EDDI
Electronic Design
Development Institute

에디로봇아카데미
jetbot_system_peripheral summery

제 1기

2021. 12. 24

박성환

CONTENTS

EDDI
Electronic Design
Development Institute

CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 LED 제어 및 인터럽트 수신

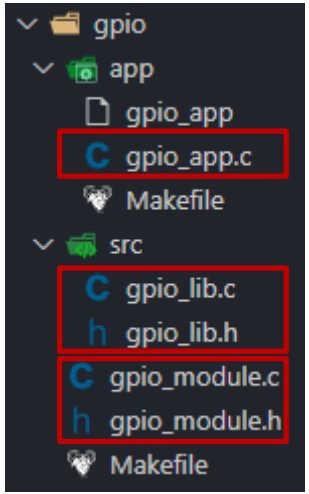
서론
- 리눅스에서 GPIO를 제어하는 방법에는 여러가지가 있음

① sys/class에서 gpio export해서 사용하는 방법
② gpio driver 구현하여 직접 컨트롤 하는 방법(커널직접/모듈)
③ 디바이스 트리를 사용하는 방법

- 위의 서론이 맞는지 정확히는 모르겠으나 우선 찾아본 바로는 그렇고
  1번은 간단히 터미널에서 실제 포트가 맞는지 확인용으로만 사용하고
  2번을 사용해서 구현할 예정

모듈 관련 명령어
- insmod : 커널 모듈 load
- rmmod : 커널 모듈 unload
- lsmod : 현재 커널에 올라와 있는 모듈 리스트
- modinfo : 현재 모듈에 대한 정보를 출력
- ls -l /dev : 현재 시스템에서 사용하고 있는 device(c/b, 주/부번호 파악 가능)
- mknod : 디바이스 파일 생성
- dmesg : 커널 메시지로 디바이스 드라이버 디버깅할 때 유용

파일 구성
- app : 실제 동작하는 main 함수로 구성
- src : 라이브러리 함수 구현
- module : gpio driver 코드 구현



예제

라이브러리

드라이버

CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 LED 제어 및 인터럽트 수신

GPIO_MODULE
- 아래 순서대로 모듈 코드 구현

① #include <linux/module.h> 등 추가
② MODULE 정보 입력
③ init/exit 구현
④ file_operations 함수 포인터 매칭
⑤ ioctl을 이용한 led 제어 코드 구현
⑥ 인터럽트 동작 구현
⑦ 드라이버 및 디바이스 파일 로딩

② MODULE 정보 입력
- 모듈 라이선스, 기본 정보 기입
- modinfo 로 아래 정보(해당 모듈의 버전이나 정보)를 알 수 있음

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Seonghwan Park");
MODULE_DESCRIPTION("GPIO input/ouput control");
MODULE_INFO(board, "Jetson AGX Xavier 4.9.253-tegra");
```

③ init/exit 구현
- 커널 모듈 등록과 해제
- 사용자 함수 gpiodrv_init, exit 구현 후 아래에 함수기입

```
module_init(gpiodrv_init);
module_exit(gpiodrv_exit);
```

④ file operations 함수 포인터 매칭
- 애플리케이션과의 저수준 입출력을 위해 file_operation 구조체 존재
- 함수 포인터로 매칭하여 사용

```
/* Match File Operation functions */
struct file_operations gpio_fops = {
    .open           = gpiodrv_open,
    .release        = gpiodrv_release,
    .write          = gpiodrv_write,
    .read           = gpiodrv_read,
    .unlocked_ioctl = gpiodrv_ioctl,
};
```

함수 포인터 매칭

CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 input/output 및 인터럽트 제어하기

③ init/exit 구현(init)

```c
static int __init gpiodrv_init(void)
{
    int ret;

    printk("GPIO Module is up... \n");

    if((result = gpiodrv_register_cdev()) < 0)
    {
        return result;
    }

#if 1 /* Interrupt initialize*/
    gpio_request(GPIO_PIN_249, "DOOR");
    door_irq = gpio_to_irq(GPIO_PIN_249);
    if( request_irq(door_irq, (void*)door_isr_func,
        IRQF_TRIGGER_RISING, "DOOR", NULL))
    {
        printk("failed to request external interrupt.\n");
        ret = -ENOENT;
        return ret;
    }
#endif

    return 0;
}
```

gpio 사용 할당(linux 제공 함수)

```c
static int gpiodrv_register_cdev(void)
{
    int error;

    /* allocation device number */
    if(gpiodrv_major) {
        gpiodrv_dev = MKDEV(gpiodrv_major, gpiodrv_minor);
        error = register_chrdev_region(gpiodrv_dev, 1, "sk");
    }
    else {
        error = alloc_chrdev_region(&gpiodrv_dev, gpiodrv_minor, 1, "sk");
        gpiodrv_major = MAJOR(gpiodrv_dev);
    }

    if(error < 0) {
        printk(KERN_WARNING "gpio : can't get major %d\n", gpiodrv_major);
        return result;
    }
    printk("major number=%d\n", gpiodrv_major);
    /* char device init & register cdev */
    cdev_init(&gpiodrv_cdev, &gpio_fops);
    gpiodrv_cdev.owner = THIS_MODULE;
    gpiodrv_cdev.ops = &gpio_fops;
    error = cdev_add(&gpiodrv_cdev, gpiodrv_dev, 1);

    if(error)
        printk(KERN_NOTICE "gpio Register Error %d\n", error);

    return 0;
}
```

주 번호 알때

주 번호 동적할당

(초기화할 cdev 구조체,등록될 fop포인터)

준비된 cdev 구조체를 커널에 등록

- 디바이스 드라이버를 사용하기 위해서는 커널에 등록하는 절차 필요
- 커널에 등록할 때는 주번호를 사용하여 등록이 필요
- 주번호는 이미 알고 있거나 동적할당이 가능

CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 input/output 및 인터럽트 제어하기

⑤ ioctl 함수를 이용한 led구현(자세히)
- 일반적으로 I/O Control에 관련한 작업을 수행하는 함수
- 대부분의 ioctl 메쏘드 구현은 cmd 인수값에 따라 올바른 동작을 수행하는 switch 문으로 구성함

