.c"

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>

int main(void)
{
    int dev, data, rdata;

    dev = open("/dev/key", O_RDONLY);
    if (dev < 0) {
        fprintf(stderr, "cannot open KEY button device\n");
        return 1;
    }
    read(dev, &rdata, 4);
    printf("key data = %x\n", rdata);
    return 0;
}
```


실행

- 커널 모듈 설치

  # make
  # mknod /dev/key c 242 0
  # insmod key.ko
  # lsmod … 확인

- 응용 프로그램

  $ cc –o app_key app_key.c
  $ ./app_key
  key data = 2
  $

Push Button 디바이스 드라이버 개선하기

- 한 번에 한 프로세스만 장치를 사용 가능하도록 함
  - open, release 함수 수정

```
static int key_use = 0;

static int key_open(struct inode *minode, struct file *mfile)
{
    if ( test_and_set_bit(0, (void *)&key_use) ) return -EBUSY;
    return 0;
}

static int key_release(struct inode *minode, struct file *mfile)
{
    key_use = 0;
    return 0;
}
```