**Embedded System Software 과제 3**

**(과제 수행 결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20171640, 박수진**

**개발기간: 2021. 05. 17. - 2021. 05. 30.**

**최 종 보 고 서**

**I. 개발 목표**

- 모듈을 통해 디바이스 드라이버를 구현하고 모듈 내에서 interrupt와 timer handling을 통해 board에서 stopwatch 기능을 구현한다. 이 때 interrupt는 top half와 bottom half로 나누어서 구현한다.

**II. 개발 범위 및 내용**

1. **개발 범위**
2. App.c

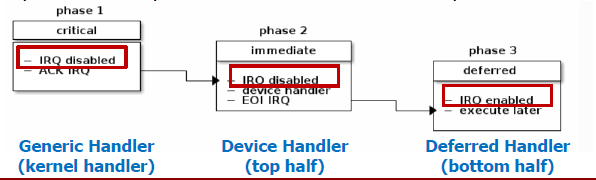
* Stopwatch device를 열어서 유저 프로그램 수행

1. Module.c

* File operation function 구현: open, write, release
* Timer 기능 구현
* Interrupt 구현: home, back, vol+, vol-
* fnd control

1. **개발 내용**

* **Top Half & Bottom Half Interrupt**



Top Half는 1) time-sensitive, 2) related to hardware, 3) ensure that another interrupt does not interrupt it 이 세가지에 속할 경우 Top Half 영역으로 작성한다. Bottom Half는 위 세가지에 포함되지 않을 경우 사용하며, Top Half와 Bottom Half를 나누어 작성하는 것이 좋다.

본 프로젝트에서는 Tasklets을 사용하여 Bottom half를 작성하였다. 구현해야 하는 Interrupt 기능은 크게 Start, Pause, Reset, Stop이 있는데, Stop을 제외한 나머지 3개는 즉각적으로 수행 되어야 하고 FND hardware를 직접 제어하기 때문에 Bottom half를 작성하지 않고 Top half로만 작성하였다. Stop 기능에서는 1) timer를 이용하여 3초가 만료된 후 2) wake up하고 프로그램을 종료하는 2가지 메커니즘으로 이루어져 있다. 이 때 timer는 time-sensitive하기 때문에 Top half로 작성하였고, wake up하고 프로그램을 종료하는 부분은 Top half에 속하는 3가지가 해당되지 않기 때문에 해당 부분을 Bottom half로 작성하고 top half에서 Bottom half를 호출한다. 이 때 top half의 timer가 expire되면 bottom half가 호출되도록 작성하였다. 또한 본인이 작성한 Bottom half의 경우 bottom half interrupt의 역할이 wake up과 프로그램 종료이기 때문에 함수 내에서 sleep 될 상황이 존재하지 않는다. 따라서 Work queue가 아닌 tasklets을 사용하였다.

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 기간 | 내용 |
| 05/17 ~ 05/19 | Application program 완성 |
| 05/20 ~ 05/23 | timer handler 구현 |
| 05/24 ~ 05/27 | Interrupt handler 구현 |
| 05/28 ~ 05/30 | FND 제어, 버그 수정 |

**나. 개발 방법**

1. App.c

* /dev/stopwatch를 open한 후, 파일을 write하고 (sleep) 끝나면 유저프로그램을 종료한다. 이 때 디바이스 파일을 close한다.

1. Module.c

* File operation function 구현: open, write, release

Open할 때 전역 변수 설정, 타이머 init, irq 설정을 수행한다.

Write할 때는 sleep on 될 수 있도록 한다.

Release 시에 타이머를 delete하고, irq를 free한다.

* Timer 기능 구현

Timer는 0.1초 간격으로 만료되도록 설정하고 1초에 한번씩 FND 값이 변화하도록 한다. (0.1sec 간격 => get\_jiffies\_64() + (HZ / 10))

또한 stop 시 3초 후에 끝내도록 하기 위해 새로운 timer를 설정한다.

이 새로운 timer는 3초 후(get\_jiffies\_64() + (3 \* HZ))에 만료되며 후에 별도의 timer add 과정이 필요없이 wake up 후 종료하면 된다.

두 타이머는 open 할 때 init하고, release할 때 delete한다.

* Interrupt 구현: home, back, vol+, vol-

Home, back, vol+ 는 즉각적인 반응이 필요하며 직접 디바이스를 제어하기 때문에 Top Half만 수행하도록 하였다.

