**Embedded 3st HW**

**(설계 프로젝트 수행 결과)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20121600 송제호**

**개발기간: 2017. 05. 23. -2015. 05. 26**

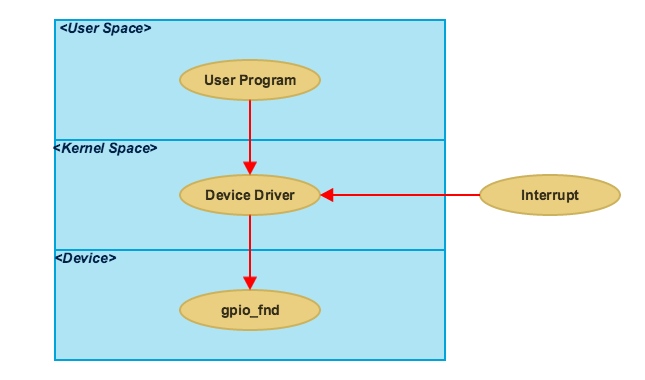
**최 종 보 고 서**

**I. 개발 목표**

Module programming, 디바이스 드라이버 구현, interrupt를 활용하여 간단한 stopwatch 프로그램을 작성한다.

**II. 개발 범위 및 내용**

**가. 개발 범위**



1. gpio\_fnd device driver, timer module 그리고 interrupt를 포함한 stopwatch 기능을 가진

module을 구현한다.

2. module을 실행시키는 app을 구현한다.

**나. 개발 내용**

1. fnd 부분의 앞의 두 자리는 분(60분), 뒤의 두 자리는 초(60초)를 표시한다.

(fnd의 초기 상태는 0000) 키 입력은 interrupt를 이용하여 수행한다.

2. gpio\_home : start -> 1초마다 fnd의 정보를 갱신 (timer 사용)

Gpio\_back : pause -> 일시정지

Gpio\_volume\_up : reset -> 초기상태로 돌아감.

Gpio\_volume\_down : 종료 -> 3초 이상 누르고 있을 시 어플리케이션을 종료하고, fnd를 꺼준다.

3. 디바이스 드라이버의 이름은 /dev/stopwatch로 통일한다. (major number : 242)

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

2017.05.23 : request\_irq를 공부하고 실습내용 복습.

2017.05.24~25 : 디바이스 드라이버, 타이머 그리고 interrupt 구현.

2017.05.26 : 테스트 및 결과보고서 작성.

**나. 개발 방법**

응용프로그램에서는 device driver를 open해주고, write를 해주는 역할을 할 뿐, device driver에서 timer interrupt와 gpio 버튼을 누를 때의 IO interrupt를 구현한다.

Timer의 경우 2개의 timer로 하나는 stop watch 역할을 하는 timer를, 다른 하나는 종료시 3초 이상 누르는 지 확인하는 timer를 구현한다.