```
#define GPIO_MAGIC  'k'
#define GPIO_MAXNR  6

typedef struct {
    unsigned int pinNumber;
    bool state;
} __attribute__((packed)) ioctl_info;
```

cmd 유효성 검사에 사용되는 파라미터

```
#define GPIO_SET_LED1       _IO(GPIO_MAGIC, 0)
#define GPIO_GET_LED1       _IO(GPIO_MAGIC, 1)
#define GPIO_SET_LED2       _IO(GPIO_MAGIC, 2)
#define GPIO_GET_LED2       _IO(GPIO_MAGIC, 3)
#define GPIO_SET_SOLENOID   _IO(GPIO_MAGIC, 4)
#define GPIO_GET_SOLENOID   _IO(GPIO_MAGIC, 5)
#define GPIO_GET_DOOR       _IO(GPIO_MAGIC, 6)
#define GPIO_GET_PHOTOSENSOR1 _IO(GPIO_MAGIC, 7)
#define GPIO_GET_PHOTOSENSOR2 _IO(GPIO_MAGIC, 8)
#define GPIO_SET_PCIE       _IO(GPIO_MAGIC, 9)
#define GPIO_GET_PCIE       _IO(GPIO_MAGIC, 10)
```

init 함수 내부(초기화)

GPIO 초기화 과정

```
/* initialize GPIO ports */
gpio_request(GPIO_PIN_424, "LED1");
gpio_request(GPIO_PIN_393, "LED2");
gpio_request(GPIO_PIN_256, "PHOTOSENS1");
gpio_request(GPIO_PIN_251, "PHOTOSENS2");
gpio_request(GPIO_PIN_250, "SOLENOID");
gpio_request(GPIO_PIN_289, "PCIE_12V");

/* 1.off the gpio pin value */
gpio_set_value(GPIO_PIN_424, 0);
gpio_set_value(GPIO_PIN_393, 0);
gpio_set_value(GPIO_PIN_250, 0);
gpio_set_value(GPIO_PIN_289, 0);
//gpio_set_value(GPIO_PIN_344, 0);

/* 2. set the control regisiter to output */
gpio_direction_output(GPIO_PIN_424, 0);
gpio_direction_output(GPIO_PIN_393, 0);
gpio_direction_output(GPIO_PIN_250, 0);
gpio_direction_output(GPIO_PIN_289, 0);

/* 3. set the input register */
gpio_direction_input(GPIO_PIN_256);
gpio_direction_input(GPIO_PIN_251);
```

```
static long gpiodrv_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{
    // sk_info ctrl_info;
    int size, err;
    int sts = 0;
```

애플리케이션에서 ioctl 함수 호출

cmd 유효성 검사

```
#if 1
    if(_IOC_TYPE(cmd) != GPIO_MAGIC) {
        printk(" _IOC_TYPE error\n");
        return -EINVAL;
    };
    if(_IOC_NR(cmd) >= GPIO_MAXNR) {
        printk("_IOC_NR error\n");
        return -EINVAL;
    };

    size = _IOC_SIZE(cmd);

    // POINT: check commands
    if (size) {
        err = 0;
        if(_IOC_DIR(cmd) & _IOC_READ)
//          err = verify_area(VERIFY_WRITE, (void *)arg, size);
            err = access_ok(VERIFY_WRITE, (void *)arg, size);
        else if (_IOC_DIR(cmd) & _IOC_WRITE)
//          err = verify_area(VERIFY_READ, (void *)arg, size);
            err = access_ok(VERIFY_READ, (void *)arg, size);
        if (err) return err;
    }
#endif
```

실제 LED On/OFF

```
// allows to access device using ioctl commands
switch(cmd) {
    case GPIO_SET_LED1:
        printk("GPIO_SET_LED1\n");
        gpio_set_value(GPIO_PIN_424, arg);
        break;
    case GPIO_GET_LED1:
        printk("GPIO_GET_LED1\n");
        sts = gpio_get_value(GPIO_PIN_424);
        break;
    case GPIO_SET_LED2:
        printk("GPIO_SET_LED2\n");
        gpio_set_value(GPIO_PIN_393, arg);
        break;
    case GPIO_GET_LED2:
        printk("GPIO_GET_LED2\n");
        sts = gpio_get_value(GPIO_PIN_393);
        break;
    case GPIO_GET_DOOR:
        printk("GPIO_GET_DOOR\n");
        sts = gpio_get_value(GPIO_PIN_249);
        break;
    case GPIO_SET_PCIE:
        printk("GPIO_SET_PCIE\n");
        gpio_set_value(GPIO_PIN_289,arg);
        break;
    case GPIO_GET_PCIE:
        printk("GPIO_GET_PCIE\n");
        sts = gpio_get_value(GPIO_PIN_289);
        break;
    default:
        printk("Can't find any cmd\n");
        sts = -1;
        break;
}
```

CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 input/output 및 인터럽트 제어하기

⑥ 인터럽트 동작 구현(App 영역에 시그널 전달 포함)
- 드라이버에서가 아닌 user 영역인 App 영역에 인터럽트 시그널 전달을 어떻게 전달하는지에 대한 과정

gpio_app.c(사용자)

⑤
```
#if 1
void signal_handler(int signum)
{
    if(gpio_getValue(Door_sensor1))
        printf("Door_sensor1 is Closed\n");
#if 0
    if(signum == SIGIO) { /* SIGIO 신호면 종료 빠져나오기 */
        printf("SIGIO\r\n");
        exit(1);
    }
#endif
}
```
시그널 처리를 위한 핸들러 등록
SIGIO 신호가 오면 signal_handler 처리

②
```
    signal(SIGIO, signal_handler); //for Interrupt

#if 1 /* only send pid */
    sprintf(buf, "%d", getpid());
    write(fd, buf, strlen(buf)); //send the pid of this process
#else /* send c
    // sprintf(
    // write(fd,
#endif

