마이크로프로세서 설계실험

Term Project 보고서

LCD인터페이스 세계시계 설계

담당교수: 한병철

실험기간: 2021.05.26 ~ 2021.06.07

실험 조: 1조

조 원: 2016116261 양준혁

2017110853 이동엽

제출일자: 2021.06.11.(금)

목차

Ⅰ 구성원 및 역할 분담

구성원	역할 분담
이동엽	소프트웨어 전체 구조 작성 및 오류 해결
양준혁	하드웨어 출력 관련 코드 작성

Ⅱ 설계 목표 및 개발일정

ⅰ. 설계 목표

LCD 인터페이스를 기반으로 하며 시간과 국가 설정이 가능한 세계 시계 구현

ⅱ. 설계 방향

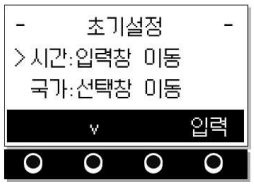
- LCD 기반 인터페이스 구동 코드 구축
- 재사용 가능한 코드를 통한 시스템 확장성 확보
- 내부 프로그램 구동코드 및 하드웨어 작동코드 모듈화

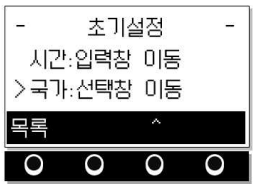
ⅲ. 개발 일정

5/21(금)	주제 설정 및 컨셉 결정, 전체 구조에 대한 초기계획서 작성 LCD 한글코드 자동화 코드 구축
5/26(수)	프로그램 기본 골격 및 핵심 구조체 코드 작성(LCD, 시간에 관한 구조체)
5/28(금)	프로그램 1/3 구축 및 기본 구조 작동 검사/디버깅
6/2(수)	프로그램 전체 작성 완료, 소프트웨어 및 하드웨어 연결을 위한 코드의 오류 해결
6/4(금)	interrupt 문제 해결 및 코드 구조 개선
6/7(수)	화면 출력 안정성 문제 해결, 코드 구조 개선, 시연 동영상 촬영

Ⅲ 프로젝트 세부기능설명

1. 초기설정화면





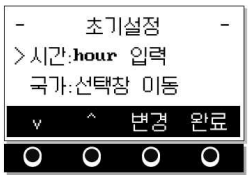
(1) 현재시간설정

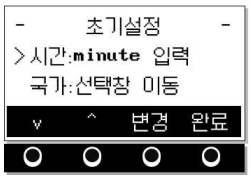
① 시간설정화면 이동

: ^, v 버튼으로 커서를 '시간: 입력창 이동'에 이동시키고 **입력** 버튼

② 현재시각을 사용자가 설정

: **변경** 버튼으로 hour/minute 설정모드 사이 전환, ^, v 버튼으로 시간변경





③ **완료** 버튼으로 초기설정화면으로 복귀

(2) 국가설정(도시선택)