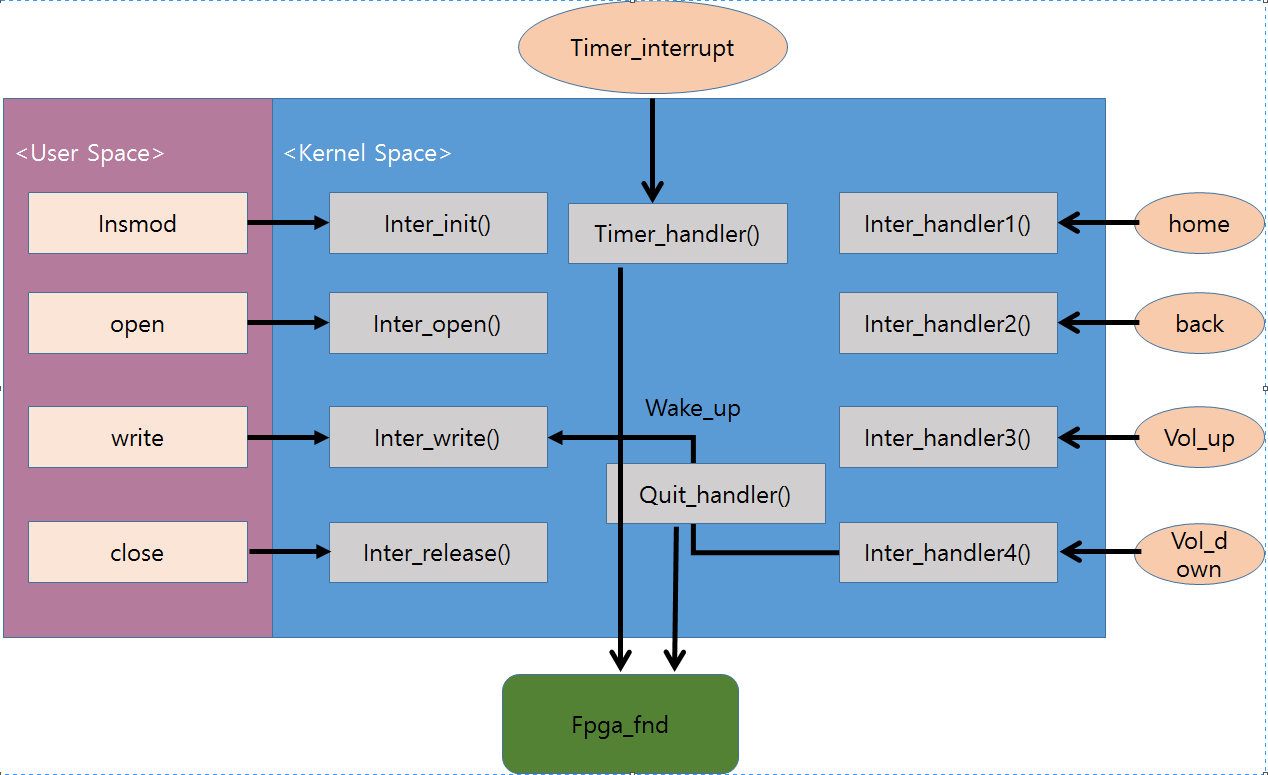
Gpio\_fnd의 경우 현재 timer로 계산된 시간을 분, 초로 출력할 수 있도록 구현한다.

Interrupt의 경우 gpio input button중에 home, back, volume up 그리고 volume down을 누를 때 interrupt로 받아 각각 해당하는 역할을 수행할 수 있도록 구현한다.

**IV. 연구 결과**

**1. 합성 내용**

**<전체 프로그램 구성도>**.



**2. 제작 내용**

**1) app.c (응용프로그램)**

응용프로그램에서는 stopwatch device driver를 통해서 open, write 그리고 close 한다.

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

**주요 전역 변수**

1. **struct timer\_list my\_timer**

timer module로 device driver에서 write함수 호출 시 1초간격으로 timer\_handler함수를 호출한다. Stop watch기능을 구현하기 위한 목적으로 쓰인다.

1. **struct timer\_list quit\_timer**

timer module로 SW6 interrupt시 falling되는 순간 3초 후에 quit\_timer\_handler함수를 호출한다. 종료 기능을 구현하기 위한 목적으로 쓰인다.

1. **static int current\_min,current\_sec**

stop watch기능 중 시간을 나타내는 변수로 fnd화면에 띄울 시간을 나타낸다.

(current\_min<<4 + current\_sec)

**주요 함수**

1. **inter\_write**

user program에서 호출되는 함수로 첫번쨰로 fnd화면에 0을 출력해주고 sleep을 시켜놓는다. 그래서 다른 interrupt handler에서 wake\_up을 해주면 다시 화면에 0을 출력해주고 return을 하여 응용프로그램에서 바로 close를 하도록하여 다른 button input을 무시하도록 해준다.

1. **Timer\_handler**

1초마다 timer interrupt가 불러주는 함수로, current\_sec값을 1 증가시켜주고 초와 분을 각각 modula 60을 취해준다. 그리고 다시 add\_timer와 expire값을 초기화해주는 작업을 해준다.