#if 1 /* read pid */
    if(read(fd, buf, strlen(buf)) == 0)
        printf("Got PID Number %s\n", buf);
#endif
```
현재 프로세서의 PID 정보를
커널에 알려줘야 함

gpio_module.c(드라이버)

③
```
/* when signal occured, register PID to send */
pid = simple_strtol(pidstr, &endptr, 10); //string to long
if(endptr != NULL) {
    task = pid_task(find_vpid(pid), PIDTYPE_PID);
    if(task == NULL) {
        printk("Error : Can't find PID from user application\n");
        return 0;
    }
}
```
시그널 발생시 보낼 프로세서 ID 등록

①
```
#if 1 /* Interrupt initialize*/
    gpio_request(GPIO_PIN_249, "DOOR");
    door_irq = gpio_to_irq(GPIO_PIN_249);
    if( request_irq(door_irq, (void*)door_isr_func,
        IRQF_TRIGGER_RISING, "DOOR", NULL))
    {
        printk("failed to request external interrupt.\n");
        ret = -ENOENT;
        return ret;
    }
#endif
```
GPIO 핀번호 IRQ번호로 매핑

커널에 IRQ요청하여 핸들러 등록

④
```
/* Implementation of functions */
static irqreturn_t door_isr_func(int irq, void *dev_id)
{
    if(irq == door_irq && gpio_get_value(GPIO_PIN_249)) {
#if 1
        static struct siginfo sinfo; /* struct for signal */
        memset(&sinfo, 0, sizeof(struct siginfo));
        sinfo.si_signo = SIGIO;
        sinfo.si_code = SI_USER;
        send_sig_info(SIGIO, &sinfo, task); /* send the signal to the process */
#else
        sys_kill(pid, SIGINT); //sys_kill is not woring well
```
인터럽트 서비스 루틴

인터럽트 서비스 루틴

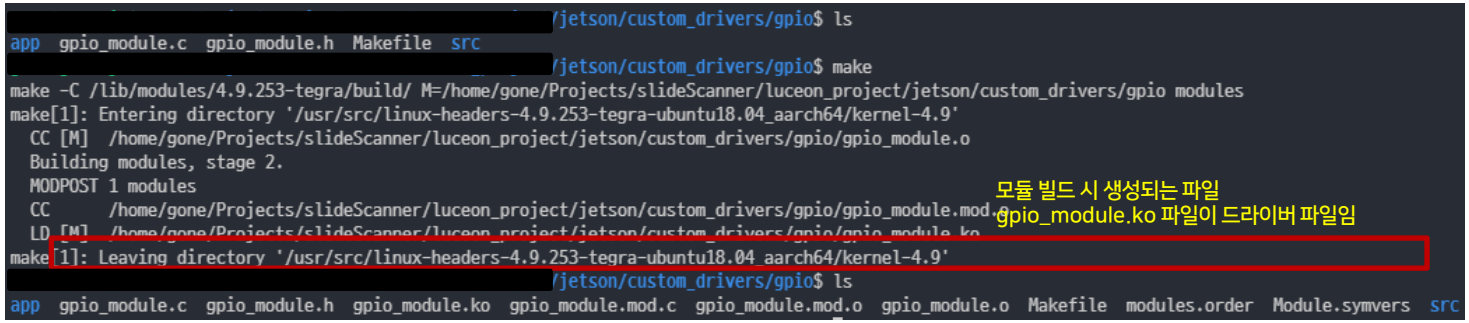
CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 input/output 및 인터럽트 제어하기

⑦ 드라이버 및 디바이스 파일 로딩

1) gpio module 빌드
- make 명령어로 Makefile 규칙대로 컴파일 실행
- xxx.ko 드라이버 파일을 포함한 여러 파일들이 생성됨



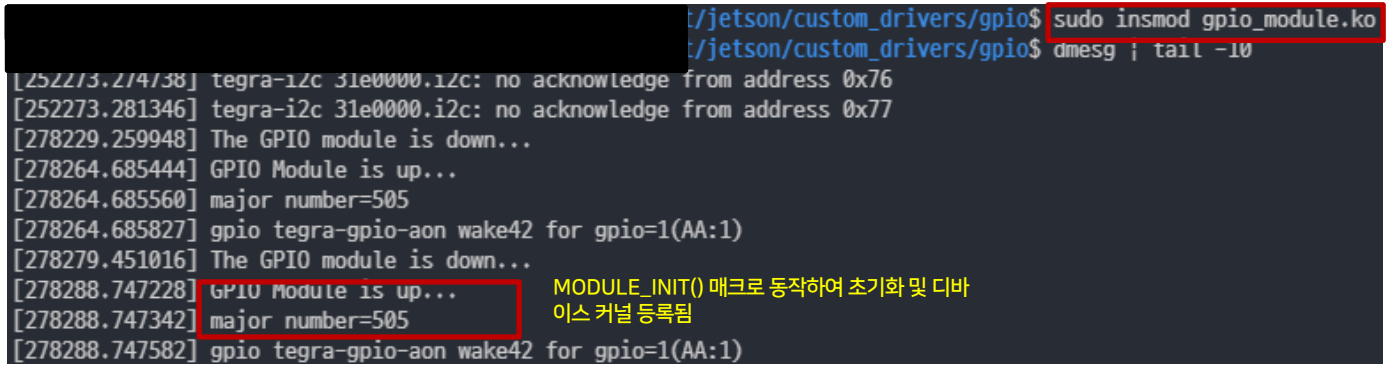
CONTENTS

EDDI
Electronic Design Development Institute

GPIO를 통한 input/output 및 인터럽트 제어하기

⑦ 드라이버 및 디바이스 파일 로딩

2) gpio module load
- insmod로 gpio module load(MODULE_INIT() 동작)
- Major number(주번호) 505번이 동적할당됨을 확인함
- xxx.ko 드라이버 파일을 포함한 여러 파일들이 생성됨



CONTENTS

EDDI
Electronic Design
Development Institute

GPIO를 통한 input/output 및 인터럽트 제어하기

⑦ 드라이버 및 디바이스 파일 로딩

3) device 파일 생성
- 사용자는 device file을 이용해 장치를 사용할 수 있음
- device driver로 사용할 장치 파일을 만드는 것
- 장치 파일은 사용자와 device driver를 연결해 주는 매개체

