**Embedded System Software 과제 3**

**(과제 수행 결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20161577, 김인호**

**개발 기간: 2020. 05. 30. – 2020. 05. 31**

**최 종 보 고 서**

1. **개발 목표**

Module programming, 디바이스 드라이버, timer, interrupt 등 강의와 실습을 통해 배운 내용을 이용하여 stopwatch 기능 구현.

1. **개발 범위 및 내용**
2. **개발 범위**

* 디바이스 드라이버: stopwatch 기능이 구현된 디바이스 드라이버를 하나의 모듈 형태로 제공
* 타이머: stopwatch 기능 구현을 위한 kernel timer 관리
* FPGA: FPGA의 FND 디바이스에 대한 출력을 간소화하기 위한 interface
* Interrupt: stopwatch 조작에 사용되는 버튼들에 대한 interrupt handling
* 유저 응용 프로그램: 개발한 디바이스 드라이버를 조작하고 stopwatch 기능을 구동 시키는 프로그램
* 로그: 명세서에 포함 안 된 부가 기능으로 디바이스 드라이버가 수행 중 유용한 정보나 FND 디바이스에 출력되는 데이터를 minicom으로 확인할 수 있도록 하는 logging 제공

1. **개발 내용**
   * 디바이스 드라이버

디바이스 드라이버는 이번 과제의 stopwatch 기능을 담고 있으며 모듈 형태로 Linux에 설치될 수 있다. 디바이스 파일에 수행되는 operation인 open(), close() 그리고 write() 각각에 대응되는 함수 포인터와 각 함수들이 구현되어 있다. 유저 응용 프로그램으로부터 넘어오는 별도의 parameter는 없으며 각 operation에 대응되는 함수마다 적절한 기능을 수행한다.

* + 타이머

타이머는 명세서의 stopwatch 기능을 구현하기 위한 핵심 요소로 이를 이용하여 stopwatch의 상태를 관리한다. 즉 stopwatch의 초를 세거나, 일시정지 하거나, 초기화 하는 등의 기능을 kernel timer를 이용하여 구현한다. 또한 Vol- 버튼의 종료 기능도 kernel timer를 이용하여 구현된다.

* + FPGA

이번 과제에 쓰이는 FPGA의 FND 디바이스를 사용하기 위한 IO-mapping 작업과 출력 interface를 구현한다. Interface를 이용하여 타이머의 stopwatch 기능 구현 부분에서는 FND 디바이스에 데이터를 넘겨야 하는 방식에 대한 로직을 간소화할 수 있다.

* + Interrupt

과제 명세서에 명시되어 있듯이 stopwatch 조작은 보드에 있는 Home, Back, Vol+ 그리고 Vol- 버튼으로 할 수 있다. 각 버튼에 대한 기능을 구현하기 위해 interrupt 등록 및 handling이 이뤄져야 하고 stopwatch 종료 전까지 유저 응용 프로그램을 sleep 시켰다가 다시 wake하는 부분을 구현한다.

* + 유저 응용 프로그램

이번 과제에는 유저 응용 프로그램으로부터 디바이스 드라이버에 넘어가는 parameter가 없기 때문에 사실상 유저 응용 프로그램은 `dev/stopwatch` 디바이스 파일을 이용하여 해당 디바이스 드라이버의 stopwatch 기능을 구동하는 담당만 한다. 디바이스 파일에 write()를 수행하면 stopwatch 기능이 시작되고 해당 유저 응용 프로그램의 프로세스는 디바이스 드라이버로 인해 sleep 하게 된다. Stopwatch 기능이 종료되면 다시 wake 되고 유저 응용 프로그램도 종료된다.

* + 로그

로그는 과제 명세서에 포함 안 된 부분이지만 개발 과정에 도움을 주고 에러 해결과 디버깅을 하는 데에 유용하게 쓰인 기능이다. 버튼을 눌렀을 때 해당 interrupt handler가 수행이 되어 어떤 함수들이 호출되었는지 기록을 볼 수 있고 stopwatch 기능을 수행하면서 FND 디바이스에 출력되는 데이터를 minicom으로도 확인할 수 있다.