Interrupt는 vol- falling 시 3초 후에 timer가 만료되면 Bottom Half를 수행하도록 작성하였다.

Bottom half interrupt에서는 프로그램을 wake up status로 바꾸고 프로그램을 종료한다.

* fnd control

FND는 io mapping 방식으로 직접 디바이스를 제어하였다.

**IV. 연구 결과**

**1) app.c (app directory)**

텍스트이(가) 표시된 사진

자동 생성된 설명

어플리케이션 프로그램은 위와 같이 간단하게 device open, device write, device close로만 구성하여 작성하였다. Device name은 dev/stopwatch로 설정하였다.

**2) stopwatch.c (module directory)**

**- File operations**

텍스트이(가) 표시된 사진

자동 생성된 설명

File operations은 실습자료의 코드를 그대로 사용하였다. Open, write, release에 대한 각각의 기능들을 구현하였다.

**Open**

텍스트, 실내, 닫기이(가) 표시된 사진

자동 생성된 설명

Open 할 때는 먼저 전역 변수들을 초기화한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그 다음 irq를 세팅한다. Home, Back, Vol+, Vol- 각각의 button에 알맞게 세팅하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

FND를 초기값인 0000으로 세팅하여 FND에 print하고, 해당 프로그램에서 사용될 두 타이머들을 init한다.

**Write**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Write는 유저 프로그램으로부터 call 된 후 바로 sleep 상태로 대기하도록 한다.

**Release**

텍스트이(가) 표시된 사진

자동 생성된 설명

Release 할 때 Open 시 할당한 irq를 모두 free한다. 또한 프로그램 종료 시 FND를 0000으로 초기화해야 하므로 보드에서 흐른 시간을 0으로 초기화한 후 FND에 0000을 출력한다. 모듈에서 사용되는 전역변수들을 프로그램 종료에 맞게 값을 초기화 하고, open할 때 init했던 타이머들을 다시 delete한다.

**- Timer**

timer에 사용된 struct는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Timer 기능을 위한 timer\_list가 있고, 0.1초씩 count하여 10번이 되면 초를 increase하기 위해 count 변수를 임의로 추가하였다. Stopwatch\_timer는 스탑워치 프로그램에 사용되는 timer이며, end\_timer는 프로그램이 stop interrupt를 받았을 때 3초간 지속되면 프로그램을 종료해야 하므로 이 3초를 세기 위한 timer이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Stopwatch timer는 home 버튼을 누르는 순간 시작된다. 각 타이머는 0.1초 (= HZ/10)마다 만료되며 만료될 때마다 timer를 update하는 함수를 부른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Stopwatch timer를 업데이트 함수는 위와 같다. 프로그램이 실행 중일 때 pause 상태가 아닌 경우 count를 1씩 증가한다. 이 때 count 값이 10이 되면 1초가 되었으므로 보드에 저장된 총 시간(now\_time)에 1을 증가한다. 만약 reset interrupt가 활성화된 상태이면 보드에 저장된 시간과 count된 0.1 단위 시간들을 모두 0으로 초기화한다. Now\_time 변수가 확정되면 FND에 그 값을 출력한다. 그리고 그 다음 timer를 같은 방식으로 add한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

End timer는 stop interrupt가 활성화 되었을 때 3초를 세기 위한 timer이다. 3초 간격으로 계속 셀 필요는 없으므로 한번만 add된다. 만약 3초를 채우지 못하고 상태가 Rising이 되면 add 했던 타이머를 다시 delete한다. 3초는 3\*HZ와 같으며 만료되면 timer를 clear하기 위한 함수를 호출한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Terminate flag를 1로 set하여 프로그램의 종료를 다른 함수들에게 알리고, stop이 되면 FND를 0000으로 세팅해야하므로 0000을 FND에 출력하도록 한다. 이 때 FND를 제어하는 부분은 hardware를 제어하기 때문에 Top Half Interrupt에서 관리하고 sleep된 상태를 wake up으로 바꾸기 위해 Bottom Half Interrupt를 호출한다.

