Embedded System Software HW#1

Due Date : 2021년 4월 15일 23시 59분

1. 목표

디바이스 컨트롤과 IPC를 이용하여 주어진 Clock, Counter, Text editor, Draw board를 구현한다.

2. 구현

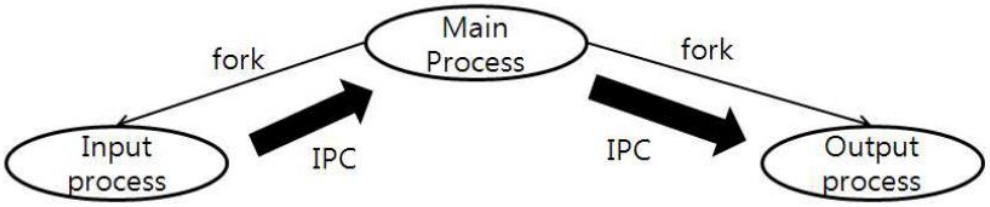


그림 1 Process

Main process는 2개의 프로세스를 fork한다. 본 프로그램은 총 3개의 프로세스로 구성되고, 프로세스 간의 통신은 IPC를 사용한다.

각 프로세스의 구성은 다음과 같다.

Input process

이 프로세스는 계산을 위해 입력을 받는 입력 장치를 관리한다. 주기적으로 입력 장치로부터 새로 들어온 값이 있는지 확인하고, 새로 들어온 값을 main process로 IPC를 사용하여 전달한다.

Main process

이 프로세스에서는 연산의 실질적인 수행을 한다. 입력 프로세스에게서 값을 받아서, 각 모드에 따른 적절한 연산을 한다. 그 다음, 연산된 값을 IPC를 통해 출력 프로세스로 전달하는 역할을 한다.

Output process

이 프로세스는 Main process로부터 전달 받은 값을 해당하는 장치에 출력하는 역할을 한다. 본 프로그램의 IPC는 **shared memory** 방법을 이용하여 IPC를 구현하고, 이를 문서에 상세히 명시해야 한다.

Device Control

디바이스를 컨트롤하는 방법에는 mmap()함수를 이용하는 방법과 디바이스 드라이버를 사용하는 방법이 있다. **LED 디바이스는 mmap()함수를 사용**하고, **나머지 디바이스들은 디바이스 드라이버를 사용**하여 프로그램을 구현하고, 이를 문서에 상세히 명시해야 한다.

3. 기능

Clock, Counter, Text editor, Draw board 기능을 구현한다. 모드 1에서는 Clock 기능을, 모드 2에서는 Counter 기능을, 모드 3에서는 Text editor를, 모드 4에서는 Draw board를 구현한다. 기본적으로 모드가 변경되면, **각 모드의 초기 상태**가 된다. **(각 모드에서 언급이 없는 나머지 디바이스의 초기 상태는 빈 화면, 혹은 불이 모두 꺼진 상태이다.)**

또한, 프로그램 시작 시 **기본 모드(default mode)**는 모드 1이다.

각 **모드 변경 시, 현재 사용 중인 모드를 제외한 다른 모드에서 사용하는 디바이스들은 모두 초기화** 시킨다.

전체 프로그램은 "BACK" 키를 입력하였을 때만 정상 종료되며 이 때 **모든 device는 각자의 가장 초기 상태**가 된다. **(FND 0000, 나머지 device off)**