① 국가목록화면 이동

: ^, v 버튼으로 커서를 '국가: 선택창 이동'에 이동시키고 **입력** 버튼

② 국가(도시)를 사용자가 선택

: ^, v 버튼으로 커서를 이동하여 국가(도시) 선택, **선택** 버튼으로 현재 커서가 위치한 국가가 선택된 초기설정화면으로 복귀

참조1 선택가능한 국가 목록

2. 메인화면

① 메인화면으로 이동

: 국가 설정이 끝난 초기설정화면, 현재시간설정 화면에서 **완료** 버튼으로 메인화면으로 이동

② 세계시계화면으로 이동

: ^, v 버튼으로 커서를 '1.세계시계'에 이동시키고 **선택** 버튼

③ 현재시간설정화면으로 이동

: ^, v 버튼으로 커서를 '2.현재시간설정'에 이동시키고 **선택** 버튼

3. 세계시간화면

① 세계시계

: ^, v 버튼으로 커서를 이동하여 국가(도시)의 시간 확인

② 메인화면으로 이동

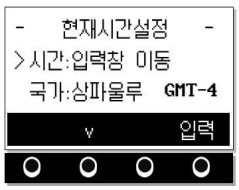
: < 버튼으로 메인화면으로 이동

참조2 세계시계화면에서 확인가능한 국가 목록

4. 현재시간설정화면

① 현재 시간과 선택된 국가 변경 가능

② 변경 방법은 초기설정화면과 동일함

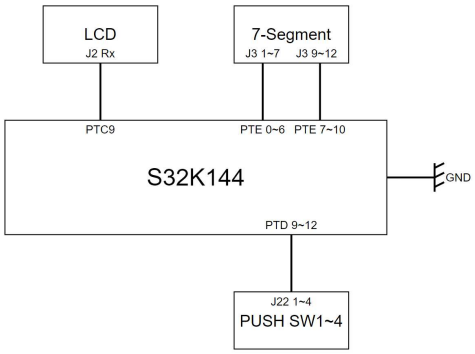


Ⅳ 하드웨어 구성 및 회로도

1) 입출력 하드웨어 구성 및 기능

입출력보드 장치		기 능
7-Segment		현재 시간 출력
LCD 화면		표시부/ 조작부기능 표시부
PUSH 버튼		조작부

2) 회로도



3) 결선 상태

	입출력보드	S32K144
LCD	J2-Rx	PTC9
PUSH	J22-1~4	PTD9~12
7-Segment	J3 1~7 J3 9~12	PTE0~6 PTE7~10

Ⅴ 소프트웨어구성 및 플로우차트

ⅰ 파이썬을 이용한 한글코드 생성 자동화

LCD 기반의 인터페이스를 유연하게 이용하기 위해서는 출력화면 변경이 간단하여야 하나, 입출력보드 상 LCD에 한글을 표시하기 위해서는 각 자모의 코드표를 보고 수작업으로 바꾸어야 하는 번거로운 작업을 거쳐야 한다. 해당 프로젝트에서는 이를 해소하기 위해 파이썬코드를 이용한 자동화를 진행하였다.

해당 코드는 whitespace로 구분된 한글 단어 목록이 있는 텍스트 파일을 입력받아 변환한 내용을 출력파일에 저장한다.

```
infile = open("in_KR_list.txt", 'r')
outfile = open("out_KR_LCD code.txt", 'w')
text = infile.read()
```

리스트에 저장된 단어들을 text.split(' ')을 이용하여 단어 단위로 분리하여 list에 저장한다. 단어는 for문으로 글자 단위로 분리, 글자는 jamo 패키지로 초성/중성/종성으로 분리한다.

한글 코드표에 있는 초성, 중성, 종성을 각각 string에 저장한다. 이때 string의 index와 LCD 코드가 일치하도록 한다.

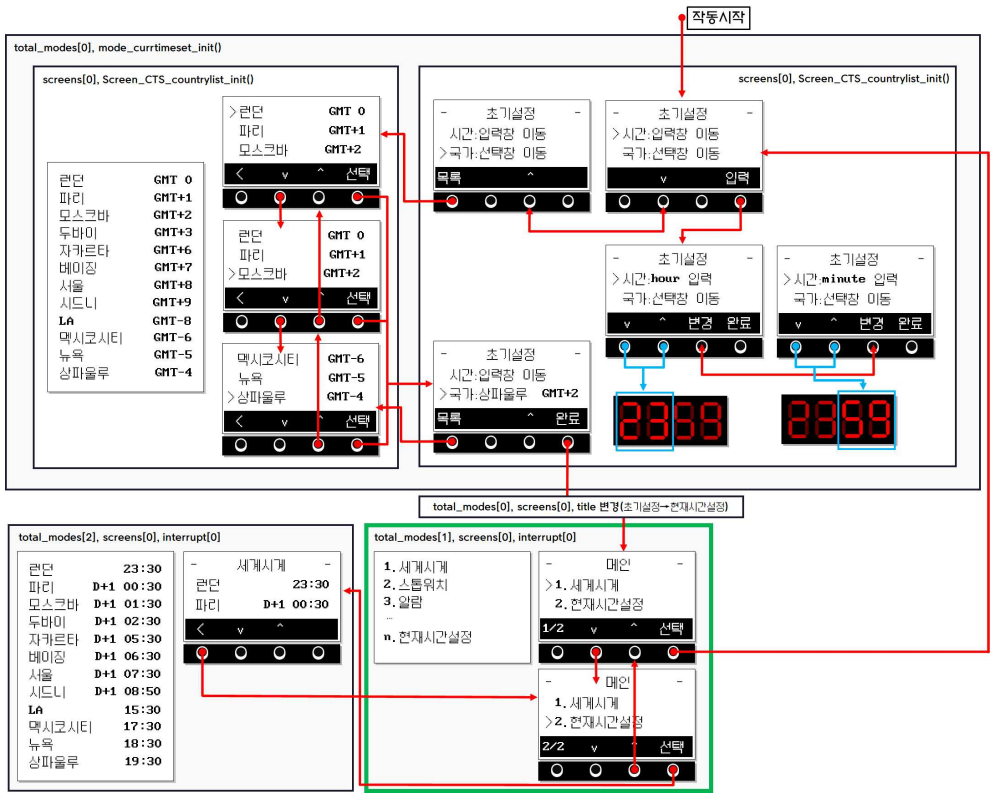
```
cho= "  ㄱㄲㄴㄷㄸㄹㅁㅂㅃㅅㅆㅇㅈㅉㅊㅋㅌㅍㅎ"
jung= "  ㅏㅐㅑㅒㅓ  ㅔㅖㅕㅗㅘㅙ  ㅚㅛㅜㅝㅞㅟ  ㅠㅡㅢㅣ"
jong= "  ㄱㄲㄴㄵㄶㄷㄹㄺㄻㄼㄽ  ㄾㅀㅁㅂㅄㅅㅆㅇㅈㅊㅋㅌㅍㅎ"
```

