Embedded System Software 과제 3

(과제 수행 결과 보고서)

과목명: [CSE4116] 임베디드시스템소프트웨어

담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 20171667, 이용욱

개발기간: 2020. 06. 04. -2020. 06. 05.

**최 종 보 고 서**

**I. 개발 목표**

본 프로젝트의 목표는 실습시간에 배운 Module programming, 디바이스 드라이버 , interrupt 등을

활용하여 간단한 stopwatch 개발하는 것이다.

**II. 개발 범위 및 내용**

1. **개발 범위**
2. 테스트 응용 프로그램

사용자가 실행하면 모듈을 실행시키는 역할을 한다.

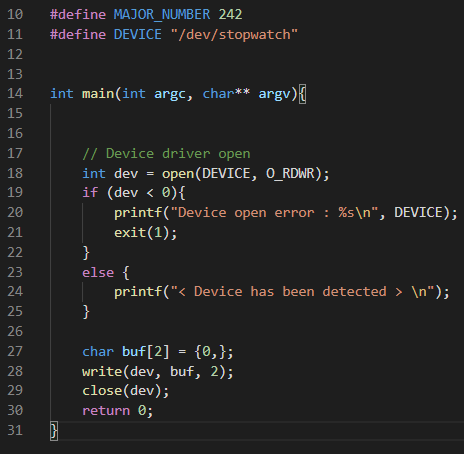
1. Stopwatch 모듈

Device driver를 활용하여 fnd display에 시각을 출력한다. 이때 이 시각은 timer를 통해 매 초 갱신하며, interrupt를 통해 시작, 정지, 리셋, 종료 등의 동작을 할 수 있도록 한다.

**나. 개발 내용**

1. 테스트 응용 프로그램 (app.c)

Device의 open, write, close로 이루어지는 간단한 프로그램이다.



Open을 통해서 stopwatch 모듈을 시작시킨다.

이후 write를 하는데 이는 어플리케이션이 타이머가 동작하는 동안 sleep하기 위함이다.

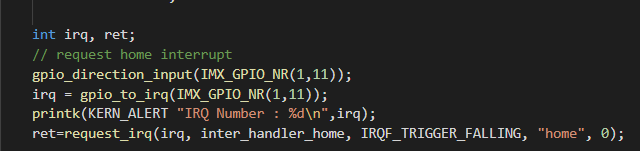
Write를 하면 모듈에서 interruptible\_sleep\_on 함수를 통해 stopwatch의 동작이 끝날 때 까지 유저 프로세스를 sleep 상태로 만든다.

stopwatch의 동작이 끝나면, close를 실행한다.

1. Stopwatch 모듈
   1. 인터럽트 등록

이번 과제에서는 인터럽트를 사용하였다.

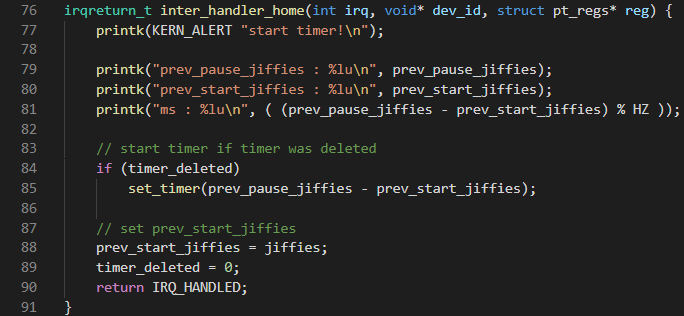
모듈의 .open 함수로 등록된 iom\_open함수에서 각 general propose input output device에 대해 interrupt handler를 등록하였다. 그 코드는 아래와 같다.



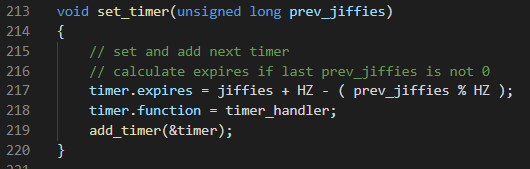
인터럽트 핸들러로는 inter\_handler\_home, inter\_handler\_back, inter\_handler\_volup, inter\_handler\_voldown가 있다. 각 버튼이 눌렸을 때(fall)의 인터럽트를 제어하는 함수이다. 단, voldown키의 경우에는 rise와 fall을 모두 감지한다.

* 1. inter\_handler\_home

inter\_handler\_home 함수에서는 타이머를 시작하는 역할을 한다. 이때 timer가 처음부터 시작하는 것이 아니라, back키를 눌러 멈춘 상태에서 시작하는 경우에는 소수점 이하의 시간을 유지하여야 한다. 이를 위해 prev\_pause\_jiffies, prev\_start\_jiffies 변수를 사용하였다. 이 변수들은 각각 마지막으로 pause, start 했을 때의 jiffies를 저장한다. 둘을 빼면 마지막으로 타이머가 동작한 기간을 구할 수 있다.



이를 set\_timer함수에 넘겨주면, set\_timer함수가 소수점 이하의 시간을 구하고 이를 고려하여 timer.expires를 구한다.



