**Embedded System Software 과제 2**

**(과제 수행 결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20181593, 계인혜**

**개발기간: 2021. 05. 11. - 2021. 05. 16.**

**최 종 보 고 서**

**I. 개발 목표**

타이머 디바이스 드라이버 및 관련 테스트 응용 프로그램을 개발한다. 타이머 디바이스 드라이버 모듈내에는 타이머 기능 뿐만 아니라 fpga의 다른 장치들(led, lcd, fnd, dot)의 출력도 가능하도록 한다. 또한, ioctl을 통해 사용자의 입력을 타이머 디바이스 드라이버에 전달하고, 타이머 디바이스를 구동한다. 마지막으로 이를 수행할 수 있는 응용 프로그램 또한 제작해 본다.

**II. 개발 범위 및 내용**

**가. 개발 범위**

1. timer device driver : 디바이스 드라이버와 타이머 기능을 포함한 하나의 모듈을 구현한다.
2. 테스트 응용 프로그램 : 사용자로부터 타이머 디바이스 구동 옵션을 받아 ioctl을 통하여 타이머 디바이스 드라이버에 전달하고 ioctl을 통해 타이머 디바이스를 구동한다.
3. timer 기능 구현 : 유저의 입력에 따라 TIMER\_INIT 상태로부터 TIMER\_INTERVAL을 주기로 TIMER\_CNT번 커널 내에서 타이머를 수행한다.
4. fpga 모듈 출력 기능 구현 : ioctl을 통해 전달된 옵션을 이용하여 FND, LED, DOT, TEXT\_ LCD의 요구 사항에 따라 디바이스에 동시에 출력한다.

**나. 개발 내용**

1. timer device driver

디바이스 드라이버를 구현하기 위해 커널에 모듈 형태로 적재할 수 있도록 timer 모듈을 제작하였다. 디바이스 드라이버를 짜기 위해서는 device interface table의 함수 포인터에 연결할 함수들(device\_open, device\_ioctl, device\_release)을 작성해야 하고, 해당 디바이스를 device interface table에 function pointer로 연결해야 한다(register). 해당 모듈이 insert 될때 호출되는 timer\_device\_init 함수 내에서 명세서에 주어진 MAJOR\_NUM, DEVICE\_NAME을 사용하여 디바이스를 register 하였다. 또한 rmmod를 할 때 호출되는 timer\_device\_exit 함수 내에서 디바이스를 unregister 하였다.

1. 테스트 응용 프로그램

두 개의 ioctl 명령어를 사용하며, 이는 다시 Ioctl의 cmd에 따라 두 가지 경우로 나뉘는데, SET\_OPTION은 유저의 입력을 타이머 디바이스에 전달하고 COMMAND는 타이머 기능을 수행한다. Argument의 전달은 구조체를 이용하며, 올바르지 않은 입력이 들어오면 경고문을 출력하고 프로그램이 종료된다.

1. timer 기능 구현

사용자의 입력(TIMER\_INTERVAL)을 주기로 타이머를 동작시키며, 타이머가 expire되어도 TIMER\_CNT 만큼 타이머가 동작해야 하기 때문에 handler 내부에서 계속 타이머를 넣어주었다. 이 과정에서 출력 값들을 갱신하였으며, timer\_list 구조체의 멤버 (expires, function, data)값을 설정한다. 해당 기능은 이번 프로젝트의 핵심 부분으로 TIMER\_CNT만큼 타이머가 돌고 나면 FND를 제외한 모든 디바이스의 출력을 끄고, 타이머가 도는 동안 fpga의 출력을 올바르게 설정한다.

1. fpga 모듈 출력 기능 구현

명세서에 주어진 대로 각 디바이스에 올바른 값을 출력한다. 조교님께서 첨부해주신 fpga\_XXX\_driver 파일을 참고하여 physical address와 출력 방식을 확인할 수 있었다. 이렇게 알아낸 physical address를 ioremap하여 타이머 디바이스에서 활용할 수 있었다.

