**Embedded System Software**

**HW1**

과목 명 : 임베디드 시스템 소프트웨어

담당 교수 : 박성용 교수님

20141544 안시현

개발 기간 : 4/2 ~ 4/14

**프로젝트 제목 : hw1**

**제출일 : 4/15**

1. **개발 목표**

* **디바이스 컨트롤과 IPC를 이용하여 주어진 clock, counter, text editor, draw board를 구현한다. 4가지 모드의 구현을 하고, 자유로운 주제로 fpga모듈을 이용하여 추가구현을 한다.**

1. **추진 일정**
   1. **추진 일정**

* **4/2~ 4/6 : fork , message queue 구현**
* **4/6 ~ 4/8 : 모드1, 모드2 구현**
* **4/4 ~ 4/10: 모드3 구현**
* **4/11 : 모드 4 구현**
* **4/12 : 추가구현 및 보고서 작성**

1. **연구 결과**
   1. **구현한 함수**

20141544.h (include한 header는 제외하겠습니다)

1. **사용 한 전역 변수 설명 및 macro 설명**

int pre;

// 전에 입력된 버튼을 저장하여 한 버튼을 꾹 누루고 있었을때 fnd count가 중복으로 올라가는것을 방지

int check;

// 전에 입력된 버튼을 저장하여 한 버튼을 꾹 누루고 있었을때 fnd count가 중복으로 올라가는것을 방지

int terminate;

// back button이 눌리면 terminate 변수를 1로 설정하여 종료를 위한 flag로 사용

unsigned char quit;

// switch 에서 버튼 입력 종료를 위한 flag

/\* mode 1 \*/

int time\_mode; // mode 1에서 시간 변경 모드인지 아닌지를 판단하기 위해 사용

unsigned char last\_clock[4]; // 현재 fnd 시간 저장

/\* mode 1 end \*/

/\* mode 2 \*/

unsigned char mode2\_flag;

int digit; // 현재 진법이 몇진법인지 저장

unsigned char last\_num[4]; // 현재 fnd의 number 저장

/\* mode 2 end \*/

/\* mode 3\*/

int previous\_input;// mode 3에서 전이랑 같은 input이 들어 왔을 시 문자를 바꿔주기 위한 변수

int previous\_idx; // mode 3에서 전이랑 같은 input이 들어 왔을 시 문자를 바꿔주기 위한 변수

char text[9][3]; // 문자판을 저장하기 위한 배열

char str[33]; // LCD 출력을 위한 string

char number[10]; // 0~9까지 숫자를 담고있는 배열

int button\_cnt; // button이 몇개 눌렸는지를 판단

int num\_mode; // 숫자 모드인지 , 영어 입력 모드인지 판단

int len; // 현재 lcd에 몇 글자가 입력되어 있는지를 저장하는 변수

/\*mode 3 end\*/

/\* mode 4 \*/

char map[10][7]; // 각 칸이 눌렸는지, 안눌렸는지 저장하는 배열

int x; //현재 dot이 입력돼야할 x 좌표

int y; //현재 dot이 입력돼야할 y 좌표

/\* mode 4 end \*/

/\* switch나 key를 위한 macro \*/

#define KEY\_RELEASE 0

#define KEY\_PRESS 1

#define BUFF\_SIZE 64

#define MAX\_LCD\_BUFF 9

#define MAX\_BUTTON 9

#define BACK 158

#define MODE\_PLUS 115

#define MODE\_MINUS 114

/\* switch나 key를 위한 macro \*/

/\* 파일 open을 위한 define macro \*/

#define FND 0

#define LED 1

#define LCD 2

#define DOT 3

#define FND\_DEVICE "/dev/fpga\_fnd"

#define LED\_DEVICE "/dev/mem"

#define LCD\_DEVICE "/dev/fpga\_text\_lcd"

#define DOT\_DEVICE "/dev/fpga\_dot"

