**Embedded System Software HW#3**

**Document**

**20131540**

**김준호**

**1. 목표**

**Module Programming, 디바이스 드라이버 구현, interrupt 등, 실습 시간 때 배운 내용을 활용하여 간단한 stopwatch 프로그램을 작성한다.**

**2. 구현**

**(1) fpga\_fnd device driver와, timer module, interrupt를 포함한 stopwatch 기능을 가진 module을 구현한다.**

(1)번의 구현 순서는 다음과 같았다.

1. interrupt handler 등록
2. timer 구현
3. interrupt handler 구현
4. fpga\_fnd device의 write 함수 구현

구현한 module은 실습시간에 제공된 interrupt.c 파일을 기반으로 만들었다.

먼저 interrupt 구현 부분이다.

우선 아래와 같이 stopwatch\_open 함수에서 irq값을 구한 후에 request\_irq 함수를 이용하여 interrupt handler를 등록한다. 아래는 stopwatch\_open 함수의 일부이다.

…

irq = gpio\_to\_irq(IMX\_GPIO\_NR(1,11));

…

ret=request\_irq(irq, inter\_handler1, IRQF\_TRIGGER\_RISING, "interrupt", NULL);

…

inter\_handler1은 해당 interrupt가 발생할 때 실행될 handler 함수이다.

IRQF\_TRIGGER\_RISING이라는 flag는 버튼이 눌렸다 올라왔을 때 interrupt가 발생하도록 설정하는 것이다.

마지막 parameter는 dev\_data로 interrupt handler에 data를 넘겨주고 싶을 때 사용하는 것이다. 본인은 이번 과제에서는 NULL로 셋팅하고 전역변수를 이용했다.

전역변수를 사용한 이유는 이번 과제에선 여러 개의 process가 같은 구현한 interrupt handler에 access할 일이 없이 때문이다.

다음으론 interrupt handler 함수들인데, 그 전에 timer에 대한 설명을 한다.

본 모듈은 2가지의 timer를 사용하는데 하나는 stopwatch 기능을 위한 timer, 다른 하나는 stopwatch 종료를 위한 timer이다.(3초 이상 VOL- 버튼을 누를 시 종료)

…

static struct struct\_mydata

{

struct timer\_list timer;

int min;

int sec;

int pflag;

unsigned long term;

};

struct struct\_mydata mydata;

static struct timer\_list exit\_timer;

void kernel\_timer\_func(unsigned long timeout);

void kernel\_exit\_timer\_func(unsigned long timeout);

ssize\_t kernel\_timer\_write(void);

…

mydata는 stopwatch 기능을 위한 timer이고 exit\_timer는 3초를 세기위한 timer이다. 먼저 mydata에 대한 설명이다.

min과 sec은 각각 분과 시간을 나타내고 pflag는 타이머가 paused 상태인지 아닌지를 나타내는 flag이다. term은 pasued될 때 타이머가 올라가기까지 남은 시간을 저장하기 위한 변수이다. kernel\_timer\_write함수는 기존의 동록되어 있던 timer를 delete하고 **term값을 기반**으로 expired 시간을 갱신하여 새로운 timer를 등록하는 함수이고 kernel\_timer\_func은 timer가 expired 될 때 호출되는 callback 함수이다. 이 함수에서는 mydata의 min과 sec값을 기반으로 iom\_fnd\_write 함수를 호출해서 fnd에 화면을 갱신시켜주고, 항상 expired 시간을 HZ 이후로 설정 후에 새로운 timer를 달아준다. (즉 paused 되었다가 다시 Back 버튼이 눌리면 remaining time 이후에 kernel\_timer\_func함수가 호출되고 그 이후의 expired 시간은 HZ 이후이다. ) kernel\_exit\_timer\_func는 exit\_timer 가 expired 되면 호출되는 callback 함수이며 기능은 후에 설명하겠다.

이제 다시 돌아가서 interrupt handler에 대한 설명이다. 먼저 Home 버튼이 눌렸을 때 호출되는 inter\_handler1이다.

이 handler의 기능은 단순하다. 현재 stopwatch가 paused상태가 아니라면 새로운 timer를 kernel\_timer\_write함수를 통해 달아준다.