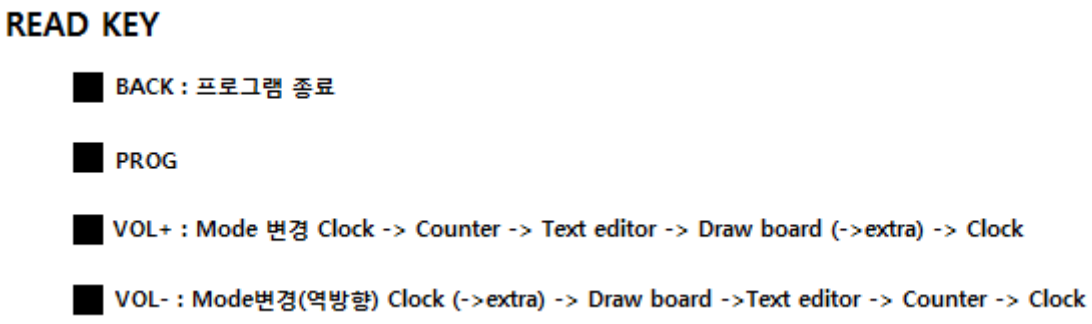


Figure 1. 모드 변경

1) Clock - 모드 1

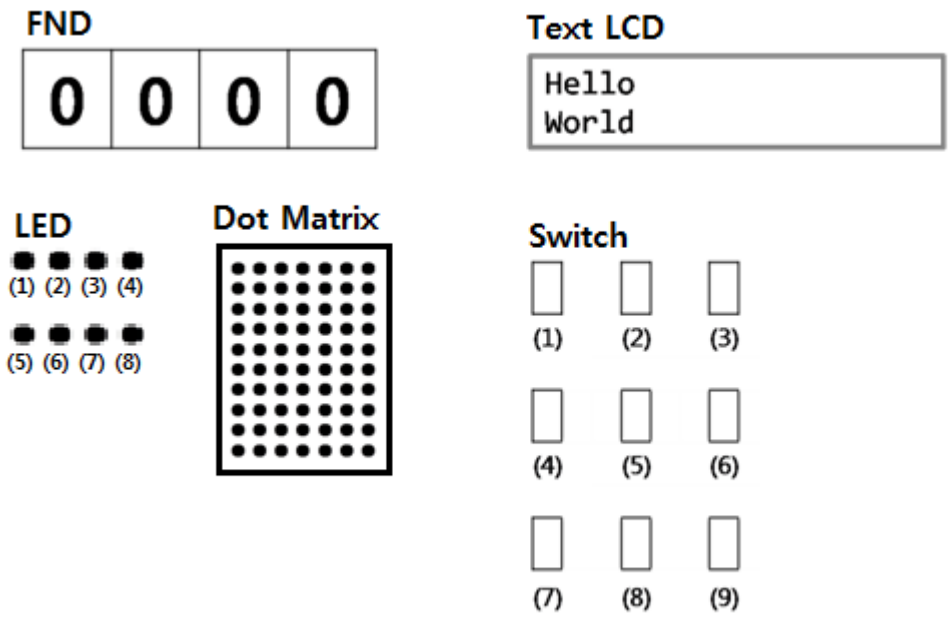


Figure 2. fpga 모듈 구성 요소

- FND : 시간을 출력한다. (앞 두 자리는 시간(24시간), 뒤 두 자리는 분(60분)). **초기 상태는 보드의 시간**이다.
- LED : **초기 상태는 (1)번 LED 에만 불이 들어온 상태**이다. 보드 시간을 변경하기 시작하면 (3)번 (4)번 LED에 불이 들어오고 1초에 하나씩 번갈아 가면서 불이 들어오게 한다. (한 버튼을 1초씩 켰다 껐다를 반복) 보드 시간 변경이 끝나면 (1)번 LED 에만 불이 들어오게 한다.
- SW(1) : 보드의 시간을 변경할 수 있도록 하고, 변경된 시간으로 저장하는 버튼
- SW(2) : 보드의 시간으로 reset 시켜주는 버튼이며 시간 변경 시 사용한다.
- SW(3) : 시간을 1시간 증가 시키는 버튼이며 이후 SW(1)을 눌러야 변경 사항이 저장된다.
- SW(4) : 분을 1분 증가 시키는 버튼이며 이후 SW(1)을 눌러야 변경 사항이 저장된다.
- **언급이 없는 나머지 디바이스는 빈 화면, 혹은 불이 모두 꺼진 상태**
  (나머지 모드에서도 동일하게 적용)

2) Counter – 모드 2

- FND : 카운팅 된 숫자를 출력한다.십진수의 숫자는 끝의 세 자리에만 입력된다. **초기 상태는 0000** 이다.