/\* 파일 open을 위한 define macro \*/

/\* LED mmap을 위한 macro와 변수\*/

#define FPGA\_BASE\_ADDRESS 0x08000000

#define LED\_ADDR 0x16

unsigned long \*fpga\_addr;

unsigned char \*led\_addr;

/\* LED mmap을 위한 macro\*/

/\* msg queueing을 위한 key, switch, FND, LED, LCD, DOT 선언\*/

1. **구조체 및 구조체변수**

typedef struct \_key{

long msgtype;

int type;

int value;

int code;

int seq;

}key\_queue;

typedef struct \_swtich{

long msgtype;

unsigned char push\_sw\_buff[MAX\_BUTTON];

int button\_num;

}switch\_queue;

typedef struct \_FND{

long msgtype;

unsigned char data[4];

}fnd;

typedef struct \_LED{

long msgtype;

unsigned char data;

int status;

}led;

typedef struct \_LCD{

long msgtype;

char string[33];

}lcd;

typedef struct \_DOT{

long msgtype;

char matrix[10];

char dotmatrix[10][7];

int status;

int x;

int y;

}dot;

key\_t key\_id;

fnd send\_fnd;

led send\_led;

lcd send\_lcd;

dot send\_dot;

dot rcvdot;

int dev[5];

/\* msg queuing을 위한 key, switch, FND, LED, LCD, DOT 선언\*/

1. **함수 설명**

/\* msg를 주고 받는 함수들 \*/

void listen();

void \*receive\_from\_key();

* Main process에서 이용되며 Key를 받아주는 함수

void \*receive\_from\_switch();

* Main process에서 이용되며 Switch를 받아주는 함수

void \*rcv\_msg\_for\_led();

* Main process에서 이용되며 Led를 받아주는 함수

void \*rcv\_msg\_for\_fnd();

* Main process에서 이용되며 fnd를 받아주는 함수

void \*rcv\_msg\_for\_lcd();

* Main process에서 이용되며 Lcd를 받아주는 함수

void \*rcv\_msg\_for\_dot();

* Main process에서 이용되며 Dot을 받아주는 함수

void \*key\_msg();

* Input process에서 이용되며, key를 받아주는 함수

void \*switch\_msg();

* Input process에서 이용되며, switch를 받아주는 함수

void \*send\_time();

* Input process에서 이용되며, time을 계산해주는 함수

void send\_msg();

void user\_signal1(int sig);

/\* msg를 주고 받는 함수들 \*/

/\* mode별로 함수를 선언 \*/

void(\*fp[5])(int, unsigned char[]); // mode함수의 주소를 담고있는 function pointer

int mode; // 현재 mode를 저장하는 변수

void mode1(int n,unsigned char[]);

* Clock 기능을 구현한 함수로, time\_mode 변수에 따라 시간 변환 모드와 기본 모드를 전환하고 last\_clock[4] 변수에 현재 시간을 계속 적어주어 1분이 지날 때 마다 증가시켜 구현하였습니다.

void mode2(int n,unsigned char[]);

* Mode 2를 구현한 함수로, digit 변수를 통해 현재 진법을 저장하고 change\_digit 변수를 통하여 진법을 변환해 주고 output process로 msg를 전달합니다

void mode3(int n,unsigned char[]);

* Mode3을 구현 한 함수로 str[33] 변수에 현재 string을 저장하고 previous\_input 변수를 통해 같은 변수가 입력 되었는지 아닌지를 구분하였습니다. String이 완성되면 msg를 output process로 보내주었습니다.
* 또한 현재 버튼이 눌린 횟수를 count에 저장하여 fnd를 증가시켜 output process로 보내 주었습니다.

void mode4(int n,unsigned char[]);

* Mode4를 구현한 함수로 전역변수 x,y를 이용하여 현재 위치를 저장하고 버튼이 입력 될 때마다 위치를 변경해 주었습니다. 그래서 send\_dot 변수의 dotmatrix 변수를 업데이트 시켜 주고 그것을 output process로 보내서 출력 하였습니다. 이때, cursor mode와 non-cursor mode는 state를 통하여 구분 해 주었습니다.
* Mode3과 같이, fnd count를 두어서 버튼이 눌린 횟수 만큼 output process에서 button을 출력 할 수 있도록 하였습니다.