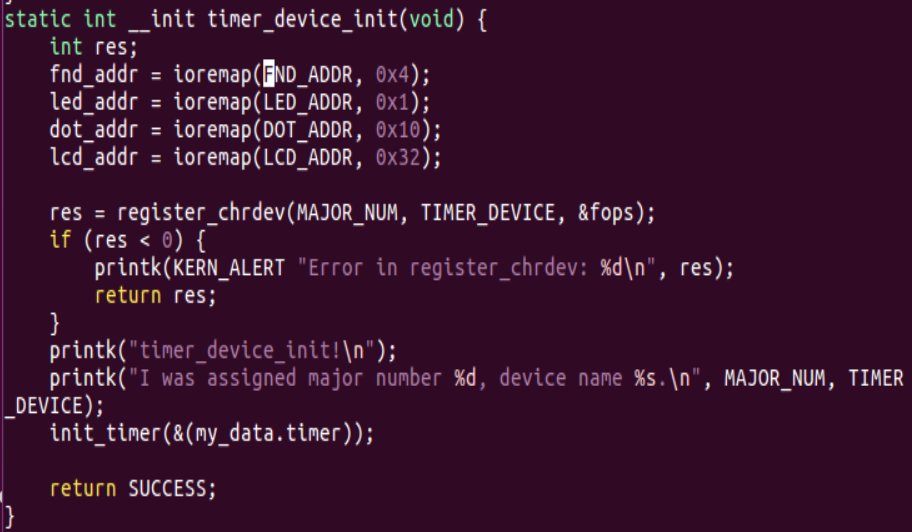
**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 21.05.11~21.05.12 | 프로젝트 명세서 분석 |
| 21.05.12 ~ 21.05.13 | Timer device driver 구현, timer 기능 구현 |
| 21.05.13 ~ 21.05.14 | 테스트 응용 프로그램 구현, fpga 출력 기능 구현 |
| 21.05.15 | 디버깅 및 주석 작성 |
| 21.05.16 | 보고서 작성 |

**나. 개발 방법**

1. timer device driver
2. module\_init



타이머 모듈을 insert하면 수행되는 함수로 fpga 출력을 위한 장치를 사용할 수 있도록 알려진 physical address로 ioremap한다. 또한 file\_operations 구조체의 멤버에 직접 구현한 함수들을 연결하여 드라이버를 register하고 타이머를 초기화한다. 이 과정에서 에러가 생기면 경고문을 출력하고 종료하며, 성공적으로 등록되면 디바이스의 MAJOR\_NUM, DEVICE\_NAME을 출력한다.

1. struct file\_operations

텍스트이(가) 표시된 사진

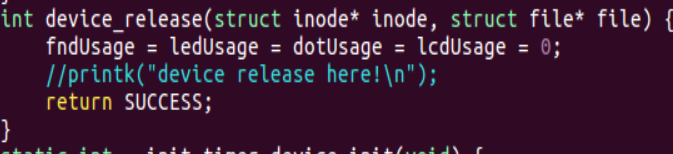
자동 생성된 설명

File\_operations 구조체는 <linux/fs.h>에 정의된 것으로 함수 포인터들을 멤버로 갖는다. 정의되어 있는 함수 포인터에 직접 작성한 함수들을 연결하여 등록한다.