1. **inter\_handler1-4**

각 gpio input에 따른 interrupt handler다.

1. start : add\_timer를 통해서 timer\_handler가 수행되도록 해주고 이전 jiffies값을 가져와서 expire값을 다시 설정해준다.(ex. 0.5초 후에 pause후에 다시 start 하면 0.5초 후에 interrupt가 일어나도록 설정)
2. pause: del\_timer를 통해서 timer\_handler가 수행되지 않도록 해주고 현재 jiffies값을 전역변수에 저장해 놓는다.
3. Reset: expire값을 다시 현재 jiffies값 + HZ로 설정해주고 current\_min과 current\_sec을 0으로 초기화해주고 화면에 0을 띄워준다.
4. Quit: 누르는 순간, quit\_timer에 quit\_handler를 add해주고 expire값을 jiffies\_64+ HZ\*3으로 설정해주어 초후에 quit\_handler가 실행되도록 한다. 그러나 3초 전에 rising input이 들어오면 del\_timer를 통해서 종료하지않도록 설정해준다.

**3. 시험 및 평가 내용**

평가는 각각의 interrupt시 문서에 있는 기능이 잘 수행되는 지 확인하였다. 추가적으로 종료시나 반복적인 프로그램 수행 시 문제가 없는지 확인하였다.

**Test 1 –start**

시작시 gpio\_fnd의 값이 1초씩 증가하여 출력됨을 확인할 수 있었고, 분과 초가 잘 계산되어 출력됨을 확인할 수 있었다.

**Test 2 – stop**

interrupt시 del\_timer를 통해서 fnd화면에 변화를 주지않도록 한다.

**Test 3 – reset**

interrupt시 outw를 통해서 fnd에 0000을 출력하고 my\_timer.expire 값을 다시 초기화 해주어서 다시 1초후에 timer interrupt가 일어나도록 유도했다.

**Test 4 – quit**

interrupt시 3초이상 버튼을 누르고 있을 때 종료됨을 확인할 수 있었다. 3초 전으로 버튼을 누르고 띄었을 때 프로그램이 종료가 되지 않고, 원래대로 잘 수행되고 있음을 확인할 수 있었다.

**Test 5 – 여러 번 시행**

프로그램을 여러 번 시행했을 때 문제가 없는지 확인할 수 있었다. 그러나 device driver에서 선언하였던 전역변수들이 값이 변화한 그대로 프로그램을 다시 수행하였더니 문제가 발생하였다. 그래서 전역변수들을 device driver를 open할 때 초기화를 해주어 문제를 해결하였다.

위 시험을 통해 문제점을 파악하고, 수정하였다. 여러 번의 시험을 통해 보건 및 안전성을 갖추었다. Program 시작 시 gpio\_fnd, timer 그리고 전역변수들을 초기화하고, program 종료 시 fnd를 꺼주고, timer를 삭제해주는 과정이 잘 진행됨을 확인할 수 있으므로 생산성 및 내구성을 갖추었다고 볼 수 있다.

**V. 기타**

**1.** **연구 조원 기여도**:

20121600 송제호 (100%)

**2. 느낀점**

Gpio\_fnd와 timer의 경우 저 번 숙제에서 구현해 보아서인지 쉽게 구현할 수 있었다. 그러나 interrupt의 경우 실습이 후 처음 구현해 봄으로써 수업시간에 배운 내용을 더욱 명확히 할 수 있었다. 그러나 이번처럼 pause/start처럼 timer interrupt를 sleep하고 del\_timer는 처음해보는 기능이어서 이미 add\_timer했던 list\_timer에 다시 add\_timer를 했던가 del\_timer를 잘못하는 바람에 문제점이 발생하였다. 그리고 종료시키는 interrupt구현시 드라이버의 함수인 release가 수행되고도 종료 버튼에서 손을 띄는 순간에도 해당 interrupt 코드가 수행이 되어서 kernel panic이 발생했었는데 이를 보완하면서 interrupt에 대한 지식을 더욱 확실히 할 수 있었다.