void mode5(int n,unsigned char[]);

* 현재 map에서 탈출 하는 것을 추가 구현 하였습니다. Stage 변수는 현재 stage 값을 담고 있으며 stage에 따라 map이 바뀝니다. 현재 map이 벽이면 1, 아니면 0으로 표현하여 벽인 곳에 도달 하였을 시에 game over가 되고 다시 재시작 되도록 구현 하였습니다.

/\* mode별로 함수를 선언 \*/

/\* initialize 함수들 \*/

void init\_main();

* Main initialize 함수로 각각의 device의 msg type을 정해주고 전역 변수들을 적절한 값으로 초기화 해주는 함수입니다.

void init\_output();

* Output process를 초기화 하는 함수로 각각의 device의 msg type을 정해주고 전역 변수들을 초기화 해주는 함수입니다.

void init\_input();

* Input process를 초기화 해주는 함수로 각각의 device의 msg type을 정해주고 전역변수들을 초기화 해주는 함수입니다.

void init\_led();

* Led 관련 변수들을 초기화 합니다

void init\_text();

* Mode3을 위한 text 배열을 완성합니다

void mode\_initialize(int mode);

/\* initialize 함수들 \*/

* Mode가 바뀌었을 때 fnd나 led , lcd 등을 적절하게 초기화 해 주는 함수입니다. 현재 mode값에 따라서 초기화 해줍니다

/\* 시간 관련 함수들 \*/

struct tm\* get\_board\_time();

* 보드의 현재 시간을 받아오는 함수입니다

void print\_cur\_time();

* 현재 시간을 출력해주는 함수입니다.

void print\_last\_time();

* 시간이 변경되었을 시 출력해주는 함수입니다.

void set\_last\_clock();

* Last\_time을 초기화 해주는 함수입니다.

/\* 시간 관련 함수들 \*/

void led\_start(unsigned char data); // led 출력 함수로 , data로 전송된 값을 출력합니다

void change\_digit(); // 진법 변환 함수로 현재 수를 다음 진법 수로 변환합니다

void calculate\_matrix(int sec); // dot matrix를 계산해주는 함수입니다

void clear\_func(); // 각 모드에서 다른 모드로 갔을때 모든 resource들을 초기화 해주는 함수입니다

#endif

* 1. **추가 구현**

추가 구현한 모드는 주어진 map에서 탈출하는 게임입니다. 시작 위치는 제일 밑줄 cursor를 통하여 알 수 있고, mode4 처럼 2,4,6,8 switch를 방향키로 조작하여 이동할 수 있습니다. 이동 중 벽에 닿으면 죽게 되며 제일 상단 까지 벽에 닿지 않고 도달시 stage clear이며 다음 stage로 이동하게 됩니다.

Stage는 총 5개로 이루어져 있습니다. 또한 led를 통해 각각의 stage에 따라 불이 들어오도록 하였습니다.

버튼 7을 누르면 stage를 깨지 않고도 다음 stage로 이동 가능합니다.

1. **기타**
   1. **소감**

* **20141503 안시현 : 우선 첫 시작부터 굉장히 당황스러웠습니다. Fork도 해본적이 없었고 message queue 또한 처음 들어 보는 것이었기 때문입니다. 처음 이틀 정도는 사실상 프로젝트 파악에 시간을 소요 했습니다. 구글에 검색을 해보며 많은 정보를 얻을 수 있었습니다. 또한 처음에 버튼과 스위치를 동시에 받을 수 없어 많은 시간을 헤매었고, thread가 따로 필요하다는 것을 파악 하게 되기 까지 또한 많은 시간이 필요하였습니다. 그렇게 고생하여 구조를 완성한 뒤 부터는 그래도 수월하게 진행 할 수 있었던 것 같습니다.**