**Embedded System Software**

**HW #3**

**STOPWATCH**

**(설계 프로젝트 수행 결과)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20141526, 박영후**

**개발기간: 2019. 05. 24. -2019. 05. 29.**

**I. 개발 목표**

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

**II. 개발 범위 및 내용**

- 자신들이 설계한 개발 목표를 달성하기 위하여 어떠한 내용의 개발을 수행할 지 그 범위와 개발 내용을 기술할 것.

**(1) User Program**

**(2) Module**

**가. 개발 범위 및 개발 내용**

**(1) User Program**

device file을 open하고, write함수를 호출하여 module을 실행시킨다. module내에서 sleep되고, “VOL-버튼을 3초이상 누르고 있을 시” 라는 종료조건에 의해 깨어나서 돌아온 후에, open했던 device file을 close하고 종료한다.

**(2) Module**

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

* **fpga\_fnd device driver**
* fnd부분의 앞의 두 자리는 분(60분), 뒤의 두 자리는 초(60초)를 표시한다.
* **interrupt**
* Home BUTTON : start -> 1초마다 fnd의 정보를 갱신한다.
* Back BUTTON : pause -> 일시 정지(단, 소수점 이하의 시간은 유지한다)
* VOL+BUTTON : reset -> 초기상태로 돌아간다. 초기상태라 함은 fnd에 출력되는 시간의 값을 0000으로 초기화 해줌을 뜻한다.
* VOL- BUTTON : 종료 -> 3초 이상 버튼을 누르고 있을 시에 프로그램을 종료한다. 프로그램이 종료될 때에 fnd의 값을 0000으로 초기화 시켜준다.
* **timer module**
* 시간이 지남에 따라 fnd의 값이 update되는 부분을 timer module을 이용하여 구현한다. 또한 end\_timer라는 하나의 타이머를 더 추가하여 종료조건을 확인하여 만족될 시 프로그램을 종료해준다.

**III. 추진 일정 및 개발 방법**

1. **추진 일정**

05.24 : 프로젝트 요구사항 분석 및 interrupt관련 실습자료 공부, module programming관련 실습자료 공부.

05.25 : 유저프로그램 및 모듈 코드 구현

05.26 : 공지된 질문 및 요구사항에 맞게 코드 수정

05.29 : 주석 추가 및 코드 최종 점검, 보고서 작성

**나. 개발 방법**

강의자료와 그동안의 실습자료 및 실습 시 작성했던 코드들을 통해 모듈프로그래밍과 인터럽트 Handler를 구현하는 부분을 공부하여 프로젝트를 진행하였다.

특히 fpga\_fnd device를 control하는 부분, timer module을 이용하는 부분, interrupt handler를 구현하는 부분을 많이 참고하여 구현하였다.

**fpga\_fnd device –** HW#2코드를 참고하여 모듈이 실행될 때 주소를 mapping하여 outw함수를 이용해 fnd에 데이터를 출력해주었다.

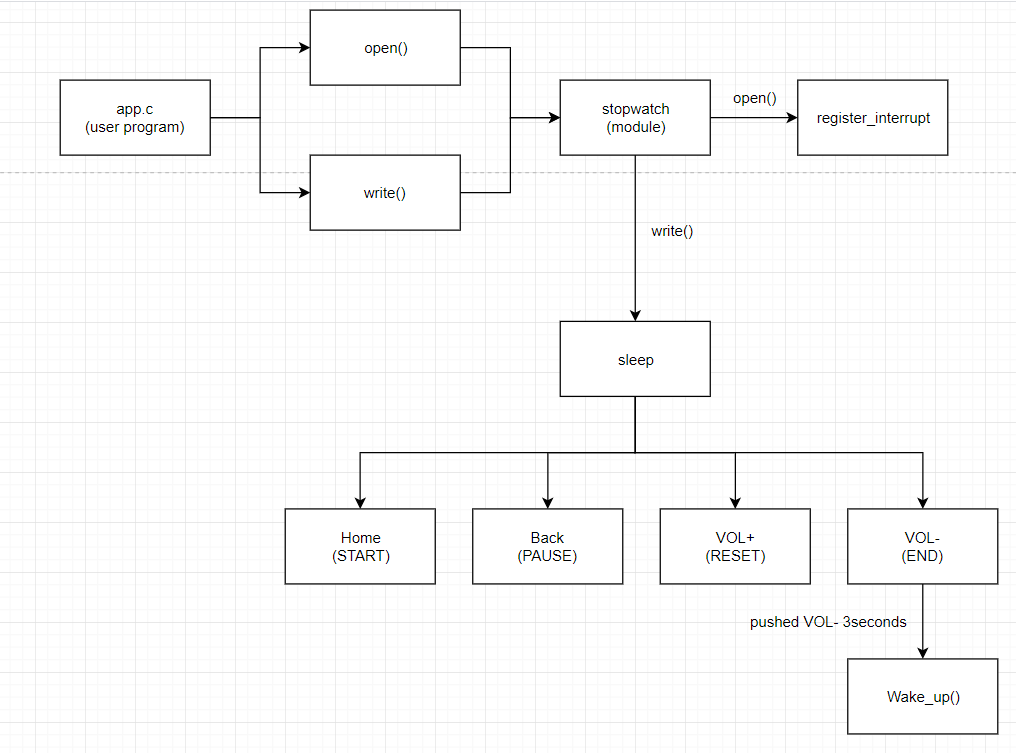
**timer module –** Home button이 눌려 스탑워치가 실행되면 fnd의 값이 1초마다 1씩 증가하는 부분을 하나의 timer를 이용하여 구현하였고, VOL- button이 3초이상 눌려서 프로그램이 종료되는 조건을 또다른 end\_timer라는 하나의 timer를 이용하여 구현하였다.

