**Embedded System Software 과제 2**

**(과제 수행 결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20161577, 김인호**

**개발 기간: 2020. 05. 22. – 2020. 05. 24**

**최 종 보 고 서**

1. **개발 목표**

이번 과제는 강의 내용 중에 배운 device driver, module development, kernel timer, interrupt 등을 바탕으로 FPGA 모듈을 활용한 가상의 타이머 디바이스 드라이버를 모듈 형태로 구현하고 이를 설정 및 실행할 수 있는 유저 응용 프로그램을 만드는 것이 목표이다.

1. **개발 범위 및 내용**
2. **개발 범위**

* 디바이스 드라이버: 타이머 기능을 담고 있는 모듈 형태의 디바이스 드라이버
* 타이머: 커널의 타이머 이용을 통해 명세서에 명시된 타이머 기능을 구현
* FPGA 모듈: FPGA 디바이스에 대한 출력을 간소화하기 위한 interface 제공
* 유저 응용 프로그램: 개발한 디바이스 드라이버를 조작하고 기능을 구동 시키는 프로그램
* 로그: 명세서에 포함 안 된 부가 기능으로 디바이스 드라이버가 수행 중 유용한 정보나 FPGA 디바이스에 출력되는 데이터를 minicom으로 확인할 수 있도록 하는 logging 제공

1. **개발 내용**
   * 디바이스 드라이버

디바이스 드라이버는 커널 모듈 형태로 개발되며 디바이스를 커널에 등록 및 등록 해제 작업을 담당하고 open(), ioctl(), close() 등의 디바이스 파일에 수행되는 각 operation에 대응되는 함수를 포함한다. 여기서 ioctl()은 `IOCTL\_SET\_OPTION`과 `IOCTL\_COMMAND` 이 2가지 명령에 대한 기능을 수행해야 한다. `IOCTL\_SET\_OPTION` 명령으로는 유저 응용 프로그램으로부터 parameter를 전달받고 `IOCTL\_COMMAND` 명령으로는 타이머 기능을 구동한다.

* + 타이머

타이머는 이번 과제 명세서에 포함된 타이머 관련 핵심 기능들을 FPGA 모듈을 이용한 구현을 포함한다. 유저 응용 프로그램으로부터 주어진 parameter를 기준으로 초기 상태를 설정하고 다음 타이머 등록과 타이머 상태 갱신을 해준다. Parameter는 TIMER\_INTERVAL (각 timer expire 사이의 HZ 시간 값), TIMER\_CNT (timer expire의 발생 횟수) 그리고 TIMER\_INIT (FND 디바이스에 출력될 초기 문양)이 주어진다.

* + FPGA 모듈

FPGA 모듈은 과제 명세서에 명시된 타이머 기능 구현에 포함된 FPGA 디바이스들을 제어하는 부분이다. 사전에 알려진 각 FPGA 디바이스의 물리적 주소를 IO-mapping하여 해당 디바이스 드라이버에서 활용할 수 있도록 인터페이스를 제공하게 된다. 사용되는 FPGA 디바이스는 4가지이며 Dot Matrix, FND, LED 그리고 Text LCD이다.

* + 유저 응용 프로그램

유저 응용 프로그램은 개발한 디바이스 드라이버에 parameter를 전달하고 구현된 기능을 실행하기 위한 프로그램이다. Parameter는 command line으로부터 주어지며 이를 확인 및 parsing을 담당한다. `/dev/dev\_driver` 이름으로 된 디바이스 드라이버 파일을 통하여 ioctl()의 `IOCTL\_SET\_OPTION` 명령으로 parameter를 전달한다. 그리고 ioctl()의 `IOCTL\_COMMAND` 명령으로 타이머 기능을 실행한다.

* + 로그

로그는 명세서에는 명시는 안 되어있지만 추가한 부가 기능으로 Linux 보드의 출력을 볼 수 있는 minicom interface로 디바이스 드라이버의 여러 log를 확인할 수 있는 기능을 포함한다. 에러 발생 또는 어떤 함수의 실행 등 디바이스 드라이버의 실행 관련 log도 확인할 수 있지만 특히 타이머 기능과 관련해서 어떤 FPGA 디바이스에 어떤 출력이 이뤄졌는지 실시간으로 확인할 수 있다.

1. **추진 일정 및 개발 방법**
2. **추진 일정**
   * 5월 22일 (금): 프로젝트 구조 및 디바이스 드라이버 구조 정의
   * 5월 23일 (토): 유저 프로그램, FPGA 모듈 및 로그 기능 구현
   * 5월 24일 (일): 타이머 기능 구현 및 테스팅
3. **개발 방법**
   * 디바이스 드라이버

디바이스 드라이버는 모듈 형태로 커널의 일부가 된다. 따라서 터미널의 `insmod` 명령어를 통해 모듈이 커널에 설치가 되는데 이 때 실행되는 함수는 `module\_init()` 매크로로 다음과 같이 지정할 수 있다.