```
gone@gone-jetson:~/Projects/slideScanner/luceon_project/jetson/custom_drivers/gpio$ sudo mknod /dev/GPIODEV c 505 0
gone@gone-jetson:~/Projects/slideScanner/luceon_project/jetson/custom_drivers/gpio$ ls -al /dev/GPIO*
crw-r--r-- 1 root root 505, 0 12월 17 15:52 /dev/GPIODEV
```

CONTENTS

EDDI
Electronic Design
Development Institute

CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

서론
- jetson에서 제공하는 커널에 기본 i2c 드라이버를 제공해줌
- 사용자 영역에서 파일 컨트롤으로 쉽게 제어가 가능함
- 아래 include 해줘야 함

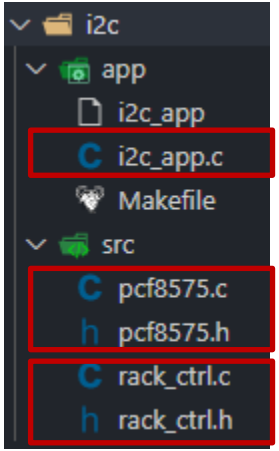
```
#include <linux/i2c-dev.h>
#include <linux/i2c.h>
```

현재 생성된 i2c 관련 디바이스 파일 리스트 확인

```
gone@gone-jetson:~$ ls -l /dev/i2c*
crw-rw---- 1 root i2c 89, 0 12월 14 10:32 /dev/i2c-0
crw-rw---- 1 root i2c 89, 1 12월 14 10:32 /dev/i2c-1
crw-rw---- 1 root i2c 89, 2 12월 14 10:32 /dev/i2c-2
crw-rw---- 1 root i2c 89, 3 12월 14 10:32 /dev/i2c-3
crw-rw---- 1 root i2c 89, 4 12월 14 10:32 /dev/i2c-4
crw-rw---- 1 root i2c 89, 5 12월 14 10:32 /dev/i2c-5
crw-rw---- 1 root i2c 89, 6 12월 14 10:32 /dev/i2c-6
crw-rw---- 1 root i2c 89, 7 12월 14 10:32 /dev/i2c-7
crw-rw---- 1 root i2c 89, 8 12월 14 10:32 /dev/i2c-8
```

사전 준비
- 아래 패키지 설치 필요
  sudo apt-get install i2c-tools
  sudo apt-get install libi2c-dev

폴더 구성
- app : 실제 동작하는 main 함수로 구성
- src : 라이브러리 함수 구현



예제

Expansion IO

Rack 관련 라이브러리

CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

i2c 핀맵

Jetson AGX Xavier Expansion Header

Sysfs GPIO	Connector Label	Pin	Pin	Connector Label	Sysfs GPIO
	3.3 VDC *Power, 1A max*	1	2	**5.0 VDC** *Power, 1A max*	
	I2C_GP5_DAT *General I2C #5 Data* *1.8/3.3V, I2C Bus 8*	3	4	**5.0 VDC** *Power, 1A max*	
	I2C_GP5_CLK *General I2C #5 Clock* *1.8/3.3V, I2C Bus 8*	5	6	**GND**	
gpio422	**MCLK05** *Audio Master Clock* *1.8/3.3V*	7	8	**UART1_TX** *UART #1 Transmit* *3.3V*	
	GND	9	10	**UART1_RX** *UART #1 Receive* *3.3V*	

i2c 장치 확인

- 디바이스 파일 확인 : ls -al /dev/i2c*

```
[redacted]/jetson$ ls -al /dev/i2c*
crw-rw---- 1 root i2c 89, 0 12월 14 10:32 /dev/i2c-0
crw-rw---- 1 root i2c 89, 1 12월 14 10:32 /dev/i2c-1
crw-rw---- 1 root i2c 89, 2 12월 14 10:32 /dev/i2c-2
crw-rw---- 1 root i2c 89, 3 12월 14 10:32 /dev/i2c-3
crw-rw---- 1 root i2c 89, 4 12월 14 10:32 /dev/i2c-4
crw-rw---- 1 root i2c 89, 5 12월 14 10:32 /dev/i2c-5
crw-rw---- 1 root i2c 89, 6 12월 14 10:32 /dev/i2c-6
crw-rw---- 1 root i2c 89, 7 12월 14 10:32 /dev/i2c-7
crw-rw---- 1 root i2c 89, 8 12월 14 10:32 /dev/i2c-8
```

- 장치 연결 확인 : i2cdetect -r 8 (r: return 값, 8: 디바이스 파일 번호)

```
[redacted]/jetson$ i2cdetect -r 8
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-8 using receive byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n]
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: 20 21 22 -- 24 25 -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

