**Embedded System Software HW#2**

**(설계 프로젝트 수행 결과)**

과목 명 : [CSE4116] 임베디드 시스템 소프트웨어

담당 교수 : 박성용 교수님

개발 기간 : 2019. 05. 09 ~ 2019. 05. 13

학번 : 20141602

이름 : 황기덕

**프로젝트 제목 : Embedded System Software HW#2**

**제출일 : 2019년 5월 15일**

**참여 조원 : 황기덕**

1. **개발 목표**

* System call programming, module programming, 디바이스 드라이버 구현 등, 실습 시간 때 배운 내용을 활용하여 프로그램을 작성한다.

1. **개발 범위 및 내용**
   1. **개발 범위, 내용**

* **System call**

Parameter들을 받아서 하나의 변수로 만들어 return해주는 system call을 구현한다. 이 때 함수를 사용하지 않고 system call을 사용하여 필요한 값을 return 받도록 한다.

* **Device driver**

보드에서 fpga\_fnd, fpga\_led, fpga\_dot, fpga\_text\_lcd, timer module을 전부 포함한 하나의 module을 구현한다. 이를 위해 각각의 device driver에서 하는 일들을 새로운 device module을 만들어 그 안에 구현을 해 주어야 한다.

* **Application program**

device driver와 system call을 이용하여 간단한 출력을 해 주는 응용프로그램을 구현한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

|  |  |
| --- | --- |
| **일정** |  |
| **5.09** | **요구사항 분석** |
| **5.10 ~ 5.12** | **system call, module, application program 작성** |
| **5.13** | **보고서 작성** |

* 1. **개발 방법**
* **System call**

system call을 추가하기 위해선 커널소스 폴더에서 unistd.h, calls.h, syscalls.h, makefile을 건드리고 새로운 syscall을 만든 후 커널을 재컴파일해야 한다. 그러므로 수정이 필요한 코드들은 따로 빼고, .sh 파일을 통해 복사한 다음 커널을 재컴파일 해 주었다.

* **Device driver**

지금까지는 보드의 각 output device들을 따로따로 열고 그 open한 것에서 write하고 read하고 했었다. 하지만 이번에는 하나의 module을 만들고, open한 뒤 그 module에서 write를 해야 한다. 그러기 위해 각 device driver에서 해 주는 일을 module 하나에서 전부 해 주어야 한다. 그러므로 module init 할 때 ioremap 해 주고, module exit 할 때 unmap을 해 주어, write 할 때 사용할 수 있게끔 해 주었다.

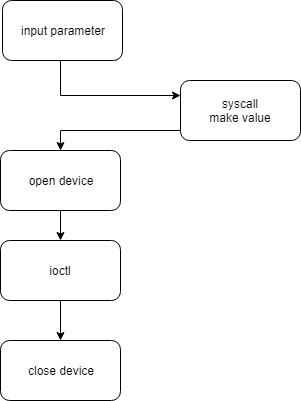
* **Application program**

실행을 해서 각 인자를 전달받고, 그 인자가 valid한 지 검사 후 syscall로 넘겨 dev driver에서 사용할 수 있는 value로 만들어 준다. 그 후 dev driver를 open한 후 ioctl을 통해 인자를 넘겨 주어서 원하는 결과를 출력할 수 있게 해 준 다음 close하였다.