1. **추진 일정 및 개발 방법**
2. **추진 일정**
   * 5월 30일 (토): 프로젝트 구조 및 디바이스 드라이버 구조 정의
   * 5월 31일 (일): 유저 프로그램, interrupt handling, stopwatch 기능 구현
3. **개발 방법**
   * 디바이스 드라이버

모듈 형태로 구현된 디바이스 드라이버는 터미널의 `insmod` 명령어로 설치할 수 있다. 이 때 다음과 같이 `module\_init()`으로 지정한 함수가 시작되어 모듈의 등록과 초기화 작업을 할 수 있다. 추가적으로 사용할 FPGA의 디바이스에 대한 IO-mapping을 하고 커널 타이머들을 초기화 한다.

A screenshot of a cell phone

Description automatically generated

그리고 모듈을 제거하기 위한 `rmmod` 명령어를 실행하면 `module\_exit()` 매크로로 지정한 함수가 실행된다. 해당 디바이스 드라이버에 대한 등록을 해제하고 등록된 타이머를 제거하고 IO-mapping 했던 디바이스를 다시 unmapping 한다.

A screenshot of text

Description automatically generated

설치된 디바이스 드라이버는 `mknod`로 만든 디바이스 파일에 대한 operation으로 활용할 수 있다. 각 operation에 대응되는 함수는 아래 `struct file\_operations` 자료구조에 함수 포인터로 지정한다.

A picture containing drawing

Description automatically generated

먼저 `open()`에 대응되는 함수는 다음과 같다. 드라이버의 사용 여부를 확인하고 버튼들에 대한 interrupt를 등록한다. `insmod` 대신 `open()`이 될 때 interrupt를 등록하는 이유는 stopwatch 기능을 수행할 때만 interrupt handling을 하기 위해서이다.

A screenshot of a cell phone

Description automatically generated

위와 반대로 `close()`에 대응되는 함수는 다음과 같다. 드라이버를 사용 안 하는 상태로 지정하고 등록했던 interrupt를 해제한다.

A screenshot of a cell phone

Description automatically generated

마지막으로 `write()`에 대응하는 함수이다. 이는 stopwatch 기능을 초기화 및 시작하고 해당 유저 응용프로그램 프로세스를 stopwatch 기능이 종료될 때까지 wait queue에 넣는다.

A screenshot of a cell phone

Description automatically generated

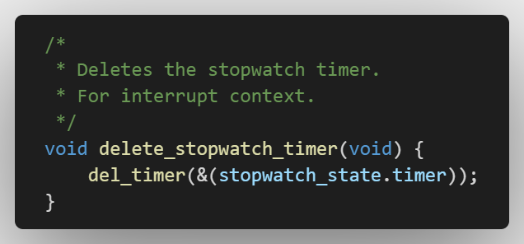
* + 타이머

먼저 사용한 kernel timer에 대한 초기화와 삭제를 하는 함수는 다음과 같다. 삭제를 할 때 interrupt context에서는 `del\_timer()`를 해주고 그 외에는 `del\_timer\_sync()`로 등록된 kernel timer를 삭제할 수 있다. 총 2개의 kernel timer를 활용하였는데 하나는 stopwatch의 초를 세기 위한 timer이고 다른 하나는 Vol- 버튼의 종료 기능을 구현하기 위한 timer이다.

A picture containing food

Description automatically generated A screen shot of a computer

Description automatically generated

 A picture containing drawing

Description automatically generated

Stopwatch의 초를 세는 timer와 관련된 부분을 먼저 살펴보면 먼저 아래는 stopwatch 기능의 상태를 관리하기 위한 자료구조이다. 멤버 변수로는 kernel timer 자료구조인 `timer`, 마지막으로 timer가 expire 됐을 때의 jiffies 값 `last\_callback`, 일시정지 시 소수점 아래 초를 저장하는 `pause\_offset` 그리고 총 몇 초를 세었는지 저장하는 `elapsed\_seconds`가 있다.

A screenshot of a cell phone

Description automatically generated

Stopwatch를 초기화하는 함수는 다음과 같다. FND 디바이스에 ‘0000’을 출력하고 stopwatch 상태 자료구조의 멤버를 초기화한다.