다음은 Back 버튼이 눌렸을 때 호출되는 inter\_handler2이다. 여기선 우선 pflag를 검사한다. 즉 만약 paused상태가 아니라면 mydata의 term을 다음과 같이 변경한다.

mydata.term = mydata.timer.expries – get\_jiffies\_64();

이 식을 통해 term에는 남은 remaining time이 저장된다.

이후 현재 등록되어 있는 timer를 지워주고 pflag를 1로 셋팅한다.

만약 핸들러에 들어왔는데 paused 상태이면 kernel\_timer\_write 함수를 호출하여 term값을 기반으로 새로운 timer를 등록해주고 pflag를 0으로 바꿔준다.

다음은 VOL+ 버튼이 눌렸을 때 호출되는 inter\_handler3이다. 여기선 mydata의 min과 sec을 0으로 바꿔주고 fnd값을 갱신하며 (0분 0초) kernel\_timer\_write함수를 통해 새로운 timer를 달아준다.

마지막으로 VOL- 버튼이 눌렸을 때 호출되는 inter\_handler4이다. 먼저 이 handler는 위의 3개의 handler와 다르게 trigger flag가

IRQF\_TRIGGER\_FALLING | IRQF\_TRIGGER\_RISING

위와 같이 설정되어 등록되어 있다. 즉 버튼이 눌릴 때와 떼어질 때 호출된다.

이 handler에 interrupt context가 들어오면 먼저 버튼이 눌렸을 때인지 떼어질 때인지를 검사한다. 만약 눌렸을 때(gpio\_get\_value(IMX\_GPIO\_NR(5, 14))값이 0이다.)에는 exit\_timer를 새로 달아준다. 이 timer는 3초 후에 expired되며(exit\_timer.expires = get\_jiffies\_64() + 3 \* HZ) 그때 호출되는 call back 함수가 kernel\_exit\_timer\_func이다. 이 함수에서는 기존의 mydata.timer와 exit\_timer를 삭제하고 fnd를 0000으로 초기화하며

\_\_wake\_up(&waitQueue, 1, 1, NULL);

를 통해 user process를 wake up 시켜준다.

(app에선 device file를 open 후에 write함수를 호출하는데 구현한 module의 write함수에선

interruptible\_sleep\_on(&waitQueue);

를 통해 user process를 sleep 시키는 것 뿐이다. waitQueue는

DECLARE\_WAIT\_QUEUE\_HEAD(waitQueue);

와 같이 동적으로 선언하고 초기화 해주었다.)

다시 inter\_handler4로 돌아가서 만약 버튼이 떼어졌을 때(gpio\_get\_value(IMX\_GPIO\_NR(5, 14)) 값이 1이다.)에는 단순히 exit\_timer를 지워준다. 그러면 user process는 계속 sleep 상태일 것이고 mydata의 timer도 계속 돌 것이다.

마지막으로 iom\_fnd\_write함수이다. 이 함수는 min과 sec를 파라메터로 받아서 4 byte의 unsinged char에 정보를 저장하고 iomapping을 통해 fpga\_fnd device에 써준다.

**(2) module을 실행시키는 application을 구현한다.**

(2)번의 구현은 실습 때 주어진 intertest.c와 다 똑같고 open 하는 device file만

/dev/stopwatch로 다르다. 먼저 device file를 open 하고 write 함수를 호출하며(넘기는 parameter는 fd 이외엔 의미 없다.)이후엔 file을 close 해준다.

**3. 기능**

(1) fnd부분의 앞의 두자리는분(60분), 뒤의 두자리는 초(60초)를 표시한다.

(fnd의초기상태는0000)

**키입력은interrupt를이용하여수행한다.**

(2) Home 버튼 : start-> 1초마다 fnd의 정보를 갱신(timer 사용)

Back 버튼 : pause-> 일시정지**(소수점이하의 시간유지)**

VOL+ 버튼 : reset-> 초기상태로 돌아감

VOL-버튼 : 종료->**3초 이상 누르고 있을 시 어플리케이션을 종료(즉, user application은 해당 버튼을 눌러 종료될 때까지 끝나지 않음)하고, fnd를 0000으로 초기화 한다.**

(3) 디바이스 드라이버의 이름은**/dev/stopwatch**로 통일한다.**(major number : 242)**

(4) **응용프로그램**

**- 디바이스파일을 open한 뒤, write함수를 호출하여 device driver내에서 sleep, 종료버튼에 의해 깨어나서 돌아온 후, close하고 종료한다.**

**4. 기타**