**Embedded 2st HW**

**(설계 프로젝트 수행 결과)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

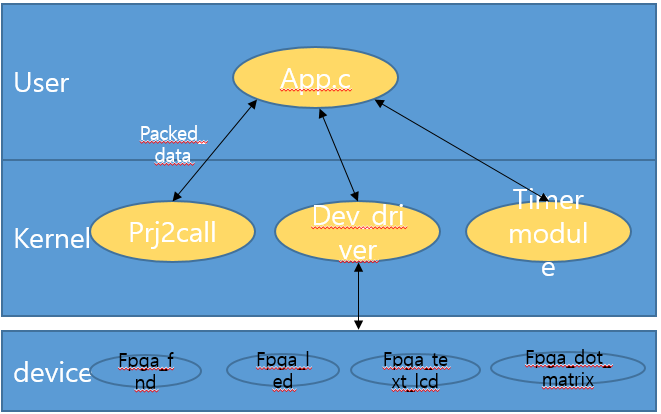
**학번 및 이름: 20121600 송제호**

**개발기간: 2017. 05. 08. -2017. 05. 11.**

**최 종 보 고 서**

**I. 개발 목표**

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



**II. 개발 범위 및 내용**

**가. 개발 범위**

1. 디바이스 드라이버 및 타이머: Device driver(fpga\_fnd, fpga\_led, fpga\_dot, fpga\_text\_lcd)와 timer module을 포함한 하나의 module을 구현한다 (/dev/dev\_driver).

2. Parameter들을 받아서 하나의 변수로 packing 해주는 system call을 구현한다.

(packed data의 구성은 밑의 내용에서 자세히 설명)

3. system call을 사용하여 4byte로 압축된 data를 받고 ioctl을 사용하여 device에 출력 명령을 내리는 application을 구현한다.

**나. 개발 내용**

**1. 디바이스 드라이버 및 타이머**

- 모든 디바이스의 값들이 바뀌는 시간은 타이머를 기준으로 한다.

**- fpga\_led**

: 현재fpga\_fnd에서출력중인문양의번호를나타낸다. (D1 : 1번, D2 : 2번, D3 : 3번, D4 : 4번, D5 : 5번, D6 : 6번, D7: 7번, D8 : 8번)모든 문양의 출력이 끝나면 fpga\_led의불을꺼준다.

**- fpga\_fnd**

: FND왼쪽4자리를출력으로컨트롤하고, 위와 같은 순서의 문양을timer에따라반복해서출력되게한다. 출력하는 위치는 한번의 로테이션이 끝날 때마다 우측으로 이동한다.(ex. 3번째자리에서1번문양부터시작하였을때, 8번문양까지출력한뒤, 4번째자리로이동하여다시1번문양부터8번문양까지출력한다.)지정횟수만큼의 출력이 끝나면FND의불을꺼준다.

**- fpga\_dot**

: 현재 fpga\_룽 에서 출력중인 문양과 같은 모양의 문양을 출력한다. fpga\_fnd의 문양이 바뀐다면, fpga\_dot도 fpga\_fnd와 같은 문양으로 함께 바뀐다. 지정횟수만큼의 출력이 끝나면 dot의 불을 꺼준다.

**-fpga\_text\_lcd**

첫 번째 줄에는 자신의 학번을 입력하고, 두 번째 줄에는 자신의 이름을 영문으로 입력한다. 두 줄 모두 timer에 따라 오른쪽으로 한 칸씩 shift이동을 하고, 오른쪽 칸에 더 이상의 공백이 없을 경우, 왼쪽으로 shift이동을 시작한다. 왼쪽 shift이동도 마찬가지로, 왼쪽 칸에 더 이상의 공백이 없을 경우, 오른쪽으로 shift 이동을 시작한다. 이 과정은 모든 문양의 출력이 끝날 때까지 계속되고, 종료 시 text\_lcd를 초기화 시킨다.

**2. System call**

argv 입력값들을(timer 주기, 반복 횟수, 시작 숫자와 자리수) 넘겨줘서 4바이트로 압축된 데이터로 다시 넘겨준다.

입력 값 : 시간간격, 횟수, 시작 숫자

리턴 값 : 해당 시스템콜의 system call number(376)

**3. 응용 프로그램**

디바이스 파일을 open하여, write를 통해 출력을 시작해준 뒤, close하고 종료된다.

