**임베디드 시스템 소프트웨어 HW#1**

**학번 : 20171670**

**이름 : 이재훈**

1. **구현 아이디어 및 작동 방법**

* **2017670폴더 내 make을 하면 build디렉토리가 생긴다. 그 디렉토리 안에 test라는 실행 파일이 생긴다. 그 test 실행 파일을 adb push build/test /data/loca/tmp을 사용하여 minicom으로 넘겨준다.**
* minicom에서 /data/local/tmp 폴더로 들어간다. sh fpga\_insmod.sh 명령어를 입력해준 후 test라는 실행파일을 실행한다.
* 실행하면 main에서 총 세 개의 프로세스를 fork한다. 각각의 목적은 input, output 그리고 대부분의 작업들을 처리하는 핵심인 main process이다.
* Semaphore를 사용하여 main-input, main-output 프로세스 간 데이터를 사용하고 있을 때 바꿀 수 없게 한다.
* 각 프로세스 및 기능 구현은 밑에서 코드와 함께 진행을 하겠다.

1. **코드 설명 및 기능 구현.**

* Main.h

텍스트이(가) 표시된 사진

자동 생성된 설명

Shared memory를 구현하기 위해서 key값이 필요하다. KEY\_INPUT과 KEY\_OUTPUT은 각각 input과 output 데이터를 저장하기위한 shared memory 공간을 할당 받기위한 키 값들을 0x40과 0x50으로 지정을 해주었다.

순서대로 buffer의 사이즈와 switch button이 9개이기에 max\_button을 9로 설정해주었다.

KEY\_PRESS와 KEY\_RELEASE의 경우 switchkey나 readkey값이 눌러졌을 때를 확인하기 위한 것이다. 그 밑의 readkey들은 디바이스 내에서 return해주는 값들이다.

Semaphore의 경우 총 2개가 필요하다. Input-main사이, output-main사이, 이렇게 총 2개가 필요하기에 SEMA\_MAX를 2로 설정해주었고 세마포어를 구현하기 위해 키 값이 필요한데 0X47로 설정해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

SHM\_INPUT에선 input에서 데이터를 가져오기 위한 구조체이다. Readkey와 switchkey는 각각 입력이 되었을 때 어떠한 키가 입력되었는지 정보를 넣어주는 변수다. Check\_terminate의 경우 terminate명령이 떨어지면 set하여 전달하기 위한 변수이다.

SHM\_OUTPUT은 FND, dot, led, lcd에 출력하기위한 데이터를 저장해주는 변수들을 가지고 있다. Check\_terminate의 경우 input과 동일하게 set되었을 때 더 이상 명령을 수행하지 않고 종료하는 것을 알리기 위한 변수이다.

밑에 sem\_id와 p,v배열의 경우 semaphore를 구현하기 위해 존재한다. Sem\_id는 세마포어를 하기 위한 id이다.

**-main.c**

텍스트이(가) 표시된 사진

자동 생성된 설명

Shm\_input과 shm\_output의 경우 input과 output의 데이터를 저장하기위한 공간을 할당 받는데 그 크기는 앞서 설명한 input,output 구조체의 크기만큼 할당을 받는다.

Sem\_id값의 경우 2개의 세마포어를 semget을 통하여 지정받은후 return값으로 넣어준다. 세마포어를 초기에 1로 초기화 해주며 sem\_num이 0일 경우 input-main간, 1일 경우 output-main간 세마포어로 구분해준다. P와 v가 불릴 때 val값에 더해줄 값을 넣어준다. P의 경우 처음 critical section으로 진입할 때 -1을 하며 v의 경우 critical section에서 나올 때 불리는것이므로 +1로 설정해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

여기서 input, output, main process를 fork해주어야 하기 때문에 총 3개의 pid를 저장할 변수가 선언된다. 또한, 이 세개의 프로세스들이 끝날때까지 main함수가 종료되면 안되기 때문에 wait을 총 3번 불러준다. fork되었을 때 각 pid가 0일 때 각 프로세스의 메인 함수를 호출한다. 또한 pid의 값들이 모두 존재할 경우는 main프로세스 뿐이기 때문에 shared memory와 semaphore를 관리할 수 있는 조건으로 세 개의 pid가 전부 존재할 경우로 하였다.