A screenshot of a cell phone

Description automatically generated

해당 함수가 실행되면 디바이스 드라이버의 각 operation에 대응되는 함수 포인터를 포함하는 `struct file\_operations` 자료구조를 이용하여 드라이버를 등록하고 사용될 FPGA 디바이스를 사용할 수 있도록 IO-mapping을 한다.

반대로 `rmmod` 명령어를 실행하면 `module\_exit()` 매크로로 지정된 함수가 실행되며 다음과 같이 드라이버를 등록 해제하고 타이머 삭제와 IO-mapping된 FPGA 디바이스의 mapping을 해제하는 작업을 한다.

A screenshot of a cell phone

Description automatically generated

위에서 언급되었듯이 디바이스 드라이버는 `struct file\_operations` 자료구조에 각 operation에 대응되는 함수 포인터를 지정하여 등록된다.

A picture containing drawing

Description automatically generated

위에 대응되는 함수 포인터 중 먼저 `open()`에 대한 함수를 살펴보면 다음과 같다. 해당 드라이버의 사용 여부 확인 과정을 수행한다.

A screenshot of a cell phone

Description automatically generated

이와 비슷하게 `close()`에 대한 함수는 다음과 같다. 더 이상 사용 안 하는 상태로 바꾼 후 종료한다.

A screenshot of a cell phone

Description automatically generated

그 다음으로는 핵심 부분인 `ioctl()`관련이다. Kernel 2.6까지만 해도 `struct file\_operations`에 `.ioctl` 멤버로 존재했지만 현재 보드에 있는 커널의 버전에서는 `.unlocked\_ioctl` 멤버로 대응된다. 이미 인코딩을 한 ioctl 명령을 갖고 있는 헤더파일을 활용하여 `ioctl\_num`의 값에 다라 switch 문으로 각 명령을 구분할 수 있다. `IOCTL\_SET\_OPTION`의 경우에는 `char\*` 타입의 parameter가 전달된 것이며 이를 유저로부터 갖고 온 후 parsing한다. 그리고 타이머를 호출하여 타이머의 초기 상태를 지정한다. 그리고 `IOCTL\_COMMAND` 경우에는 별도의 parameter가 없으며 타이머 기능을 시작하는 함수를 호출한다.

A screenshot of a cell phone

Description automatically generated

위에서 언급된 ioctl 명령의 인코딩은 다음과 같다. `IOCTL\_SET\_OPTION`의 경우 `char\*` 타입의 parameter가 있고 `IOCTL\_COMMAND`는 parameter가 없다.

A close up of a black background

Description automatically generated

* + 타이머

타이머의 기능을 살펴보기 전 먼저 타이머의 상태를 저장하기 위한 `struct`는 다음과 같다. `timer`는 커널에 타이머를 등록하기 위한 자료구조이며 `count`는 타이머 발생 횟수, `COUNT\_END`는 총 발생 횟수, `INTERVAL`은 발생 간의 간격이다. 출력될 문양에 대해서는 문양 `digit`(1~8 사이 숫자)과 문양이 FND 디바이스에 출력될 위치 `digit\_index`를 저장한다. 그리고 Text LCD 디바이스에 출력될 각 줄의 글에 대한 내용은 `text`에 저장되고 각 줄의 움직이는 방향은 `text\_direction`, 그리고 각 줄의 내용이 출력이 시작될 위치는 `text\_index`에 저장된다.

A screen shot of a computer

Description automatically generated

타이머 기능은 초기화로 시작한다. 이는 위에서 본 `IOCTL\_SET\_OPTION`에서 호출되며 parameter들이 넘어온다. 이를 바탕으로 타이머 상태를 초기화 해준다.

A screenshot of a cell phone

Description automatically generated

그 다음에 `IOCTL\_COMMAND`로부터 타이머 시작 호출이 되면 다음 함수가 시작된다. 기존에 등록된 타이머가 있다면 이를 삭제하고 초기 상태를 출력하고 다음 타이머를 커널에 등록시킨다. 등록 시 `timer\_callback` 함수에 대한 포인터를 expire 할 때 실행하도록 한다.

A screenshot of a cell phone

Description automatically generated

타이머가 expire되면 `timer\_callback`이 실행된다. 여기서 타이머 상태를 다음 상태로 갱신을 해주고 그 갱신된 상태를 FPGA 모듈을 통해 출력한다. 만약 총 expire 발생 횟수에 도달했다면 더 이상 등록을 안 하고 각 FPGA 디바이스의 초기 상태를 출력한다. 갱신은 과제 명세서 대로 구현을 하였으며 Text LCD 디바이스에 출력되는 내용에 대한 갱신은 `move\_text()`라는 함수로 구현하였다. 각 줄이 움직이는 방향을 기준으로 끝에 도달했는지 확인하고 필요시 방향 전환을 해준다.