실행 예)

./app 시간간격[1-100] 횟수[1-100] 시작옵션[0001-8000]

시간 간격 : 0.1초-10초

횟수 : 타이머 호출 횟수(8->한번의 로테이션)

시작옵션의 의미 : 0040 -> 3번째 자리에서 4번째 문양부터 출력을 시작한다.

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

2017.05.8 : 시스템콜 구현.

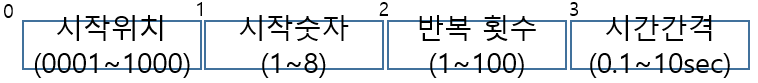
2017.05.8~10 : 디바이스 드라이버 및 타이머 구현.

2015.05.10~11 : 시험을 통해 문제점파악 및 수정.

2015.05.11 : 결과보고서 작성. 및 최종 제출

**나. 개발 방법**

응용 프로그램에서 사용자에게 시간 간격, 횟수, 시작 숫자를 입력 받는다. 입력 받은 데이터를 system call을 호출하여 넘겨준다. System call 함수에서 연산을 통해 4byte stream을 넘겨주고 성공적으로 system call을 마치면 376(해당 system call의 번호)를 리턴해준다. 여기서 4byte 중 처음 1 byte는 시작 fnd 위치를 나타내고, 다음 1 byte는 시작 fnd값을, 다음 1 byte는 시간간격, 마지막 1byte는 횟수를 나타낸다.

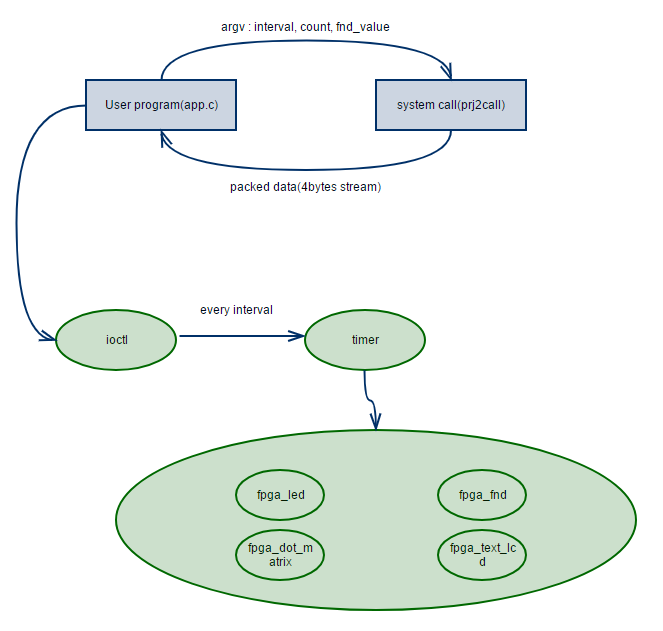


리턴된 값을 받은 응용프로그램은 디바이스 드라이버에 4byte 정보를 넘겨주고, 정보를 받은 디바이스 드라이버는 타이머를 기반으로 각 디바이스를 제어 및 작동시킨다. 매 시간간격마다(timer handler가 호출될때마다 timer->expire 값을 jiffies\_64 + interval로 초기화 시켜주고 add\_timer로 handler 함수를 다시 등록해준다. 그리고 매번 횟수(counter)가 0이되면 모든 I/O device에 화면을 지우도록 명령을 내리고 종료한다.

**IV. 연구 결과**

**1. 합성 내용**

**<전체 프로그램 구성도>**.



**2. 제작 내용**

**1) 응용프로그램**

응용프로그램에서는 사용자로부터 시간, 간격, 횟수 그리고 시작옵션을 입력 받는다. 입력 값이 prototype에 맞는지 확인한다. 시스템콜을 호출하여 입력 정보를 넘겨주고, 연산된 4byte정보를 return 받는다. 그 다음 device driver를 open하고, 4byte정보를 ioctl에 넘겨준다. 여기서 parameter 옵션은 0으로 통일했다.

**2) 시스템콜**

응용프로그램으로부터 시간간격과 횟수 그리고 시작옵션 정보를 넘겨받는다. shift연산을 이용하여 4byte에 모든 입력정보가 들어있는 값을 만들어 응용프로그램에 리턴 한다. 구현 과정은 다음과 같다.

**1. allocate a new system call number**

Kernel/arch/arm/include/asm/unistd.h 에 새로운 시스템 콜 숫자를 할당한다.

**2. Register into system call table**

Kernel/arch/arm/kernel/calls.S 에 new service routine을 추가해준다.