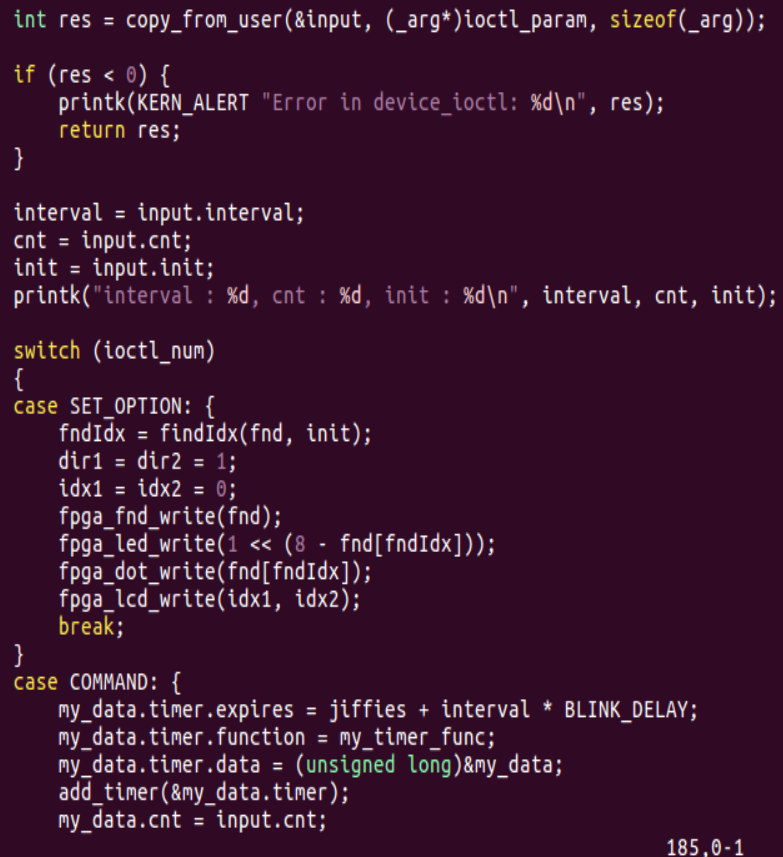
텍스트이(가) 표시된 사진

자동 생성된 설명

Device\_open 함수는 FND,LED,DOT,LCD 드라이버의 사용 여부를 usage counter로 확인하며, 하나라도 사용중일 시 -EBUSY 값을 리턴한다.



Device\_release 함수는 FND,LED,DOT,LCD 드라이버의 usage counter를 0으로 초기화하여 사용하지 않음을 표시한다.



Device\_ioctl 함수는 먼저 유저의 입력을 받아와 전역변수에 저장하고, ioctl\_num의 값에 따라 SET\_OPTION, COMMAND로 수행하는 명령들이 달라진다. SET\_OPTION의 경우 FND,LED,DOT,LCD 드라이버의 초기 값에 따라 값을 출력하며, COMMAND의 경우 타이머를 작동시킨다.

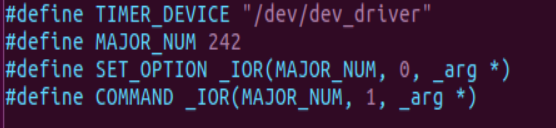
1. module\_exit

텍스트이(가) 표시된 사진

자동 생성된 설명

타이머 모듈을 delete하면 수행되는 함수로 fpga 출력을 위한 장치들을 iounmap한다. 또한 해당 드라이버를 unregister하고 타이머를 삭제한다. 성공적으로 삭제되면 제대로 삭제되었음을 출력한다.

1. 테스트 응용 프로그램



명세서에 주어진 DEVICE\_NAME, MAJOR\_NUM를 사용하였으며, command를 encoding하는데 매크로를 사용하였다. SET\_OPTION, COMMAND 모두 \_IOR로 type, number, datatype이 argument로 들어온다. type으로는 MAJOR\_NUM를 사용하였고, datatype으로는 \_arg 타입의 구조체를 사용하였다

.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ioctl command를 만들 때 필요한 argument가 여러 개기 때문에 구조체를 사용해서 넘길 수 있도록 하였다. 각각은 유저의 입력의 TIMER\_INTERVAL, TIMER\_CNT, TIMER\_INIT을 나타낸다.



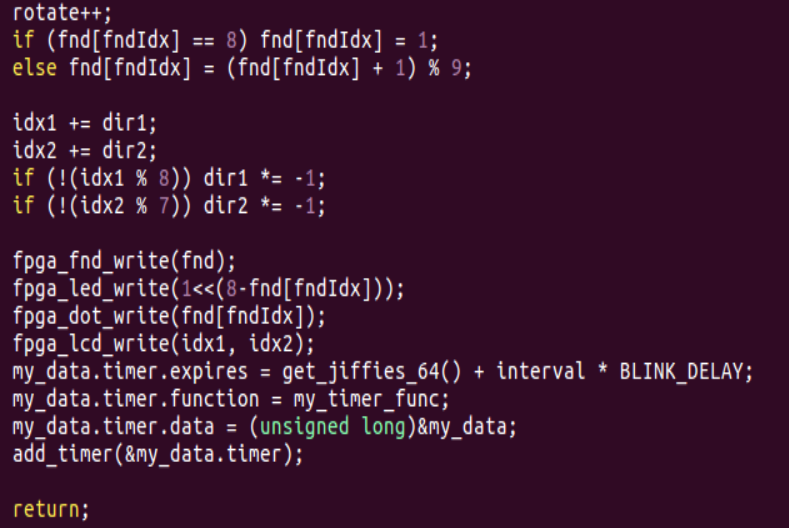
다음은 응용프로그램의 메인 함수로, 먼저 올바르지 않은 입력이 들어오면 경고문을 출력하고 프로그램을 종료한다. 입력 받은 값들을 정수로 전환하여 저장하고 ioctl로 전달하기 위해 이를 구조체에 집어넣는다. 그 후 ioctl을 두 번 호출하는데, SET\_OPTION에서는 유저의 입력을 전달하고, COMMAND에서는 타이머 디바이스 드라이버를 구동한다.