**interrupt handler –** 실습시간에 구현했던 예제코드를 수정하여, 각 button에 따른 interrupt handler를 구현하였다.

**IV. 연구 결과**

- 최종 연구 개발 결과를 자유롭게 기술할 것.

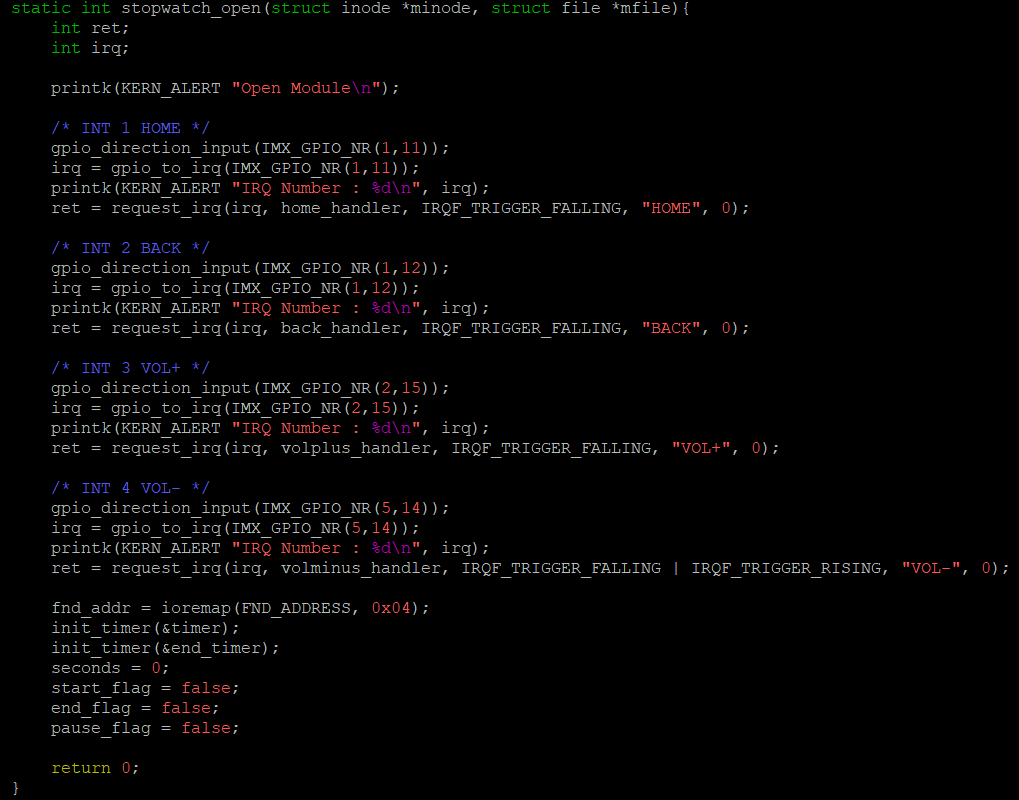
**1. 합성 내용:**



**2. 제작 내용:** 개발 결과

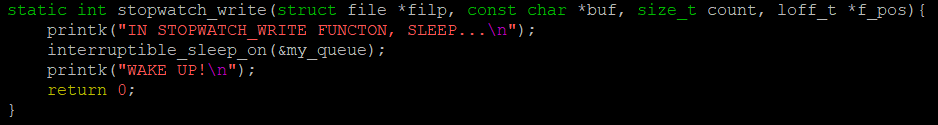
**\* FUNCTIONS**

**NAME : stopwatch\_open(struct inode \*minode, struct file \*mfile)**



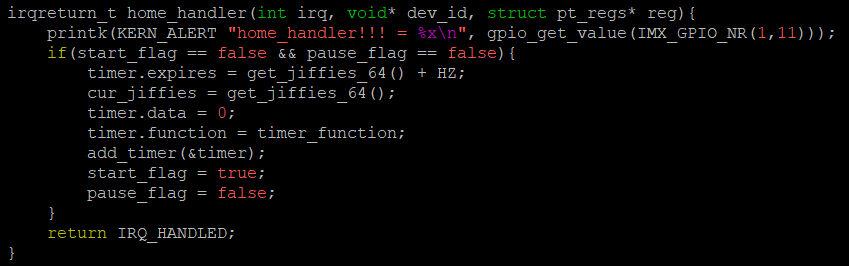
**TASK : 특정 interrupt에 대해 irq number를 얻은 후, 그 irq number에 대해 handling해주는Interrupt handler들을 등록한다.