* **Input\_process.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

input\_process은 input에서의 메인함수이다. Operating\_readkey와 operating\_switchkey의 경우 각각 readykey와 switchkey에 입력이 들어온 경우 input\_process에서 넘겨받은 shm\_input을 통해 shared memory에 입력된 키 값들을 저장한다.

* **Input\_process.c**

텍스트이(가) 표시된 사진

자동 생성된 설명

main으로부터 넘겨받은 shared memory 주소를 사용하여 할당된 주소에 연결한다. 프로그램을 종료하라는 명령이 내려오기전까지 무한반복한다. Readkey의 경우 O\_NONBLOCK을 통하여 입력을 받을 때 블락되는 현상을 해결하였다. Operating\_readkey와 operating\_switchkey를 불러주어 readkey나 switchkey중 입력이 있는지 체크한다. 그럴 경우 shared memory에 데이터를 넣어주어야 하므로 p를 통하여 val값에 -1을 더하여 0으로 만든 후 현재 critical section에 진입한 것을 알려준 후 데이터 변경이 완료되었을 때 v를 통하여 val값에 +1을 더하여 1을 만든후 critical section을 나온 것을 알려준다. Terminate 명령이 들어와서 끝내야할 때 shared memory에 attach한 것을 떼고 파일들 또한 close하고 종료한다. Usleep의 경우 디바이스가 너무 빨리 반응하며 작업하기 때문에 걸어주었습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Readkey와 switchkey의 경우 입력받은 값들에 변화가 있을 때 shared memory내 데이터로 넘겨준다. Switchkey의 경우 두개가 한번에 들어올 수 있기 때문에 prev\_temp란 배열을 하나 더 두어서 전에 들어온 값과 다를 경우 추가하여준다. 이 것은 for문이 5000번 진행될 동안 들어올 경우 동시에 들어온 것으로 간주한다.

* **Output.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

Define된 것들은 출력을 위해 device 버튼들의 주소이다. Lcd, fnd, led, dot들을 출력하기위한 함수들이고 받은 데이터들을 통해 출력할 것을 전달해주는 output\_process가 있다.

**-output\_process.c**

텍스트이(가) 표시된 사진

자동 생성된 설명

Input과 비슷하게 먼저 shared memory에 attach한다. terminate 명령이 들어올 경우 1로 셋해주어 종료한다. 아닐 경우 무한루프를 돌며 들어온 데이터를 처리하는데 semop를 사용하여 critical section을 사용 및 변경할 때 진입하지 못하게 막는다. Usleep의 경우 디바이스가 너무 빨리 반응하며 작업하기 때문에 걸어주었습니다.

* **Main\_process.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

함수에서 주가 되는 main\_process와 각 모드에서 input 데이터를 받은 후 각 명령마다 알맞은 처리를 해준 후 output 데이터로 넘겨주는 함수들이다. 자세한 것은 main\_process.c에서 설명하겠습니다.

* **Main\_process.c**

먼저 main process내 메인 함수를 설명하고 각 모드를 자세히 설명하겠습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Input과 output 프로세스와 동일하게 main으로 받은 shared memory 키를 사용하여 attach 하여 사용한다. 또한 각 메모리들을 초기화를 해준 후 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Terminate 명령이 들어오기 전까지 계속해서 while문을 돌며 명령을 수행한다. 메인 프로세스의 경우 input과 output 메모리를 접근하고 변경하기 때문에 while 문 내부를 critical section으로 설정하여 세마포어를 사용하였다. 사람이 짧게 눌렀다 하더라도 디바이스가 인식하기엔 매우 긴 시간이기에 직전 readkey 값을 기억하는 readkey\_prev를 사용한다. Readkey가 들어오지 않을때는 -1로 설정된다. 만약 volume up과 volume down 리드키가 입력 될 경우 update\_mode라는 함수에서 처리한다. Back이 들어올 경우 종료를 하란 의미므로 terminate 변수를 set하고 input과 output 데이터에도 전달해준다. Usleep의 경우 디바이스가 너무 빨리 반응하며 작업하기 때문에 걸어주었습니다.