1. timer 기능 구현

텍스트이(가) 표시된 사진

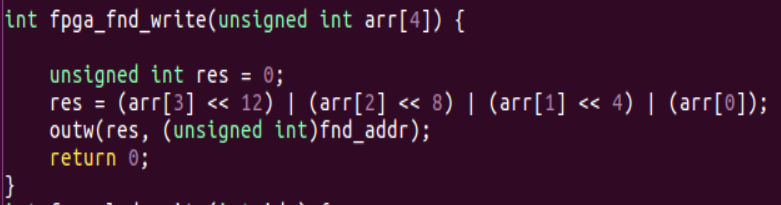
자동 생성된 설명

타이머가 한 번 수행될때마다 TIMER\_CNT를 1씩 줄여나가고, 남은 횟수가 0이 되면 명세서에 적힌 최종 값에 맞춰 FND,LED,DOT,LCD 드라이버에 출력한다. 특히 FND는 한 번의 로테이션 후 우측으로 이동하여 출력을 계속해야 하는데 이를 위해 rotate라는 변수를 선언했다. 또한 fndIdx라는 변수를 선언하여 fnd에서 0이 아닌 자리의 인덱스를 따로 저장하였다.



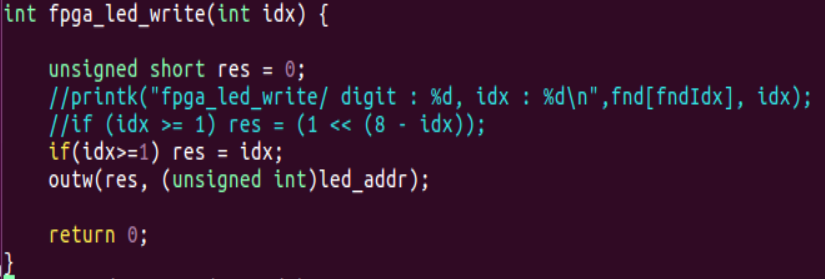
타이머가 수행될 때마다 rotate 값을 1씩 증가시키고, 현재 FND의 값보다 1 증가된 수를 FND에 재설정한다. Idx1, idx2는 각각 text\_lcd의 학번과 영문 이름의 첫 인덱스로 오른쪽으로 이동하면 dir=1, 왼쪽으로 이동하면 dir=-1로 설정하였다. 학번과 영문 이름의 길이가 다르기 때문에 두 줄을 각각 독립적으로 출력해야 했다. 저장된 값들을 기반으로 FND,LED,DOT,LCD 드라이버에 출력한다. 타이머가 expire되어도 TIMER\_CNT 만큼은 돌아야 하기 때문에 handler 내부에서 계속 타이머를 넣어주었다.

1. fpga 모듈 출력 기능 구현
2. FND



현재 FND에 출력 중인 값을 배열로 받아 shift와 or 연산을 이용해 하나의 정수 값으로 합친 후 출력한다. 출력할 physical address는 조교님께서 제공해 주신 fpga\_fnd\_driver.c 파일을 참고하였다.

1. LED



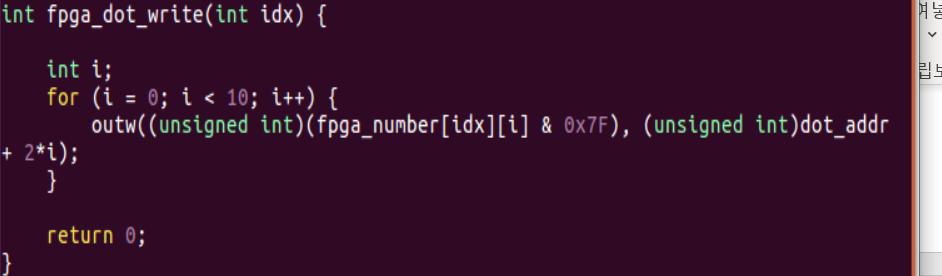
현재 DOT에 출력 중인 값을 LED 번호로 하여 출력한다. 출력할 physical address는 조교님께서 제공해 주신 fpga\_led\_driver.c 파일을 참고하였다.