**

**NAME : stopwatch\_write(struct file \*filp, const char \*buf, size\_t count, loff\_t \*f\_pos)**



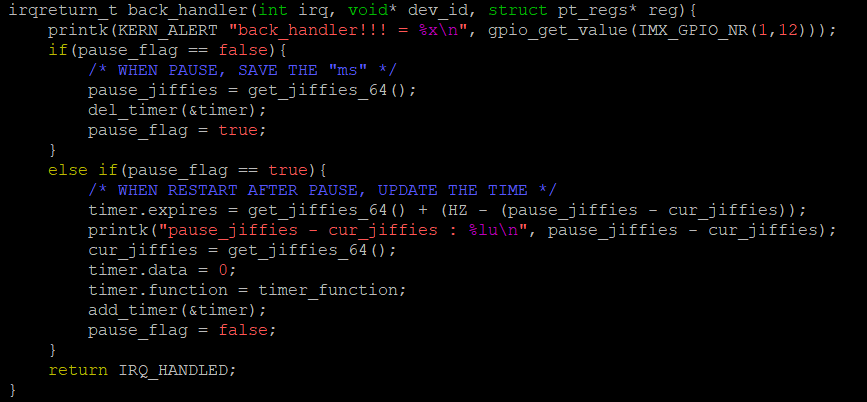
**TASK : process를 sleep시키고 특정한 조건을 통해 wake\_up 되기 전까지 기다린다.**

**NAME : home\_handler(int irq, void\* dev\_id, struct pt\_regs\* reg)**



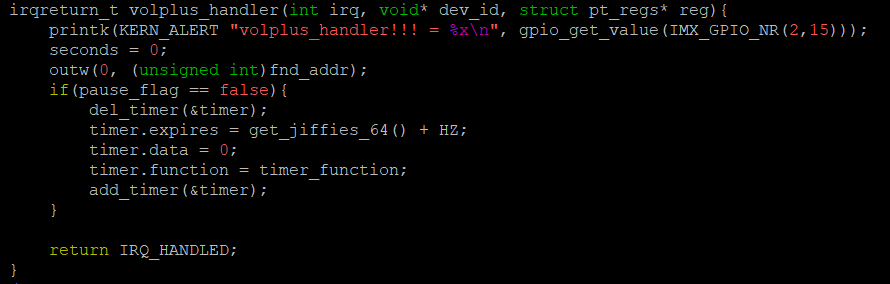
**TASK : 보드 상의 Home button이 눌렀을 경우에 해당하는 interrupt를 handling하는 interrupt handler 코드이다. 여기서는 두개의 flag 값을 확인하여 조건이 충족되면 stopwatch를 start시켜주는 기능을 한다. 즉, timer module을 이용하여 add\_timer를 통해 timer를 등록하고 일정 시간이 지나면 timer\_function을 호출하여 timer\_function내부에서 fnd값이 지속적으로 업데이트 되는 형식이다.**