Screen of a cell phone

Description automatically generated

Stopwatch를 시작하고 일시정지하는 함수는 다음과 같다. 시작할 때는 안정성을 위해 사전에 등록되어 있을 수 있는 stopwatch kernel timer를 삭제하고 새로 다음 초에 대한 kernel timer를 등록하는 함수를 호출한다. 그리고 일시정지를 할 때도 등록된 kernel timer를 삭제하고 현재의 jiffies 값을 기준으로 마지막으로 초를 증가했을 때로부터의 offset을 저장한다.

A screenshot of a computer screen

Description automatically generated A screen shot of a social media post

Description automatically generated

그 다음으로 stopwatch 초기화와 종료를 하는 함수는 아래와 같다. 먼저 초기화 기능은 등록된 kernel timer를 삭제하고 stopwatch를 초기화 해주는 함수를 호출한다. 그리고 종료 기능의 함수도 등록된 kernel timer를 삭제하고 FND 디바이스에 ‘0000’을 출력한 후 sleep 되어있던 유저 응용 프로그램 프로세스를 ready queue에 넣어 wake한다.

A screen shot of a social media post

Description automatically generated A screen shot of a computer

Description automatically generated

다음은 stopwatch의 핵심 함수 중 하나인 초를 세기 위한 다음 kernel timer를 등록해주는 함수이다. 항상 현재 jiffies값으로부터 `HZ – offset`만큼 이후에 expire 되도록 등록을 하는데 이 때 offset은 위 일시정지 기능의 함수에서 저장한 값이며 1초보다 그 `offset`만큼 빠르게 expire 되도록 kernel timer를 등록하는 원리이다. 일시정지가 안 되어있을 때는 `offset`이 0이어서 정확히 `HZ` (1초) 이후에 expire되도록 등록이 된다. 추가적으로 expire 될 때 호출될 함수와 넘겨질 data를 지정하고 stopwatch 상태 자료구조의 `pause\_offset` 멤버 변수를 0으로 설정한다.

A screenshot of a cell phone

Description automatically generated

아래는 stopwatch의 kernel timer가 expire될 때마다 호출되는 callback 함수이다. 경과한 초를 증가시켜주면서 프로그램의 안정성을 위해 `MAX\_SECONDS`로 modulo 연산을 한다. `MAX\_SECONDS`는 3600이다. 그리고 현재의 jiffies값을 저장하고 FND 디바이스에 출력을 해주는 함수를 호출한다. 마지막으로 다음 kernel timer를 등록해주는 함수를 호출한다.

A screenshot of a cell phone

Description automatically generated

Stopwatch의 초를 세는 kernel timer와 관련된 마지막 함수로 FND 디바이스에 stopwatch의 초를 갱신하는 함수이다. 경과한 초를 기준으로 fpga\_fnd\_write() 함수에 60으로 나눈 몫과 나머지를 넘기는데 해당 함수에 대한 설명은 이후에 FPGA 부분에서 나타난다.

A picture containing drawing

Description automatically generated

다음은 stopwatch의 종료 기능을 구현하기 위한 kernel timer를 시작하는 함수이다. Vol-를 3초 이상 누르고 있으면 종료를 하는데 이 때 3초를 관리하기 위해 kernel timer를 3초 이후에 expire 되도록 등록한다. Vol-로 stopwatch를 종료하는 기능의 상세한 설명은 이후 interrupt 부분에 언급된다.

A screenshot of a computer

Description automatically generated

위에서 등록한 kernel timer가 expire되면 아래 callback 함수가 호출되는데 이는 위에서 설명한 `end\_stopwatch()` 함수를 호출한다.

A screen shot of a social media post

Description automatically generated

* + FPGA

먼저 사용할 FND 디바이스의 물리적 주소에 대해 IO-mapping을 하고 이를 해제하는 함수이다. 각각 디바이스 드라이버가 kernel에 설치될 때와 제거될 때 호출된다.

A screen shot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated

아래는 FND 디바이스에 주어진 분 `minutes`와 초 `seconds`를 출력하는 함수이다. 적절한 formatting 과정을 거친 후 `outw()`함수로 IO-mapping된 주소에 쓰여진다.