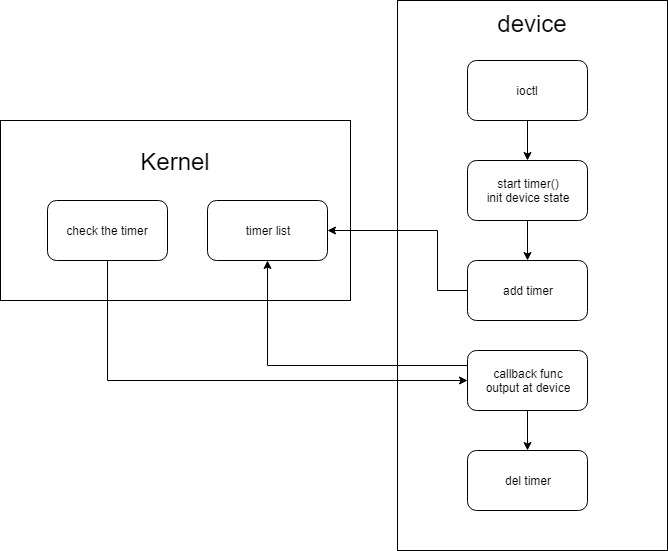
* 1. **연구원 역할 분담**
* 개인 프로젝트이다.

1. **연구 결과**
   1. **합성 내용**

* **application program**

****

* **module**

****

* 1. **제작 내용**
* **System call**

본 프로젝트에서는 parameter를 입력받은 후 바로 device open을 해서 넘기는 게 아닌, 세 개의 parameter을 syscall을 통하여 하나의 인자로 만들어 준 다음 그 값을 return해서 app program에 넘겨주고 값이 제대로 넘어왔나 err\_flag를 통하여 체크해 준 다음 dev driver에 넘겨 준다.

syscall을 새로 추가하기 위해 먼저 unistd.h 에서 새로운 syscall num을 받아 주고, calls.S에서 새로운 syscall을 호출해 주엇으며, syscalls.h에서 syscall 함수 원형 선언을 해 주었다. 커널을 재컴파일해야 하기 때문에 makefile도 새로운 syscall을 포함할 수 있게끔 만들어 주었으며, 이 과정을 전부 마치고 kernel을 재컴파일하여 syscall이 정상적으로 돌아갈 수 있게끔 하였다.

* **Module**

기존에 각각의 driver을 통해 write하던 4개의 output device들을 하나로 통합하였다. 그러기 위해 먼저 dev\_driver라는 이름을 가진 새로운 dev module을 만들어 주었고, init할 때 각각의 주소와 mapping 해 주었으며 exit할 때 unmap 해 주었다.

write를 사용하지 않고 ioctl를 이용해 구현해 주었다. 이를 위해 ioctl함수를 만들어 주고, fops에 넣어서 연결시켜 주었다. 그리고 app program에서 ioctl을 부르면 이곳으로 넘어와서 parameter를 start\_timer()라는 함수로 넘겨 주었다. 그 후 이 start\_timer함수에서 실행되게끔 해 주었다.

start\_timer() 에선 각종 상태를 초기화해주고, timer를 초기화 해 준 다음 init\_output()을 이용해 output device들을 초기화 해 주었다.

callback 함수에서는 사용자가 지정한 count가 다 되었는지 판단한 다음, 다 되지 않았다면 변수를 조정하고 device output 해 준 다음 다시 add timer 해 주었고, 만약 count가 다 되었다면 init\_output()을 이용해 output device를 초기화해주고 return 해 주었다.

print output()에서는 현재 상태가 어떤 값을 가지고 있는 지 보고, 그것에 알맞게 출력해 준 후 다음 출력을 위해 값을 적당히 변경해 주었다.

* **Application program**

사용자로부터 초기 값을 받아준 후, syscall을 통해 그 값을 하나로 통합해 주었으며 새로 만든 device module을 open, ioctl을 통해 device 출력을 하고 close해 주었다.

1. **기타**
   1. **실행 순서**

* 커널 재컴파일 -> module과 app program을 target board에 push -> insmod / mknod -> app program run
* ./20141602 <interval> <count> <start option>
  1. **연구 조원 기여도**
* 개인 프로젝트이다.
  1. **소감**
* device driver의 내부 구조에 대해 연구해 보았다. 만약 이러한 구조를 미리 알았다면 지금까지 해왔던 프로젝트들을 조금 더 효율 좋게 작성할 수 있었겠다는 생각이 들었다.