**NAME : back\_handler(int irq, void\* dev\_id, struct pt\_regs\* reg)**



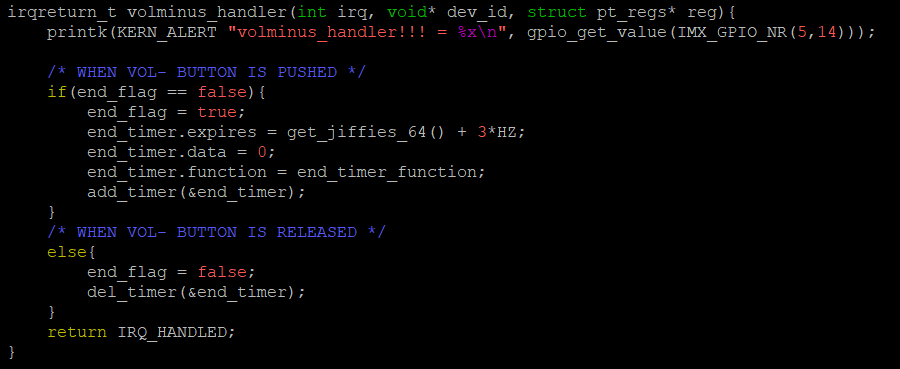
**TASK : 보드 상의 Back button이 눌렸을 경우에 해당하는 interrupt를 handling하는 interrupt handler 코드이다. pause\_flag가 false일 경우, 즉 stopwatch가 실행중인 경우에는 그 시점의 jiffies값을 pause\_jiffies라는 변수에 저장시켜 놓고, 타이머를 제거하여 fnd값의 업데이트를 멈추게 한다. pause가 된 상태에서 Back버튼의 입력이 한번 더 들어올 경우가 바로 pause\_flag == true인 경우인데, 이 때에는 기존에 기억해두었던 pause\_jiffies의 값을 이용하여 남은 시간 이후에 timer function을 호출하여 fnd값을 다시 업데이트(재실행)하는 코드를 구현한 부분이다. 예를들어 첫 Back 버튼의 입력이 0.3초가 지난 후에 들어왔다면, 다시 Back버튼을 눌렀을 때에는 0.7초뒤에 fnd의 값이 1 증가한다.**

**NAME : volplus\_handler(int irq, void\* dev\_id, struct pt\_regs\* reg)**



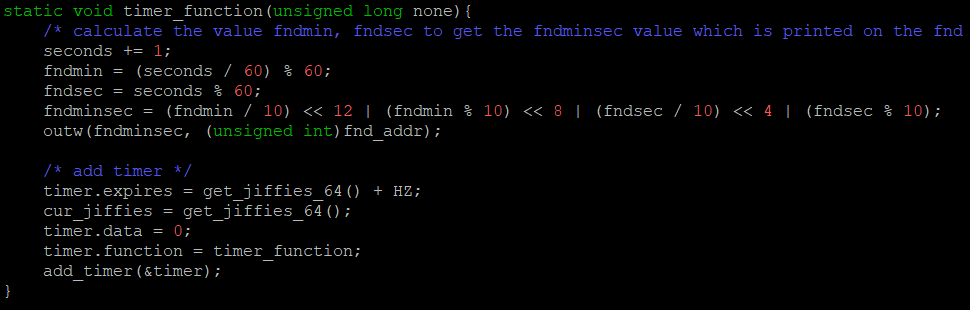
**TASK : 보드 상의 VOL+ button이 눌렸을 경우에 해당하는 interrupt를 handling하는 interrupt handler 코드이다. 이 코드는 fnd에 출력되는 값을 0으로 업데이트 해주고, 현재 진행되는 시간의 정보를 저장하고있는 seconds변수의 값을 0으로 초기화 시켜준다. pause\_flag가 false라는건, 현재 stopwatch가 진행중일 때 초기화를 시켜줬다는 얘기이므로, 이 경우, timer\_function을 다시 호출해줘서 fnd가 0부터 다시 지속적으로 1초마다 1씩 증가하도록 해준다**

**NAME : volminus\_handler(int irq, void\* dev\_id, struct pt\_regs\* reg)**



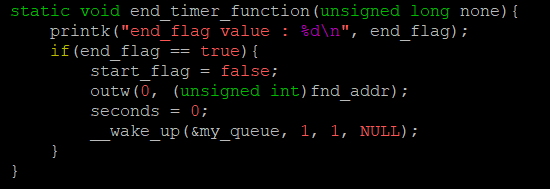
**TASK : 보드 상의 VOL- button이 눌렸을 경우에 해당하는 interrupt를 handling하는 interrupt handler 코드이다. end\_flag가 false일 경우에는 VOL-버튼이 눌렸을 경우를 말하는데, 이때 end\_flag값을 true로 업데이트 해주고, 3초뒤에 end\_timer\_function을 호출하도록 하는 end\_timer를 add\_timer함수를 이용하여 등록한다. 만약 3초가 지나기 전에 VOL-버튼이 release(떼어지면)되면, 이 interrupt handler코드가 다시 실행될 것이고, 그때의 end\_flag는 true값이였을 것이기 때문에, else에 해당하는 부분에 들어가고, 그 경우 end\_flag는 flag로 해주고 end\_timer를 제거해줌으로써 end\_timer\_function함수의 호출을 방지한다. 만약 3초가 지나도록 손가락을 떼지 않고 계속 누르고있었다면 end\_timer\_function이 호출되어 이 모듈 프로그램이 wake\_up될 것이고, 이는 유저프로그램의 종료를 의미한다.**