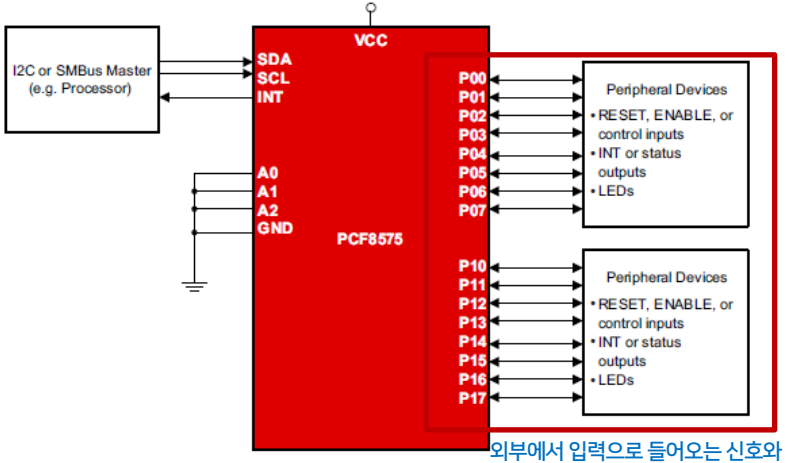
연결된 5개의 PCF8575 chip address 나타남

CONTENTS

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

PCF8575 datasheet 확인
- 총 16개의 Input/Ouput 포트 제공

Simplified Schematic



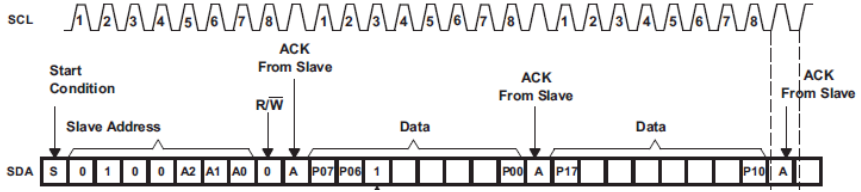
외부에서 입력으로 들어오는 신호와 내가 쓰는 신호가 충돌 경우가 생길 수 있을 것 같은데 해당 의문을 가지고 datasheet 주의사항 접근

Address

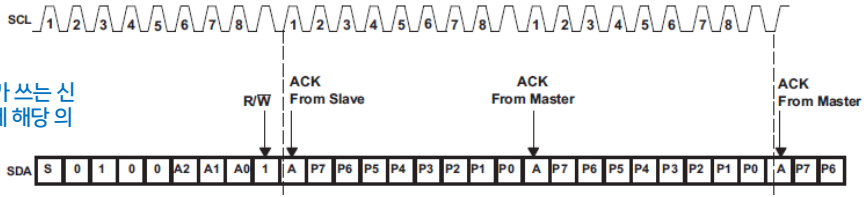
address 확인

BYTE	BIT							
	7 (MSB)	6	5	4	3	2	1	0 (LSB)
I²C slave address	L	H	L	L	A2	A1	A0	R/$\overline{W}$
P0x I/O data bus	P07	P06	P05	P04	P03	P02	P01	P00
P1x I/O data bus	P17	P16	P15	P14	P13	P12	P11	P10

Write



Read

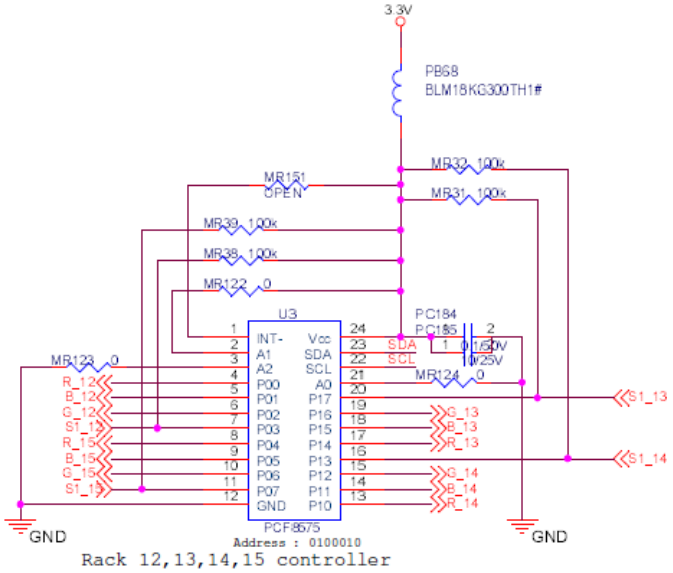


CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

회로 구성
- Output : R,G,B LED 사용
- Input : 스위치 사용



- 회로적으로 스위치가 핀 배열 곧곧에 배치되어 있어 배열로 코드 구현하기에 조금 귀찮은 면이 있음

CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

PCF8575 코드 구현

Open

```
bool PCF8575_open(void)
{
    char fileNameBuffer[32];

    sprintf(fileNameBuffer,"/dev/i2c-%d", kI2CBus);
    kI2CFileDescriptor = open(fileNameBuffer, O_RDWR);
    if (kI2CFileDescriptor < 0) {
        // Could not open the file
        error = errno ;
        return -1;
    }
    if (ioctl(kI2CFileDescriptor, I2C_SLAVE, kI2CAddress) < 0) {
        // Could not open the device on the bus
        error = errno ;
        return -1;
    }

    //printf("PCF8575 Device Address: 0x%02X\n",kI2CAddress) ;
    //printf("PCF8575 Device Bus: 0x%02X\n",kI2CBus) ;
    return 0;
}
```

이미 만들어져 있는 device file open

fd에 i2c slave 사용 지정

Close

```
void PCF8575_close(void)
{
    if (kI2CFileDescriptor > 0) {
        close(kI2CFileDescriptor);
        // WARNING - this is not quite right, need to check for error first
        kI2CFileDescriptor = -1;
        printf("PCF8575 BUS closed : 0x%02X\n",kI2CBus);
    }
}
```

CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

PCF8575 코드 구현

Write

```
int PCF8575_write(int addr, int writeValue)
{
    unsigned char buf[2];
    int toReturn;

    ioctl(kI2CFileDescriptor, I2C_SLAVE_FORCE, addr);

    buf[1] = (writeValue & 0xFF00) >> 8;
    buf[0] = writeValue & 0xFF;

    if ((toReturn = write(kI2CFileDescriptor, buf, 2)) < 0) {
        fprintf(stderr, "Writing error! (%X): %s\r\n", writeValue, strerror(errno));
        exit(EXIT_FAILURE);
        // return -1;
    }

    return toReturn;
}
```