A screenshot of a cell phone

Description automatically generated

* + Interrupt

먼저 각 버튼에 대한 interrupt를 등록하고 해제하기 위한 함수는 다음과 같다. 사용된 버튼은 Home, Back, Vol+ 그리고 Vol-이며 Vol- 버튼의 경우 누르거나 땔 때 interrupt가 발생하고 나머지 버튼은 땔 때만 interrupt가 발생하도록 등록되었다. 등록하는 함수에서는 추가적으로 사용하는 변수에 대한 초기화 작업도 하였다.

A screen shot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screen shot of a computer

Description automatically generated

아래는 유저 응용 프로그램 프로세스를 sleep 시키고 wake하기 위한 변수와 함수이다. `DECLARE\_WAIT\_QUEUE\_HEAD` 매크로로 사용할 wait queue 자료구조를 지정하고 `interruptible\_sleep\_on()` 함수와 `\_\_wake\_up()` 함수로 각각 sleep 시키고 wake 할 수 있다. Sleep을 하면 해당 유저 프로세스가 wait queue에 들어가게 되고 wake 할 때는 ready queue에 넣어 scheduling 될 수 있도록 한다.

A screen shot of a person

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

각 버튼에 해당하는 interrupt handler 함수를 살펴보기 전에 각 버튼을 2번 연속 눌렀을 때 발생할 수 있는 에러를 막기 위해 사용한 변수가 있다. 아래 `last\_pressed` 변수는 어떤 버튼이 마지막으로 눌렸는지를 저장하는 `enum`이다. Vol-을 제외한 버튼이 이 변수 값을 기준으로 2번 이상 연속으로 눌렸을 때 각 handler에서는 바로 `return`되어 아무 기능을 수행하지 않는다.

A screenshot of a cell phone

Description automatically generated

Interrupt handler 함수 중 먼저 stopwatch를 시작하는 기능인 Home 버튼 interrupt에 대한 handler이다. 먼저 2번 이상 연속을 눌렀는지 확인하고 위에서 설명한 `start\_stopwatch()` 함수를 호출한다.

A screenshot of a cell phone

Description automatically generated

그 다음으로 stopwatch의 일시정지 기능인 Back 버튼에 대한 interrupt handler이다. 2번 이상 연속으로 눌린 여부와 stopwatch 기능의 시작 여부를 먼저 확인 후 위에서 설명한 `pause\_stopwatch()` 함수를 호출한다.

A screenshot of a cell phone

Description automatically generated

아래는 stopwatch를 초기화하는 Vol+ 버튼에 대한 interrupt handler이다. 먼저 2번 이상 눌렸는지 확인하고 위에서 설명한 `reset\_stopwatch()` 함수를 호출한다.

A screenshot of a cell phone

Description automatically generated

마지막으로 3초 이상 눌렀을 때 stopwatch 기능을 종료하는 Vol- 버튼에 대한 interrupt handler이다. 변수 `is\_vol\_down\_pressing`은 Vol- 버튼이 눌러져 있을 때 1이고 아니면 0이다. 즉 0일 때 해당 interrupt가 발생한 것은 버튼을 누름으로 인한 interrupt이며 이 때는 위에서 설명한 `start\_exit\_timer()`를 호출한다. 반대로 1일 때 interrupt가 발생한 것은 버튼을 누른 상태에서 땠을 때 발생한 것으로 이 때는 등록되어 있던 kernel timer를 삭제한다. 따라서 `start\_exit\_timer()` 호출 시 3초 후에 expire 되어 stopwatch 기능을 종료시키는 callback 함수를 호출하는 kernel timer가 등록이 되는데 그 3초 전에 Vol- 버튼을 누른 상태에서 땐다면 `delete\_exit\_timer()` 함수 호출로 인해 해당 kernel timer가 삭제되는 것이다. 3초 이상의 시간이 지난 후 Vol-을 땐다면 그 전에 등록했던 kernel timer가 expire되어 stopwatch 기능이 종료되고 유저 응용 프로그램도 종료된다.

A screenshot of a cell phone

Description automatically generated

* + 유저 응용 프로그램

유저 응용 프로그램은 `/dev/stopwatch` 디바이스 파일을 `open()` 후 `write()`를 실행한 다음 `close()`를 한다. 여기서 `write()`를 할 때 디바이스 드라이버에 넘기는 별도의 parameter는 없으며 `write()`실행 후 sleep에 들어가게 된다. Stopwatch 기능이 종료하면 wake 되어 `close()`를 하고 종료하게 된다.

