

# Embedded System Software HW#2

Due Date : 2018. 5. 28.

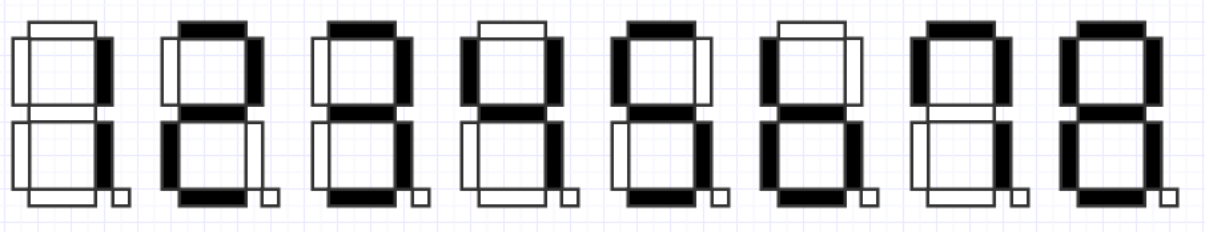
## 1. 목표

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

## 2. 구현

- (1) Device driver(fpga\_fnd, fpga\_led, fpga\_dot, fpga\_text\_lcd)와 timer module을 포함한 하나의 module을 구현한다.
- (2) Parameter들을 받아서 하나의 변수로 만들어 return 해주는 system call을 구현한다.
- (3) 구현된 디바이스 드라이버와 system call을 이용하여 간단한 출력을 해주는 응용프로그램을 구현한다.

## 3. 기능



위의 문양은 왼쪽부터 1번~8번이다.

### (1) 디바이스 드라이버 및 타이머

- 모든 디바이스의 값들이 바뀌는 시간은 타이머를 기준으로 한다.

#### - fpga\_fnd

: FND 왼쪽 4자리를 출력으로 컨트롤하고, 위와 같은 순서의 문양을 timer에 따라 반복해서 출력되게 한다. 출력하는 위치는 한 번의 로테이션이 끝날 때마다 우측으로 이동한다. (ex. 3번째 자리에서 1번 문양부터 시작하였을 때, 8번 문양까지 출력 한 뒤, 4번째 자리로 이동하여 다시 1번 문양부터 8번 문양까지 출력한다.) 만약 4번째 자리에서 한 번의 로테이션이 다 끝났다면, 1번째 자리로 이동한다.

지정 횟수 만큼의 출력이 끝나면 FND의 불을 꺼준다.

#### - fpga\_led

: 현재 fpga\_fnd에서 출력 중인 문양의 번호를 나타낸다. (D1 : 1번, D2 : 2번, D3 : 3번, D4 : 4번, D5 : 5번, D6 : 6번, D7 : 7번, D8 : 8번) 모든 문양의 출력이 끝나면 fpga\_led의 불을 꺼준다.

#### -fpga\_dot

: 현재 fpga\_fnd에서 출력 중인 문양과 같은 모양의 문양을 출력한다. fpga\_fnd의 문양이 바뀐다면, fpga\_dot도 fpga\_fnd와 같은 문양으로 함께 바뀐다. 지정 횟수 만큼의 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. (각 줄은 독립적이므로, 학번과 이름의 길이가 다를 경우, 더 이상 공백이 없는 순간이 달라져 각 줄은 다른 형태로 이동을 하게 된다)

이 과정은 모든 문양의 출력이 끝날 때까지 계속되고, 종료 시 text\_lcd를 초기화 시킨다.

#### -Timer

: Timer를 기준으로 모든 디바이스의 출력 시간을 결정한다.

- 디바이스 드라이버 이름은 /dev/dev\_driver로 통일한다. (major number : 242)  
(ioctl로 구현할 경우 추가 점수 "이 부분은 추가 구현이므로 질문을 받지 않습니다.")

#### (2) System call

입력 값을 받아서, write() 함수에 넣어줄 데이터를 return 한다.

입력 값 : 시간간격, 횟수, 시작옵션

리턴 값 : 4byte stream(1byte: 시작fnd위치, 1byte: 시작fnd값, 2byte:기타필요한것들)

->이 구성은 스스로 정하여, 보고서에 자세히 기술한다.(이 부분이 gdata로 넘어간다.)

#### (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.

예시) ./app 1 14 0700

2번째 자리 -> 7, 8, 1, 2, ..., 6번째 문양 출력. (총 출력 횟수 : 8)

3번째 자리 -> 7, 8, 1, 2, 3, 4번째 문양 출력. (총 출력 횟수 : 14)

## 4. 제출 방법

### (1) 제출 파일

-> app폴더(소스코드, Makefile)

-> module폴더(소스코드, Makefile)

-> kernel폴더(system call을 추가하는데 수정한 파일만을 포함한다. 폴더의 구조까지 동일하게 구성, Makefile)

-> Document (ex. [HW2]20001234.docx or [HW2]20001234.hwp)

-> readme.txt (본인 driver의 name과 major number를 기술)

### 파일 압축 방법

학번 폴더(20001234)를 생성하고, 그 안에 숙제 관련 파일, 폴더들을 저장한 뒤, 학번 폴더가 있는 위치에서 `tar -cvzf [HW2]학번.tar.gz ./학번폴더`

### (2) 메일 제출 형식(조교 e\_mail: embehw2017@gmail.com)

[HW2]학번\_이름(ex. [HW2]20001234\_홍길동)

### (3) Document를 출력하여 하드카피 제출(미제출시 문서 점수 없음)

하드카피 제출시간 또한 동일

## 5. 평가 기준

(1) 프로그램 80%, 문서 20%, 추가구현(ioctl로 구현) 10%

(2) Late, 하루에 10% 감점

(3) 제출 형식 틀린 경우 10% 감점

(4) Copy 적발 시 0점