사용하는 i2c address 가 5개 이므로 i2c write시 fd에 해다아는 address를 미리 할당해줘야 함

write 로 data 쓰기

Read

```
int PCF8575_read(int addr)
{
    int toReturn;
    int tmpData;
    unsigned char buf[2];
    int returnData;
    ioctl(kI2CFileDescriptor, I2C_SLAVE_FORCE, addr);

    if((toReturn = read(kI2CFileDescriptor, buf, 2)) < 0) {
        fprintf(stderr, "Reading error! (%X): %s\r\n", addr, strerror(errno));
        exit(EXIT_FAILURE);
        // return -1;
    }
    tmpData = buf[0] + (buf[1] << 8);
    // fprintf(stdout, "tmpData = 0x%04X\r\n",tmpData);

    /* Before Reading, should input Port set to high */
    tmpData = tmpData | ((0x1<<15) + (0x1<<11) + (0x1<<7) + (0x1<<3));
    // fprintf(stdout, "tmpData = 0x%04X\r\n",tmpData);

    PCF8575_write(addr, tmpData);

    if((toReturn = read(kI2CFileDescriptor, buf, 2)) < 0) {
        fprintf(stderr, "Reading error! (%X): %s\r\n", addr, strerror(errno));
        exit(EXIT_FAILURE);
        //return -1;
    }
    returnData = buf[0] + (buf[1] << 8);
    // fprintf(stdout, "Read 2byte = 0x%04X\r\n",returnData);

    return returnData;
}
```

datasheet에 근거하여 read로 핀을 읽기 전에 반드시 해당 핀을 high로 써준 후 읽어야 함

해당 핀만을 high로 써줘야 하기 때문에 우선 현재 핀 상태를 불러온 후에 해당 핀만 high로 변경해주는 과정 필요

실제 읽는 read

CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

Rack 코드 컨셉
- Rack 관련 정보를 구조체로 표현
- 배열로 정보를 구성하여 해당 정보를 찾는 컨셉

```
typedef enum _Rack_Color {
    WHITE = 0x0,
    RED = 0x6,
    BLUE = 0x5,
    GREEN = 0x3,
    OFFLED = 0x7
} Rack_Color;
```

```
typedef struct _Rack_State {
    Rack_Color led;
    bool sw;
} Rack_State;
```

Rack 1개당 관련된 정보를 구조체 화

```
typedef struct _Rack_Info {
    const Rack_Num num;
    const unsigned char addr;
    const unsigned int posParam;
    Rack_State state;
} Rack_Info;
```

```
typedef enum _Rack_Num {
    RACK_1,
    RACK_2,
    RACK_3,
    RACK_4,
    RACK_5,
    RACK_6,
    RACK_7,
    RACK_8,
    RACK_9,
    RACK_10,
    RACK_11,
    RACK_12,
    RACK_13,
```

```
Rack_Info RackInfo[] = {  { RACK 번호,  i2c addr, pos parameter, {초기 LED 상태, 초기 스위치 상태} }
    {RACK_1, 0x21, 0, {OFFLED, DISCONN}},    {RACK_2, 0x20, 8, {OFFLED, DISCONN}},    {RACK_3, 0x20, 12, {OFFLED, DISCONN}},
    {RACK_4, 0x20, 0, {OFFLED, DISCONN}},    {RACK_5, 0x20, 4, {OFFLED, DISCONN}},    {RACK_6, 0x24, 0, {OFFLED, DISCONN}},
    {RACK_7, 0x24, 4, {OFFLED, DISCONN}},    {RACK_8, 0x24, 8, {OFFLED, DISCONN}},    {RACK_9, 0x24, 12, {OFFLED, DISCONN}},
    {RACK_10, 0x21, 12, {OFFLED, DISCONN}},  {RACK_11, 0x21, 8, {OFFLED, DISCONN}},  {RACK_12, 0x22, 0, {OFFLED, DISCONN}},
    {RACK_13, 0x22, 12, {OFFLED, DISCONN}},  {RACK_14, 0x22, 8, {OFFLED, DISCONN}},  {RACK_15, 0x22, 4, {OFFLED, DISCONN}},
    {RACK_16, 0x25, 12, {OFFLED, DISCONN}},  {RACK_17, 0x25, 8, {OFFLED, DISCONN}},  {RACK_18, 0x25, 4, {OFFLED, DISCONN}}
};
```

CONTENTS

EDDI
Electronic Design Development Institute

I2C통신을 사용한 Expansion IO (PCF8575) 제어하기

Rack 관련 주요 함수 표기
- 배열로 정보를 구성하여 해당 정보를 찾는 컨셉
- 원하는 RACK에 LED 쓰는 함수와 현재 RACK 연결상태 읽는 함수 예시
- 다른 함수들은 코드 참조

```
int Rack_setLed(Rack_Num rackNum, Rack_Color rackColor)
{
    int i;
    int data;
    int toReturn;

    for(i = 0; i < RACK_CNT; i++)
    {
        if(RackInfo[i].num == rackNum)
        {
            data = PCF8575_read(RackInfo[i].addr);
            RackInfo[i].state.led = rackColor;
            data = (data & ~(0x7 << RackInfo[i].posParam)) | (rackColor << RackInfo[i].posParam);
            toReturn = PCF8575_write(RackInfo[i].addr, data);
            if(toReturn < 0) {
                printf("Cannot set the led \n");
                return -1;
            }
            return toReturn;
        }
    }

    printf("Cannot find the rack %d\n", rackNum);
    return -1;
}
```

RACK 개수 만큼 count

찾고자 하는 Rack 번호가 동일한 경우 아래 과정 실행

```
int Rack_getConn(Rack_Num rackNum)
{
#if 1
    int i;
    int data;

    for(i = 0; i < RACK_CNT; i++)
    {
        if(RackInfo[i].num == rackNum)
        {
            data = PCF8575_read(RackInfo[i].addr);
            // printf("data = 0x%04X\r\n", data);
            data = (data & (0x1 << (RackInfo[i].posParam+3))) >> (RackInfo[i].posParam+3);
            // printf("data = 0x%04X\r\n", data);
            // data = (data & (0xF << RackInfo[i].posParam)) >> RackInfo[i].posParam;
            // data = (data & (0x1 << 3)) >> 3;
            return (data? DISCONN : CONN) ;
        }
    }

    printf("Cannot find the rack %d\n", rackNum);
    return -1;
```