  * 999를 넘어갈 경우, 천 단위 이상의 출력은 생략한다. (EX. 1000→ 0000만 출력) 그러나 실제 값은 계속 유지되어 진수 변환 등에 적용되어야 한다.

  * 각 자리의 9에서 1이 증가하면, 앞자리의 숫자가 증가한다. (EX. 0090→ 0100)

  * SW(1)를 누르면 십진수로 입력된 수가 진수 변환을 한다.

- LED : **초기 상태는 (2)번 LED 에만 불이 들어온 상태**이다. 각 LED는 진수를 나타낸다. 10진수는 (2)번, 8진수는 (3)번, 4진수는 (4)번, 2진수는 (1)번 LED이다. 진수가 바뀔 때마다 LED 또한 바뀐다.

- SW(1) : 진수 변환을 하는 버튼이며, 그 순서는 십진수, 8진수, 4진수, 2진수이다.

- SW(2) : 백의 자리 숫자를 1 증가시키는 버튼이다.

- SW(3) : 십의 자리 숫자를 1 증가시키는 버튼이다.

- SW(4) : 일의 자리 숫자를 1 증가시키는 버튼이다.

* EX) 다른 진수일 때도 SW(2)~SW(3)키를 누르면 각 자리 수는 다음과 같이 증가함.

  [8진수 : 123 -> SW(2) -> 223]

* EX) 다른 진수일 때도 증가에 의해 앞의 자리수가 증가할 수 있음.

  [8진수 : 127 -> SW(4) -> 130]

3) Text editor - 모드3

- FND : fpga의 Switch가 현재 text를 입력하기 위해 몇 번 눌렸는지 count한 값을 출력한다. **초기 상태는 0000** 이다. 9999를 넘어갈 경우, 만 단위 이상의 출력은 생략한다. (EX. 10000→0000만 출력)

- SW : 알파벳과 숫자를 입력 받는 버튼이다.

(1) .QZ	(2) ABC	(3) DEF
(4) GHI	(5) JKL	(6) MNO
(7) PRS	(8) TUV	(9) WXY

초기 상태는 알파벳 입력이다.

<기본 기능>

새로운 버튼의 입력이 한 번 들어올 때마다 한 글자 씩 출력한다. 한번 눌렀던 버튼을 다시 누를 때마다 해당 알파벳을 입력 수에 맞게 바꿔준다. 즉 (2)번 버튼을 1번 누르면 A를 출력하고, (2)번 버튼을 3번 입력하면 C를 출력한다. (8),(9)번 버튼의 입력이 한꺼번에 들어오지 않으면 text LCD에 출력되는 string은 항상 덧붙여 출력한다.

(5), (6)번 버튼을 한꺼번에 누르면 알파벳 입력에서 숫자 입력으로 바꾼다. text LCD에 출력되는 string의 값은 변하지 않고, 기존에 입력된 text에 덧붙여 입력한다. 한 번 누를 때마다 해당하는 숫자를 출력한다. 숫자 입력 시에는 같은 버튼을 여러 번 누르면 버튼을 누를 때마다 새로운 숫자를 출력한다.

* EX) AMXB : (2)(6)(9)(9)(2)(2)

(2) (2) (2) (2) (5) (5) (4) (3) (1) (1) : AKGDQ

<버튼 2개를 동시에 누를 때, 각 기능을 수행 (FND count에 포함)>

(2), (3) : text LCD clear. 이 입력이 들어오면 text LCD에 있던 값을 없애고 다시 빈 상태로 만들어준다. 이 때 카운터를 제외한 나머지는 모두 초기 모드로 진입한다.

(5), (6) : 영어 → 숫자, 숫자 → 영어의 입력을 바꿔준다.

(8), (9) : 한 칸 띄운다.