A screenshot of text

Description automatically generated A screenshot of a cell phone

Description automatically generated

타이머 상태에 대한 출력은 다음 함수로 수행한다. 저장된 상태를 기반으로 FPGA 모듈에서 만들어 둔 각 디바이스 별 interface를 활용하여 출력을 한다.

A screen shot of a computer

Description automatically generated

마지막으로 해당 디바이스 드라이버를 포함하는 모듈이 제거가 될 때 타이머를 삭제해야 하는데 이는 다음 함수의 호출로 할 수 있다.

A picture containing drawing, food

Description automatically generated

* + FPGA 모듈

FPGA 모듈을 타이머의 기능에서 FPGA 디바이스로의 출력 부분을 단순화하기 위해 만들었다. 먼저 각 FPGA 디바이스의 IO-mapping부터 시작을 하는데 `fpga.h` 헤더파일에 각 물리 주소가 명시되어 있다.

A screen shot of a person

Description automatically generated

이를 사용하여 디바이스 드라이버가 `init` 될 때 IO-mapping을 다음과 같이 실행하고 `release` 될 때 unmapping을 해준다.

A screenshot of text

Description automatically generated A screen shot of a computer

Description automatically generated

마지막으로 각 FPGA 디바이스에 대한 인터페이스 함수는 다음과 같다. `fpga\_led\_write()`는 현재 출력 중인 문양을 받아 해당 문양 번호의 LED를 켜준다. 0이 들어오면 모든 LED가 꺼진다. `fpga\_fnd\_write()`는 문양을 출력할 위치와 문양을 받아서 해당 위치에 해당 문양을 FND 디바이스에 출력해준다. `fpga\_dot\_write()`도 출력할 문양을 받아서 문양을 Dot Matrix로 출력해준다. 마지막으로 `fpga\_text\_lcd\_write()`는 각 첫 줄과 두번째 줄에 출력할 내용과 시작 위치를 받아서 이에 맡게 출력을 한다. 빈 공간은 ‘ ‘ (space)로 구성한다.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated A screenshot of text

Description automatically generated

* + 유저 응용 프로그램

유저 응용 프로그램에서는 먼저 command line으로부터 받은 parameter이 알맞게 들어왔는지 확인하고 parsing 해야 한다. 과제 명세서에 따라 다음과 같이 확인한다.

A screenshot of a cell phone

Description automatically generated

위를 확인한 후 아래와 같이 디바이스 드라이버 파일을 열어서 타이머 디바이스 드라이버를 조작한다. 두번의 `ioctl()`을 호출하는데 `IOCTL\_SET\_OPTION` 명령으로 먼저 parameter를 전달해주고 `IOCTL\_COMMAND`로 디바이스 드라이버의 기능을 구동 시킨다.

A screenshot of a cell phone

Description automatically generated

* + 로그

로그는 명세서에는 포함이 안 되어 있지만 과제를 개발 과정에 도움이 되고 타이머 기능 수행 시 FPGA 디바이스 외에도 minicom으로도 어떤 출력이 볼 수 있는 편의 기능이다.

A screenshot of text

Description automatically generated A close up of text on a black background

Description automatically generated

1. **연구 결과**

명세서에 따른 기능을 성공적으로 구현할 수 있었다. 여러 parameter로 테스트를 한 결과도 문제가 없었고 Text LCD에 출력되는 글을 바꿔보아도 문제가 없었다. 타이머 기능 후에는 초기 상태 (FND 디바이스는 ‘0000’, 나머지는 다 꺼진 상태)가 되는 것을 확인할 수 있었다. 중간 중간에 여러 로그도 minicom을 통해 확인할 수 있고 이를 이용하여 에러가 생기더라도 어디서 생겼는지 파악할 수 있다.

A circuit board

Description automatically generated

디바이스 드라이버, 타이머 그리고 FPGA 모듈이 유저 응용 프로그램으로부터 잘 작동이 되는 것을 확인할 수 있었다.

1. **기타**

이번 과제를 통하여 최근 몇주간 배운 강의 내용을 복습하고 이를 집적 적용하여 결과물을 구현함을 통해 강의에 대한 이해도를 높이는 데에 매우 유익했다고 판단할 수 있다. 과제 1과도 연관성을 찾을 수 있었고 강의의 큰 흐름을 파악하기 도와주는 과정이었다.

Virtual Box의 제한된 성능으로 인해 어려운 부분이 있었지만 이 또한 해결 방안을 찾아 나가는 것도 큰 도움이 되었다고 판단할 수 있다.