그후 now\_mode란 변수를 통하여 각 기능들의 함수들이 호출된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Update\_mode의 경우 들어온 리드키에 따라서 now\_mode를 이동시켜주는 함수이다. Now\_mode가 이동 후 그 모드에서의 초기상태에서 시작해야하기 때문에 각 기능들에 해당하는 변수들을 초기화해준다. 각 변수들의 쓰임새는 함수들 코드 내에서 설명 하겠습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 첫번째 기능인 clock 기능을 구현하기위한 global 변수입니다. Clock\_mode의 경우 시간을 변경하고 있을 때를 나타내는 변수입니다. Add\_For\_clock의 경우 본래 시간보다 얼마나 더 증가시켰는지를 저장하고 있는 변수입니다. Which\_switch의 경우 led 3을 킬지 4를 킬지 결정하는 변수입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Switch key 별로 set되었을 때의 기능들을 구현하였습니다. Add\_for\_clock의 경우 한시간을 추가할 경우 100, 1분일 경우 1을 추가합니다. 만약 백의 자리가 6을 넘을 경우 100을 더해주고 그 밑을 60만큼 뺀 것으로 초기화 하여줍니다. 또한 바꾼 시간을 표시해주는 변수인 print\_clock의 경우 2400을 넘을 경우 그 나머지만 출력하도록 설정해주었습니다. 수정하는 모드에서 1초마다 led가 바뀌면서 빛나야 하므로 시간을 재는 변수인 which\_switch를 사용하였습니다. 만약 수정 중인 것을 알려주는 변수인 clock\_mode가 set이 안되어 있을 경우 수행하지 않습니다. 수정모드일때마다 which\_switch에 1을 더해주면서 5보다 작을 경우 4번째를, 클경우 3번째를 빛나게 해줍니다. main에서 usleep(100000)을 하고 input과 output에서도 동일하게 하기 때문에 5를 기준으로 하였습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Board time의 경우 localtime 함수를 사용하여 구해줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Counter 변수들입니다. counter내에서 몇 진수를 나타내는 counter\_mode와 카운터 수를 키억하는 counter\_num입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

첫번째 스위치가 눌렸을 경우 counter\_mode에 1을 더하여 진수를 바꿔줍니다. Convert\_base는 진수를 바꿔주는 함수인데 밑에서 설명하겠습니다. 나머지 switch를 눌렀을 땐 어느 자리에 +1을 해야하는지 알려주며 digit\_update라는 함수를 호출합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

진수에 따라 더해줄 수를 temp에 저장합니다. 밑에서 digit에 따라 temp를 다르게 더해줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Counter\_mode에 맞게 몇진수인지 저장하는 temp를 설정합니다. 그리고 진수에 맞게 현재 저장하고 있는 counter의 수를 변환하여 fnd\_data에 저장해줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Text\_editor를 구현하는데 필요한 변수들입니다. 먼저 입력을 위한 text\_board와 출력을 위한 fpga\_number 배열입니다. Text\_input의 경우 뭐가 입력되었는지를 나타내는 수이고 same\_cnt의 경우 같은 수가 몇 번 연속해서 들어왔는지 나타내는 변수입니다. Text\_mode의 경우 알파벳인지 숫자인지 알려주는 변수입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

너무 길어서 두 부분으로 나눠서 설명하겠습니다. 만약 두개의 스위치가 같이 눌러진 상태를 처리하여줍니다. Reset일 경우 다 초기화를 시켜줍니다. 만약 모드를 바꾸는것이면 dot에 모드에 맞게 표시할 것은 output\_data로 넘겨줍니다. Fnd\_data의 경우 키가 몇번 입력되었는지 저장하고 있는 데이터입니다. 텍스트이(가) 표시된 사진

자동 생성된 설명

만약 스페이스가 들어올경우 앞에 있는 것들을 한칸씩 앞으로 민 다음 스페이스를 추가하여 줍니다. 이 때 다른 입력이 들어온것이므로 same\_cnt를 초기화해줍니다.