1. DOT

텍스트이(가) 표시된 사진

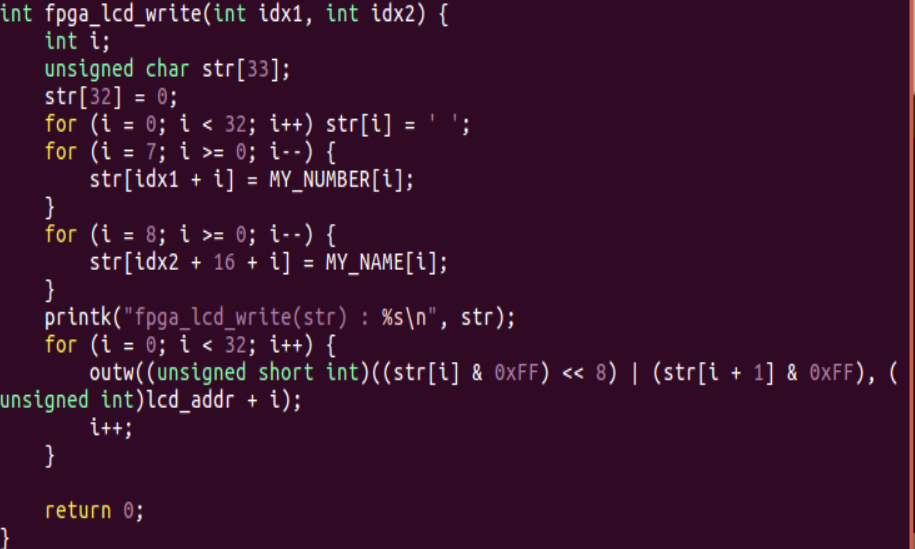
자동 생성된 설명

DOT에 출력할 값을 미리 저장해 둔 것으로 맨 윗줄은 TIMER\_CNT가 끝나고 아무것도 출력하지 않기 위해서 만들어 두었다. 1~8까지만 출력하면 되기 때문에 0과 9는 빼두었다.



위의 fpga\_number에 저장된 값을 출력한다. 출력할 physical address는 조교님께서 제공해 주신 fpga\_dot\_driver.c 파일을 참고하였다.

1. LCD



Text\_lcd는 각 줄이 16자리로 총 두 줄로 구성되어 있다. 이를 1차원 배열 하나로 저장하여 출력하였다. 먼저 모든 값을 빈칸으로 설정한 다음, 각 줄의 첫번째 인덱스부터 학번 또는 영문 이름을 하나씩 대입한다. 출력할 physical address는 조교님께서 제공해 주신 fpga\_text\_lcd\_driver.c 파일을 참고하였다.

**텍스트, 전자기기, 표지판이(가) 표시된 사진

자동 생성된 설명IV. 연구 결과 텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명**

왼쪽의 사진은 TIMER\_INIT을 8000으로 설정한 직후의 응용 프로그램 수행 화면으로 LED,FND,DOT,TEXT 모두 올바르게 출력되는 것을 확인할 수 있다. 오른쪽 사진은 TIMER\_CNT가 끝난 후의 FPGA 출력 화면을 찍은 것으로 명세서의 요구 사항 대로 FND를 제외한 모든 디바이스는 꺼져있는 것을 확인할 수 있다.

**V. 기타**

처음에는 매우 막막해서 명세서를 이틀이 넘도록 읽어보고 이전에 했던 실습들을 다시 확인해보았다. 오래도록 고민해보고 나니 어떻게 프로젝트를 진행해야 할지 가닥이 잡혔고, 시간이 모든 것을 해결해 준다는 말이 틀린 말이 아니다 싶었다. 특히 교수님께서 제공해 주셨던 linux kernel module programming guide가 큰 도움이 되었으며, 아마 이 파일이 없었더라면 프로젝트 진행에 훨씬 어려움을 겪었을 것 같다. 실습과 프로젝트를 진행할 때마다 느끼는 것이지만 크로스 컴파일과 가상 머신을 사용하는 환경이 조금 답답하기도 했다. 이론도 실습도 과제도 정말 지금껏 들었던 과목들 중 매우 어려운 편에 속하지만, 많이 배워갈 수 있어서 좋은 것 같다.