CONTENTS

EDDI
Electronic Design
Development Institute

CONTENTS

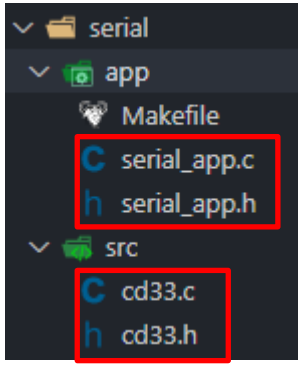
EDDI
Electronic Design Development Institute

Serial통신을 이용한 Laser센서(cd33) 제어하기

서론
- jetson에서 제공하는 커널에 기본 uart 드라이버를 제공해줌
- 사용자 영역에서 파일 컨트롤으로 쉽게 제어가 가능함
- /dev/ttyTHS0이 우리가 사용하는 파일인데 그냥 운좋게 0번이여서 동작하기는 했는데 실제 ttyTHS4 인 경우 우리가 어떻게 4번인걸 알고 찾아야 하는지 고민 필요
- #include <termios.h> 필요

```
gone@gone-jetson:~$ ls -l /dev/ttyTHS*
crw-rw---- 1 root dialout 238, 0 12월 23 11:48 /dev/ttyTHS0
crw-rw---- 1 root dialout 238, 1 12월 23 09:49 /dev/ttyTHS1
crw-rw---- 1 root dialout 238, 4 12월 23 09:49 /dev/ttyTHS4
```

폴더 구성
- app : 실제 동작하는 main 함수로 구성
- src : 라이브러리 함수 구현



app

RS422 통신을 하는 lasor 센서

CONTENTS

EDDI
Electronic Design Development Institute

Serial통신을 이용한 Laser센서(cd33) 제어하기

cd33 datasheet 확인
- CD33 datasheet로 확인함(동일 모델)

●Specification

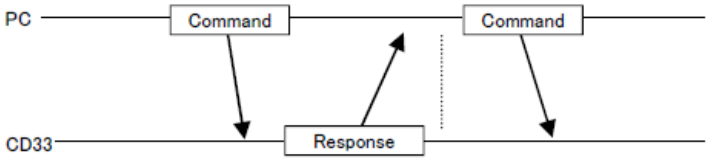
Communication method	RS422
Synchro system	Asynchronous
Baud rate	9600/19200/38400/76800 bps *
Transmission code	ASCII
Data length	8 bit
Stop bit length	1 bit
Parity check	Nil
Data classification	STX･ETX

* Baud rate :9600bps at factory set

●Communication Procedure

명령 1개당 반드시 응답 1개를 기다림

When PC sends a command to CD33 it sends back a response to the PC.
In principle one response is given to one command. When sending a command, make sure if you receive the response to the previous command.



* Avg indication will turn on whenever a command response comes.

CONTENTS

Serial통신을 이용한 Laser센서(cd33) 제어하기

●Incoming Data Format (Response)

02H		03H
STX	RESPONSE	ETX
1	2	3

1 The code showing the head of incoming data (02H).
2 The response data is set to the transmitted command.
3 The code showing the completion of incoming data (03H).

The following four responses are for the written commands:
> (3EH) :Writings completed
? (3FH) :Writings rejected due to wrong command, etc.
(Numerical value) : Measurements or settings

Continuous readout of measurement value
Readout the measurements continuously at "START_MEASURE" command. The response of this case never has STX, ETX. CR(0DH) is inserted between the measurements.
(ex.)
85.0000<CR>85.0001<CR>85.0…

Sure to use the command "STOP_MEASURE" to stop the continuous reading. Any other command will be valid until the stop command is set.
Continuous reading will not be activated simultaneously.

●Transmission Data Format (Command)

Reading out Setting/Measurement Value/Output Status

02H		03H
STX	COMMAND	ETX
1	2	3

STX , ETX 구분
COMMAND는 ASCII 문자열

1 The code showing the head of transmit data (02H).
2 Selects the command to transmit.

추후 필요시 구현

Writing the setting

02H		20H		03H
STX	COMMAND	SPACE	COMMAND	ETX
1	2	3	4	5

1 The code showing the head of transmit data (02H).
2 Selects the command to transmit.
3 Shows the separation between Command and Command (20H).
4 Set the Setting/Measurement Value/Output Status.

CONTENTS

EDDI
Electronic Design Development Institute

Serial통신을 이용한 Laser센서(cd33) 제어하기

시작과 끝이 STX, ETX가 아닌 경우도 존재

<For diffuse reflection /specular reflection type>

Command		type*	Initial value	Description	Example of Response
Read the measurements	START_MEASURE	CR	–	Start continuous reading of measurements	85.0000[CR]85.0001[CR]85.0…
	STOP_MEASURE	–	–	Stop continuous reading of measurements	[STX] > [ETX]
	MEASURE	R	–	Read the measurements	[STX] 85.0000 [ETX]
	START_MEASURE_S	CR	–	Start continuous reading of measurements and sensitivity *1	85.0000 121[CR]85.0001 121[CR]85.0…
	STOP_MEASURE_S	–	–	Stop continuous reading of measurements and sensitivity *1	[STX] > [ETX]
	MEASURE_S	R	–	Read the measurements and sensitivity	[STX]85.0000 121[ETX]
START_Q2		CR	–	Start continuous Q2 output	ON[CR]ON[CR]OFF[CR]OFF…
STOP_Q2		–	–	Stop continuous Q2 output	[STX] > [ETX]
Q2 setting	Q2	R	–	Read Q2 output	[STX]ON[ETX]
	Q2_HI	R	–	Read actual setting of Q2 Hi	[STX]105.0000[ETX]
	Q2_LO	R	–	Read actual setting of Q2 Lo	[STX]65.0000[ETX]
	Q2_HI()60.000	W	–	Set Q2 Hi for example to 60mm *2	[STX] > [ETX] or [STX]?[ETX]
	Q2_LO()40.000	W	–	Set Q2 Lo for example to 40mm *2	[STX] > [ETX] or [STX]?[ETX]
	Q2 DEFAULT	R	●	Set Q2 to default (Self-diagnosis output)	[STX] > [ETX]
Avg. setting	AVG	R	–	Read setting of the response time	[STX]FAST[ETX]
	AVG()FAST	W	–	Set response time to Fast	[STX] > [ETX]
	AVG()MEDIUM	W	●	Set response time to Standard	[STX] > [ETX]
	AVG()SLOW	W	–	Set response time to High resolution	[STX] > [ETX]