**NAME : timer\_function(unsigned long none)**



**TASK : stopwatch의 전체적인 흐름을 담당하는 timer\_function이다. 즉 스탑워치가 pause가 아닌 상태, 실행중인 상태인 경우 1초에 한번씩 호출되어 fnd값을 업데이트 해주는 timer function이다. 적절한 계산을 통해 fndminsec변수에 분과 초에 대한 시간정보를 저장해둔 후 이 값을 fnd에 출력해준다.**

**NAME : end\_timer\_function(unsigned long none)**



**TASK : VOL-버튼이 3초이상 눌리고 있었을 경우에 호출되는 timer function이다. 이 함수에서는 fnd의 출력값을 0으로 초기화 시켜주고 \_\_wake\_up함수를 호출하여 sleep되어 wait하고 있던 프로세스를 깨워준다.**

**3. 시험 및 평가 내용:**

- 평가 방법에 대한 설명을 기술하라.(수행 sequence 등)

- 자신들의 결과물이 갖는 **보건 및 안정**, **생산성과 내구성**에 대하여 반드시 기술할 것.

insmod를 통해 모듈을 등록하고, mknod로 디바이스 파일을 생성한 뒤, HOST PC에서 cross compile한 app(실행파일)을 minicom 환경에서 실행시키면 프로그램이 종료되지 않고 슬립된 상태로 대기한다.

이후 각 button을 누르면 그에 해당하는 interrupt handler가 호출되어 기능을 수행한다.

요구사항에서 요구한대로, Home버튼을 누를 경우 stopwatch가 start되어 1초마다 fnd의 정보를 갱신하고, Back버튼을 누를 경우 stopwatch가 일시정지 된다. 즉 fnd의 값이 업데이트 되지 않는다. 이후 다시 실행시키고 싶을 경우 Back버튼을 한번 눌러주면 fnd의 값이 다시 업데이트 되기 시작한다. 이때 만약 처음 Back버튼이 눌렸을 때 0.3초가 흐른 상태에서 Back버튼을 눌렀었다면, 두번째 재실행을 위한 Back버튼을 눌렀을 때에는 0.7초가 지난 뒤 fnd값이 업데이트 된다. 또한 stopwatch가 실행된 상태에서 VOL+버튼을 눌러주면 fnd의 값이 0000으로 바뀌고 계속해서 실행된 상태를 유지한다.(즉, 0000에서부터 다시 1초마다 1씩 증가한다). 만약 pause가 된 상태에서 VOL+버튼을 눌렀다면, fnd값은 0000으로 초기화되고 pause된 상태를 유지한다. 또한 VOL-버튼을 3초이상 누르고 있을 시, stopwatch가 실행중인 상태이거나 pause인 상태이거나 상관없이 wakeup되어 프로그램이 종료된다.

프로그램을 두번째 재실행 했을 경우에도 첫번째 실행과 같이 문제 없이 동작함을 확인할 수 있다. 여러가지 예외 케이스를 넣어 봤음에도 문제없이 요구사항에서 요구한 대로 안정성이 보장되는 프로그램을 작성하였다.

**V. 기타**

- 기타 관련 내용을 기술할 것.

**1.** **연구 조원 기여도**: 박영후(100%)

**2.** 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

1번 프로젝트를 진행했을 때 보다는 2번이, 2번 프로젝트를 진행했을 때 보다는 이번 프로젝트가 보다 수월했던 것 같다. 물론 요구사항 자체가 보다 무난했던 점도 있었겠지만, 디바이스를 다루는 코드라던가 모듈 프로그래밍을 하는 과정에서 코드를 해석하고 작성하는 부분에서 전보다 훨씬 수월하게 진행하고 있는 내 자신을 발견할 수 있었다. 특히 interrupt handler를 작성하는 코드를 처음해보는 것 같은데, 새로운 분야에 대해서도 이전의 프로젝트들의 영향 때문인지 수월하게 진행할 수 있었고, 2번 프로젝트를 진행했을 때 보다 훨씬 timer module을 능수능란하게 다룰 수 있었던 내 자신을 보고 이 과목을 통해 많은 것을 배우고 있다는 사실을 느꼈다. 앞으로 남은 프로젝트와 마지막 과제까지도 열심히 해서 많은 것을 배우고 이번학기를 마치겠다는 생각을 했다.