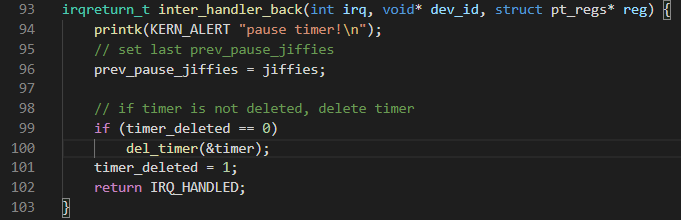
예를 들어 0초에 타이머가 시작하여 3.5초에 동작이 끝났다면

prev\_start\_jiffies = 0, prev\_pause\_jiffies = 350 (HZ=100) 이다

이 둘을 빼고 HZ로 나눈 나머지를 구하면 소수점 이하의 시간 50(0.5sec) 을 구할 수 있다. 이만큼을 jiffies + HZ 에서 빼서 timer.expires를 구하면 이 timer는 0.5초 후에 만료된다. 소수점 이하의 시간(0.5초)을 유지한 것이다.

* 1. Inter\_handler\_back

Back 키를 눌렀을 때, 타이머를 멈추는 역할을 한다. 타이머를 멈추기 위해 del\_timer함수를 호출한다. 이때 이미 타이머가 삭제된 경우의 에러를 방지하기 위해서 timer\_deleted 라는 flag변수가 사용된다. 타이머가 이미 삭제된 상태라면 1(True)값을 가진다. 그리고 prev\_pause\_jiffies 변수를 갱신한다.



* 1. Volup

이 함수도 타이머를 멈추기 위해 del\_timer함수를 호출한다. 다른 점은 스톱워치를 reset하기 위하여 변수들을 초기화한다는 점이다. 그리고 fnd를 갱신하기 위해서 fnd\_write를 호출한다.

* 1. Voldown

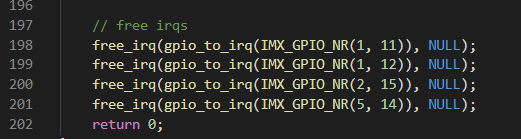
3초 이상 누를 경우 프로그램을 종료하는 함수이다. Gpio\_get\_value함수를 통해 VOL-키의 값을 가져온다. 실험 결과 누를 때 0, 올라올 때 1을 가짐을 알게 되었다.

누를 때 prev\_voldown\_jiffies 변수를 갱신한다. 마지막으로 VOL-키가 눌린 jiffies를 저장하는 변수이다.

버튼이 때질 때 현제 jiffies와 prev\_voldown\_jiffies를 비교하여 3초 이상이 흘렀다면 app 프로그램을 깨운다. 그러면 app에서 close를 호출하여 stopwatch를 종료한다.

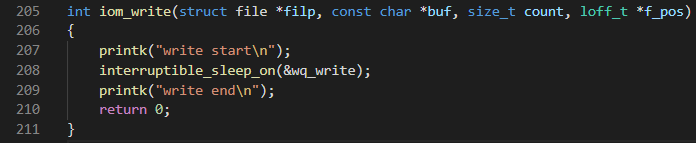
* 1. Release

사용한 irq넘버는 시스탬의 자원이므로 해제해주어야 한다. 이를 iom\_release함수에서 실행한다.



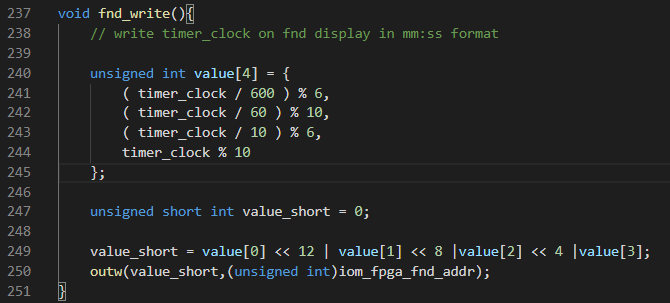
* 1. Iom\_write

User 프로그램에서 write를 call하는 경우 user프로그램을 stopwatch가 끝날 때 까지 sleep시킨다.



* 1. Fnd write

Timer\_clock을 분, 초로 나누고 십의자리와 일의 자리로 나누어 fnd에 출력하는 역할을 한다. 다음과 같이 계산하여 구하였다.



**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

6월 4일 : 과제 파악, 어플리케이션 구현

6월 5일 : timer와 각종 device의 동작 구현, 테스트, 보고서 작성, 주석 작성

**나. 개발 방법**

Virtualbox로 Ubuntu 가상환경을 가져오고 이 환경에서 소스코드를 크로스컴파일 하였다.

컴파일 된 app 파일과 driver.ko 파일을 ADB를 통해 임베디드 기기로 전송하였다.

minicom을 통해 모듈을 install 하고 app을 실행시킨다.

보드의 입출력 장치를 통해 기능이 정상적으로 작동함을 확인하였다.

**IV. 연구 결과**

명세서에서 요구한 사항이 모두 잘 동작하였다.

**V. 기타**

이번 프로젝트를 계기로 인터럽트가 어떻게 동작하는지, 리눅스 커널에 대해서 더 자세히 알게 되어서 좋았다.

또한 저번 프로젝트는 구현할 요소가 너무 많아 시간이 부족했는데, 이번 프로젝트는 딱 알맞은 정도의 구현 과제가 있어서 좋았다.