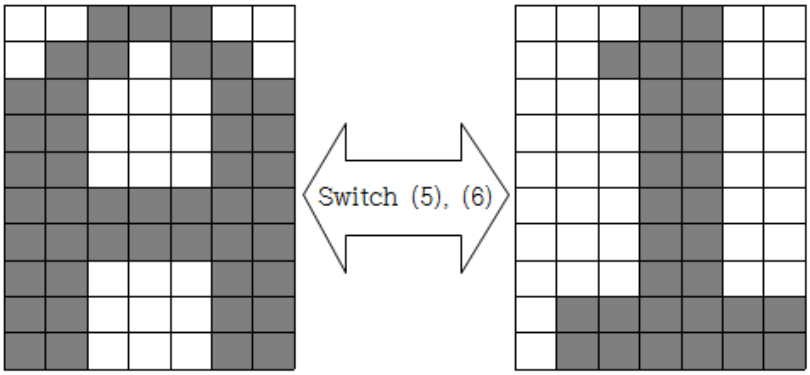
- Text LCD : 스위치를 통해 생성된 text 값을 출력한다. **초기 상태는 LCD가 빈 상태**이다. Text LCD의 최대 출력 범위를 넘어가면 기존의 string에 가장 앞에 있던 문자를 제거하고 한 칸 씩 앞으로 밀고 나서, 새로 들어온 text를 출력한다.

  (EX. Text LCD의 최대 출력 범위를 **8**이라고 가정하고 기존의 text를

ABCDEFGH라할 때, 9번째 text인 W가 들어오면 BCDEFGHW만 출력한다.)

- Dot Matrix : 현재 switch로 받는 입력이 알파벳인지 숫자인지 나타낸다. **초기 상태는 A**이다. 입력 받는 값이 알파벳이면 Dot Matrix에 A를, 입력 받는 값이 숫자면 Dot Matrix에 1을 출력한다.

* EX)



4) Draw board – 모드 4

- FND : fpga의 Switch가 몇 번 눌렸는지 count 한 값을 출력한다. **초기 상태는 0000**이다. 9999를 넘어갈 경우, 만 단위 이상의 출력은 생략한다. (EX. 10000→0000만 출력)

- SW : 스위치는 아래와 같은 기능을 담당하여 Dot Matrix에 그림을 그린다.

(1) reset	(2) ↑	(3) 커서
(4) ←	(5) 선택	(6) →
(7) clear	(8) ↓	(9) 반전

SW(2), (4), (6), (8) : 방향키로 동작한다. 커서를 이동하는데 사용된다. 커서의 위치는 rotation 기능을 지원한다. (EX. 만약 커서가 맨 윗줄일 때 SW(2)가 입력되면 커서는 맨 아랫줄로 이동한다.)

SW(5) : 현재 커서가 위치한 자리를 선택하여 Dot Matrix에 나타나게 한다.

SW(1) : 현재 그리고 있는 그림을 지우고 초기 상태로 reset한다.

SW(3) : 커서 출력 ON/OFF 기능을 담당하는 버튼이다.

* **초기 상태는 커서를 왼쪽 첫 줄에 표시하여 1초마다 깜빡이는 상태**이다.

* ON일 경우 커서는 1초마다 깜빡인다.

* OFF일 경우에도 커서는 보이지만 않을 뿐, 방향키로 위치를 계속 움직일 수 있다.

SW(7) : 현재 그리고 있는 그림을 Clear. 현재 커서의 위치나 커서 표시등은 바뀌지 않는다.

SW(9) : 현재 그림을 흑백 반전시킨다.

- Dot Matrix : 그림을 출력한다. **초기 상태는 커서를 왼쪽 첫 줄에 표시하여 1초마다 깜빡이는 상태**이다.

4. 제출 방법

1) 제출 파일

- **소스코드, Makefile, Readme, 보고서** 파일을 포함한 **[HW1]학번.tar.gz**
(EX. [HW1]20001234.tar.gz)

- 파일 압축 방법: 학번 폴더를 생성하고, 그 안에 숙제 관련 파일을 저장한 뒤, 학번 폴더가 있는 위치에서

tar –cvzf [HW1]학번.tar.gz ./학번폴더

2) 사이버 캠퍼스에 업로드

3) 제출 due date : 2021년 4월 15일 23시 59분까지

5. 평가 기준

1) 프로그램 80% (주석 점수 포함), 보고서 20%

2) Late : 하루에 10% 감점 / 5일 이상 late시 0점

3) 제출 형식 틀린 경우 10% 감점

4) Copy 적발 시 0점