A screenshot of a cell phone

Description automatically generated

* + 로그

Logging은 과제 명세서에 포함이 안 된 부분이지만 디바이스 드라이버의 여러 부분에서 유용한 정보를 minicom에 출력하여 쉽게 flow를 파악할 수 있도록 도움을 준다. 주어진 logging level인 `level`에 따라 `printk()`의 첫 부분이 달라지고 나머지는 일반적으로 `printk()`와 동일하게 formatting을 하여 출력한다.

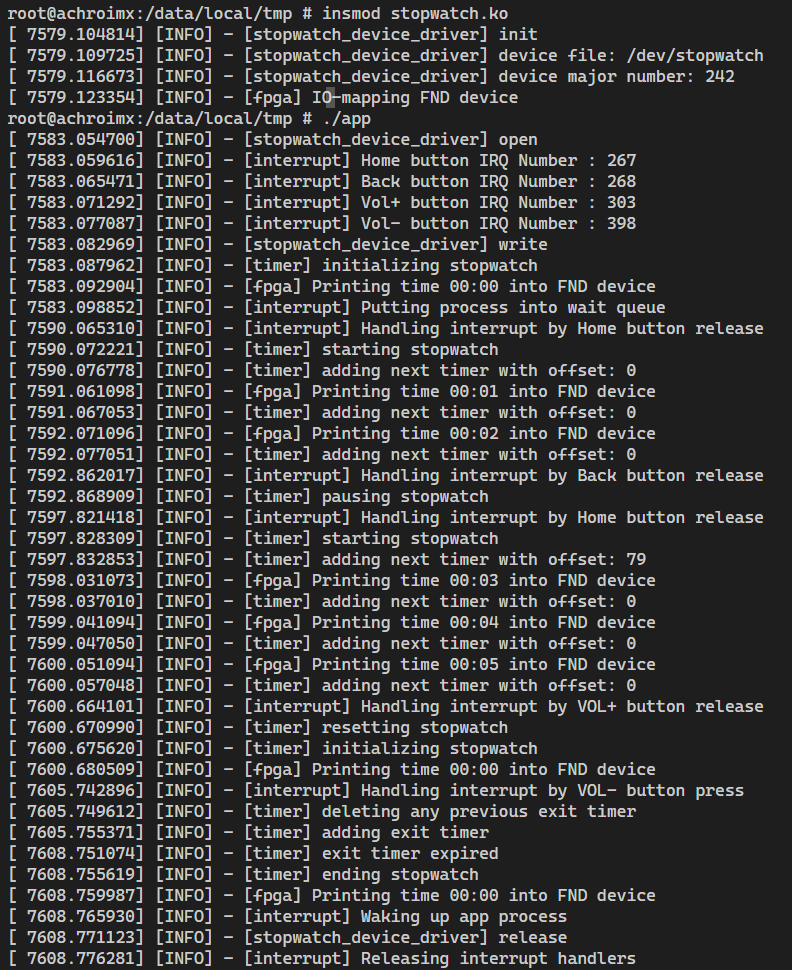
A screenshot of text

Description automatically generated

1. **연구 결과**

최종적으로 과제 명세서에 명시된 모든 기능을 성공적으로 구현할 수 있었다. 프로그램의 안정성을 위해 여러 조치를 하였고 logging을 통하여 어떤 부분이 실행되고 있는지 또는 어떤 출력이 일어나고 있는지 파악할 수 있다. 정상 종료 후 초기화 된 상태로 되는 것도 확인할 수 있었으며 `insmod`와 `rmmod`로 인한 디바이스 드라이버 설치 및 제거도 정상적으로 작동하였다.

A circuit board

Description automatically generated 

1. **기타**

이번 과제를 통하여 과제2에서 구현한 내용과 interrupt handling 내용을 추가적으로 함께 복습할 수 있었던 유익한 과정으로 판단할 수 있다. 과제2에서 구현 범위 별로 파일을 나눠 모듈화를 잘 했기 때문에 이번 과제를 과제2 기반으로 해서 수월하게 진행할 수 있었다.