이제 두개의 키가 눌린게 아닌 하나의 키가 눌렸을 때 처리를 해줍니다. Prev\_input을 저장해주어 전과 같다면 same\_cnt를 늘려줍니다. 또한 input이 들어온 것을 알리기 위해 input\_check를 set하여 줍니다. 만약 input\_check이 set되어 있고 알파벳 모드일 경우이며 전과 같을 경우 마지막 배열의 값을 변경 해주면 되지만 전과 다를 경우 다른 키를 새로 입력해야하기 때문에 그전의 값들을 앞으로 밀어주고 마지막에 추가합니다. 만약 배열이 꽉찼을 경우에 자연스레 앞의 것은 사라지게 됩니다. 숫자의 경우 같아도 바로 출력해야하기 때문에 앞에 것들을 한칸씩 밀어주며 마지막 칸에 그 숫자를 추가합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Draw board 기능을 구현하기 위한 변수입니다. Cursor\_mode는 커서가 깜빡이는지 안깜빡이는지 나타내며 board\_cnt는 지금까지 눌린 키의 횟수이며 display\_mode는 빛 반전입니다. Dot\_info의 경우 진행하면서 선택된 dot들을 가지고 있는 변수입니다. Blink는 1초마다 커서가 깜빡여야 하므로 그 시간을 나타내는 변수이며 now\_i와 now\_j의 경우 커서의 현재 위치입니다. Max\_dot의 경우 dot을 output\_data로 넘겨줄 때 bit로 되어 있으므로 0~6까지 모든 bit가 set되어 있는 변수입니다. 넘겨 줄 때 dot\_info와 &연산을 통해 1로 set되어 있는 위치를 알려주는 역할을 합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

Blink의 경우 위에서 설명한 clock에서 깜빡이는 것과 같은 조건으로 작동합니다. 5이상이면 빛을 안내고 5미만의 경우 set되는것으로 설정합니다. Switch 2,4,6,8이 들어올 경우 각 방향에 맞게 now\_i,now\_j를 조정합니다. 만약 경계선을 나갈 경우 반대편으로 옮길 수 있도록 if문으로 초기화 해주었습니다. 1번 버튼의 경우 모든 것을 리셋해줍니다. 3번 버튼의 경우 깜빡이는 것을 설정하는 것이므로 현재와 반대로 설정되도록 1에 cursor\_mode를 빼주어 반대로 설정해주었습니다. 5번째 버튼이 눌렸을 경우 선택 된것이기에 select 변수를 set해주어 후에 처리해줍니다. 7번의 경우 지금까지 선택된 dot들을 초기화 해주는 것이므로 1번 리셋과는 다르게 선택된 dot을 저장하는 dot\_info만 리셋해줍니다. 9번이 눌렸을 경우 표시하는 것을 반전 시켜주어야하므로 1에 현재 display mode를 빼주어 저장하였습니다. 밑에서 select가 set되어 있을 경우 현재 위치의 dot\_info 비트에 1로 set하여 줍니다. 그 후 output\_data에 dot\_info를 넘겨줍니다. 이 때 blink가 5를 초과하거나 cursor\_mode가 set되어 빛을 내는 상태가 아니라면 output\_data에 현재 위치의 비트를 꺼줍니다. 또한 display\_mode가 set되어서 선택된 것들이 빛을 내지 말아야한 다면 아까 설정한 max\_dot과 xor을 합니다. 왜냐하면 현재 선택된것들과 선택되지 않은 것들이 display하는 방식이 반대가 되어야하기 때문입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Makefile의 경우 build라는 디렉토리를 만든 후 그 안에 test라는 실행파일을 만들어주었다.