Kernel/include/linux/syscall.h 에 new system call prototype을 추가해준다.

**3. make out new system call**

Kernel/kernel/ 에 위에서 언급한 새로운 시스템 콜을 구현한다. Makefile에도 추가해준다.

**4. recompile the kernel**

Kernel을 다시 컴파일하여 target board에 올려준다.

**3) 디바이스 드라이버**

응용프로그램에서 ioctl을 통해 입력정보 값을 받아오고, 값을 기반으로 device들을 제어한다. 넘어온 값들을 다시 fnd위치, fnd값, 시간간격, 횟수로 바꿔준 뒤 timer를 설정한다. timer가 expire되는 시간간격을 설정하고, 시간간격마다 호출되는 함수도 설정해준다. Timer 설정 방법은 다음과 같다.

|  |
| --- |
| my\_timer->expires = get\_jiffies\_64() + time\_interval\*HZ/10;  my\_timer->function = timer\_handler;  add\_timer(my\_timer); |

Timer\_handler함수는 계속 시간간격마다 호출되므로 여기서 각 device들의 출력함수들을 호출해준다. 그리고 시간간격마다 fnd위치, fnd값, 횟수가 변경될 수 있도록 설정해준다. 호출된 각 device 함수들은 fnd위치, fnd값, 횟수에 알맞게 device에 출력해준다.

**3. 시험 및 평가 내용**

평가는 각 device별로 문서에 있는 기능이 잘 수행되는지 확인한다. 추가적으로 종료시나 다른 입력을 했을 때를 확인하였다.

**Test 1 – prototype과 틀린 입력**

./app 0 10 1000

./app 1 0 1000

./app 10 10 1100

./app 10 10 0000

./app 10 10 9000

등의 여러 틀린 입력 케이스를 참고하여 구현하였으므로 다른 에러 테스트 케이스에 대해서도 만족할 것이라 생각된다.

**Test – fpga\_led 확인**

명세서에 쓰인 조건들을 만족하였다. Gpio\_fnd가 출력하는 문양의 번호가 나타내고, 지정된 횟수 이후 led의 불이 꺼짐을 확인하였다.

**Test – fpga\_fnd 확인**

명세서에 쓰인 조건들을 만족하였다. 총 횟수가 출력되고, 매 시간마다 1씩 감소하며 마지막에 0으로 초기화됨을 확인하였다.

**Test – fpga\_dot 확인**

명세서에 쓰인 조건들을 만족하였다. 현재 출력되고 있는 문양을 출력하고, 지정된 횟수 이후 dot의 불이 꺼짐을 확인하였다.

**Test – fpga\_text\_lcd 확인**

명세서에 쓰인 조건들을 만족하였다. 첫번째 줄은 학번, 두번째 줄은 이름이 나오고, 매 시간마다 오른쪽으로 shift하고 공백이 없을 경우 왼쪽으로 shift됨을 확인하였다. 그리고 모든 문양 출력 후 lcd가 초기화됨을 확인하였다.

**Test – 종료시 모든 device 초기화 확인**

지정된 횟수 이후 모든 device들이 초기화되는지 확인하였다.

위 시험을 통해 보완해야 할 부분을 수정하였다. 문서의 명세화된 경우를 여러 번 시험을 통해 보건 및 안정성을 갖추었다. 또한 입력에 대해 다양한 오류처리를 해주었고, 종료 시 모든 device가 제대로 초기화됨을 볼 때 생산성 및 내구성을 갖추었다고 볼 수 있다.

**V. 기타**

**1.** **연구 조원 기여도**:

20121600 송제호 (100%)

**2. 느낀점**

시스템콜과 디바이스 드라이버 그리고 타이머 모듈을 구현하면서 수업시간에 배운 내용을 직접 구현해보니 더 명확히 이해할 수 있었다. 처음에 어떻게 구현을 할지, 무엇을 먼저 구현을 할지 고민하고, 여러 시행착오가 있었지만 시스템콜과 응용프로그램을 먼저 구현하고 그 다음으로 디바이스 드라이버를 구현하니 순차적으로 잘 진행되었던 것 같다. 디바이스 드라이버의 경우 지난 프로젝트 때 과제나 실습으로 사용은 해보았으나 직접 구현하는 것은 처음이었다. 매 번 과제가 나올 때마다 힘들게 과제를 하지만 이를 통해 더 많이 배울 수 있었던 것 같다. 다음 프로젝트도 열심히 할 수 있도록 노력해야겠다.