이 자체가 Command가 됨
() 표기는 한칸 뛰는 것을 의미함

- 가장 우선으로 사용될 명령어 먼저 구현
- Command는 문자열 ASCII 형태로 주고 받음
- Measure의 단위는 mm

CONTENTS

EDDI
Electronic Design Development Institute

Serial통신을 이용한 Laser센서(cd33) 제어하기

Lasor 센서 코드 구현

```
typedef enum {
    UART_STATE_HEAD,
    UART_STATE_DATA,
} CD33_STATE;

typedef struct {
    unsigned char buf[BUF_SIZ];
    unsigned int index;
} CD33_DATA;
```

CD33 관련 정보 구조체 화

```
typedef struct _cd33_info {
    CD33_STATE state;
    CD33_DATA data;
    bool ready;
} CD33_INFO;
```

CD33 관련 정보 구성

```
void cd33_init(void)
{
    fd = cd33_open(dev_name, baud, 0, 1);
    if(fd < 0)
    {
        printf("serial device open failed\n");
    }

    cd33.state = UART_STATE_HEAD;
    cd33.ready = false;
    cd33.data.index = 0;

    memset(cd33.data.buf, 0, sizeof(BUF_SIZ));

    return;
}
```

CD33 관련 정보 초기화

```
int cd33_open(char *dev_name, int baud, int vtime, int vmin)
{
    struct termios oldtio = { 0 };
    struct termios newtio = { 0 };

    int fd = open(dev_name, O_RDWR | O_NOCTTY);
    if (fd < 0) {
        return -1;
    }

    tcgetattr(fd, &oldtio);

    newtio.c_cflag = CS8 | CLOCAL | CREAD; //No-rts/cts
    newtio.c_iflag = 0; // IGNPAR | ICRNL
    newtio.c_oflag = 0;
    newtio.c_lflag = 0; // ICANON
    newtio.c_cc[VTIME] = vtime; //timeout 0.1s
    newtio.c_cc[VMIN] = vmin; //wait until get vmin messages

    switch(baud)
    {
        case 115200: newtio.c_cflag |= B115200; break;
        case 57600: newtio.c_cflag |= B57600; break;
        case 38400: newtio.c_cflag |= B38400; break;
        case 19200: newtio.c_cflag |= B19200; break;
        case 9600: newtio.c_cflag |= B9600; break;
        case 4800: newtio.c_cflag |= B4800; break;
        case 2400: newtio.c_cflag |= B2400; break;
        default: newtio.c_cflag |= B115200; break;
    }

    tcflush(fd, TCIOFLUSH);
    tcsetattr(fd, TCSANOW, &newtio);

    //Set to non-blocking mode, this will be used when reading the serial port
    fcntl(fd, F_SETFL, O_NONBLOCK);

    return fd;
}
```

이미 존재하는 dev 파일 열기

현재 serial 세팅 가져오기

새로운 setting 설정

새로운 setting 적용

이 부분은 공부 필요

CONTENTS

Serial통신을 이용한 Laser센서(cd33) 제어하기

Lasor 센서 코드 구현
- 우선은 내 생각대로 코드 구현함
  (추후 다른 사람들이 짜는 방식들 참고하여 개선해 나갈 것)

```
void cd33_sendData(unsigned char *cmd)
{
    unsigned char sendBuf[BUF_SIZ] = { 0 };
    unsigned int index = 0;

    sendBuf[index++] = STX;          // 시작

    strcat(sendBuf, cmd);            // 입력받은 cmd
    index += strlen(cmd);

    sendBuf[index++] = ETX;          // 끝

    write(fd, sendBuf, index);       // 보내기

    // printf("%d\n", index);
    printf("%s\n", sendBuf);

    // cd33_recvData();
}
```

```
void cd33_recvData(void)
{
    unsigned char readBuf = 0;
    int ret = read(fd, &readBuf, 1);   // 1 Byte 단위 read
    if(ret > 0) {
        cd33_parse(readBuf);           // read 한 Byte가 있으면 Parsing 함수에서 명령어 구분
        // printf("check parse\n");
    }
}
```

시작과 끝이 STX, ETX가 아닌 경우도 존재

Example of Response
85.0000[CR]85.0001[CR]85.0···
[STX] > [ETX]
[STX] 85.0000 [ETX]

```
void cd33_parse(char data)
{
    switch (cd33.state)
    {
    case UART_STATE_HEAD:
        switch (data)
        {
            case STX:
                cd33.state = UART_STATE_DATA;
                cd33.data.index = 0;
                break;
            case CR:
                cd33.data.index = 0;
                cd33.ready = true;          // 필요한 response 준비되면 true
                break;
            default:
                cd33.data.buf[cd33.data.index++] = data;
                if(cd33.data.index == BUF_SIZ) {
                    cd33.data.index = 0;
                }
                break;
        }
        break;
    case UART_STATE_DATA:
        switch (data)
        {
            case ETX:
                cd33.state = UART_STATE_HEAD;
                cd33.ready = true;          // 필요한 response 준비되면 true
                // printf("check ETX\n");
                break;
            default:
                cd33.data.buf[cd33.data.index++] = data;
                if(cd33.data.index == BUF_SIZ) {
                    cd33.data.index = 0;
                }
                break;
        }
        break;
    }
}
```

실제 response 위치

실제 response 위치

- STX, ETX로 시작과 끝이 아닌 경
  하기 때문에 Parsing 함수 구현 시
  려해야 함