**- Interrupt**

**Start**

텍스트이(가) 표시된 사진

자동 생성된 설명

Home button이 Falling되면 start가 시작된다. 다른 기능이 활성화되지 않은 채로 start를 누르면 타이머가 시작되었음을 알리기 위한 전역변수들을 세팅하고 timer를 세팅하여 add한다. 만약 이미 start가 활성화된 상태라면 start가 새로 기능하지 못하게 하였고, pause 또는 reset 상태에서 start가 호출된 경우에는 중단된 stopwatch가 재시작 되도록 하였다.

**Pause**

텍스트이(가) 표시된 사진

자동 생성된 설명

Pause는 잠시 중단되었다는 flag만 set하고 timer 함수에서 이 flag를 인식하여 pause에 해당하는 명령을 수행한다. Pause가 되면 0.1초마다 증가하던 timer 함수의 count increase가 이전 상태에서 멈추고 이로 인해 stopwatch의 초가 증가하지 않게 된다. Pause가 풀리면 멈춰 있던 timer의 count변수가 이전 값을 그대로 유지한 상태에서 다시 count를 increase하기 시작한다. stopwatch가 count 4(즉 0.4초)에서 pause 됐다면 재시작 후에는 count 4를 유지한 채로 0.1초마다 증가하기 시작하여 재시작 0.1초 후에는 count 값이 5가 된다. 이 count 값이 10이 되면 1초가 증가한다. 만약 stopwatch가 아예 시작된 상태가 아니고 이미 pause가 된 상태라면 아무 변화 없도록 한다.

**Reset**

텍스트이(가) 표시된 사진

자동 생성된 설명

reset에서도 마찬가지로 flag만 set하고 timer 함수에서 이를 확인하여 알맞을 기능을 수행한다. Reset이 불리면 stopwatch에서 count된 시간이 0으로 초기화되고, timer의 count도 0으로 초기화된 채로 pause상태가 된다. 즉, reset이 호출되면 보드에 누적된 시간 0, 0.1초 단위의 count도 0인 상태로 다시 start 버튼을 눌려 재시작하기 전까지 멈춘다. 보드에서 보기에는 프로그램이 호출되었을 때와 같은 상태 (FND 0000)으로 유지되지만 프로그램 내에서는 0.1초마다 계속 타이머가 호출된다. 그러나 FND 값의 변화는 없다.

**Stop**

텍스트이(가) 표시된 사진

자동 생성된 설명

Stop에서는 Top half interrupt와 Bottom half interrupt를 나누어 구현하였다. 우선 3초를 세는 타이머는 time-sensitive 하므로 top half interrupt이며 만약 3초 이전에 Vol- 버튼에서 손을 떼면 add했던 타이머를 다시 delete한다. 만약 3초 이상 누르고 있으면 end timer가 expire되고 함수가 호출된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이 함수에서는 프로그램이 종료되었음을 알리기 위해 terminate flag를 set하고 종료 전 FND를 0000으로 초기화한다. Device를 제어하는 부분이므로 top half interrupt로 분류하였다. Top half interrupt에서 bottom half interrupt를 위해 tasktlet\_schedule을 호출한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Bottom half interrupt에서 호출된 함수는 sleep 상태에서 wake up 상태로 바꾸고 프로그램이 정상적으로 종료되도록 한다. 이 때 함수 내에서 sleep하는 부분이 없으므로 workqueue가 아닌 Tasklet으로 간단하게 구현하였다.



Tasklet은 위와 같이 DECLARE\_TASKLET macro를 사용하여 정의하였다.

**- FND control**





FND는 IO 디바이스의 주소를 알고 이를 mapping하여 접근하는 방식을 취하였다. 모듈을 init할 때 mapping이 되도록 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

FND device에 출력할 때 보드에 기록된 총 seconds를 minutes과 seconds로 나눈 후 차례대로 FND device에 출력하였다. 출력 시 outw 함수를 사용하며, FND address를 이용하여 접근한다.



Mapping된 IO device는 모듈 exit 시 다시 unmap된다.

**V. 기타**

- 인터럽트는 저학년 때부터 자주 들어오던 개념이었는데 제대로 배운 건 거의 처음인 것 같습니다. 인터럽트를 강의만 들었을 때는 조금 어려웠는데 직접 해보니까 생각보다 좀 더 어려웠고 하고 나니까 이해가 되는 것 같았습니다. 그러나 인터럽트의 옵션을 잘못 쓰고 계속 오류를 못 찾는 바람에 과제가 지각 제출이 되어 조금 아쉽습니다.