분리된 초성/중성/종성은 find() 함수를 통해 코드표 string에서 일치하는 초성/중성/종성의 index로 변환된다. 이 3가지 값을 2개의 hexcode로 변한 뒤, 그 hex코드는 해당 단어와 함께 출력 텍스트 파일에 출력된다.

코드화하고자 하는 단어를 입력 파일에 적고 코드를 실행하기만 하면 출력 파일에 해당 hex코드가 저장되기 때문에 한글을 코드화 과정을 간소화할 수 있고, 배열에 붙여넣기 좋은 형식으로 출력되므로 설계속도를 높일 수 있다.

ii 프로그램 구조

프로그램 동작 schematics



(1) main 함수 내 작동(프로그램 실행 함수)

main 함수 내의 코드를 최대한 가독성을 위해 각 작업을 함수로 묶어 사용하였다. 하드웨어 설정→시계설정(제외)→프로그램 빌드→프로그램 시작 순서로 실행된다.

(2) 프로그램 작동 구조 특징

1) 전역변수

해당 프로그램은 3가지 목적의 전역변수를 가지고 있으며, 모두 프로그램이 작동하는 데 중요한 역할을 하고 있습니다.

전역변수	목적
total_modes	모든 함수에서 프로그램 전체에 접근 가능
currentscreen, curr_modenum, curr_screennum, interruptnum	현재화면의 데이터 쉽게 지정 및 접근 하드웨어 입출력 함수 내부 간결화
프로그램 빌드용 문자열	발생문제 4)에서 다룰 예정

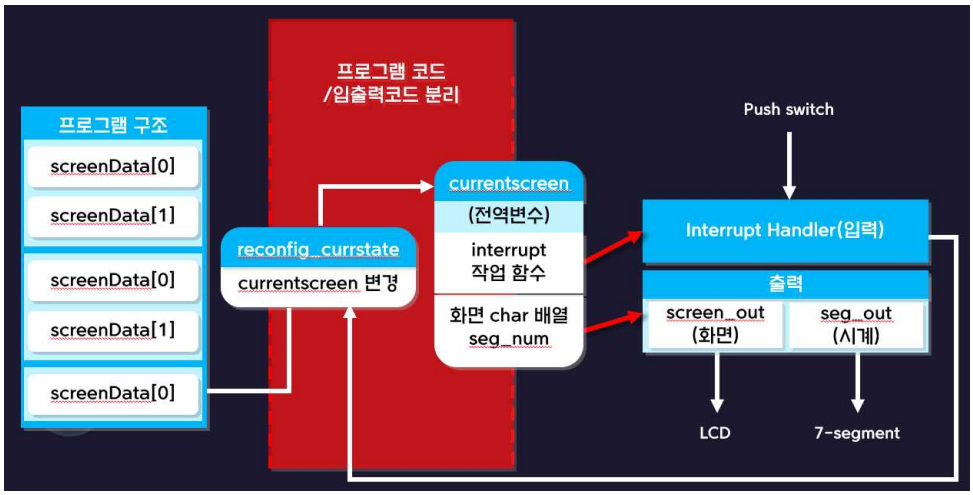
2) currentscreen & reconfig_currstate()

currentscreen은 화면구성 및 인터럽트 등의 정보를 담고 있는 screenData의 포인터 변수이다. 이름이 말해주다시피 currentscreen은 LCD 출력함수인 screen_out()의 input 매개변수가 되며, currentscreen 내의 interruptfunc이 현재 인터럽트 작업 함수로 실행되게 된다.
interrupt handler의 내부나, 프로그램 실행 코드 내부에 각 화면의 경우마다 서로 다른 화면 포인터를 넣은 screen_out()을 사용할 필요 없이, currentscreen에 할당하는 포인터만 변경하면 된다.

```
void PORTD_IRQHandler(void){
        currentscreen->inturruptfuncs[interruptnum]();
        screen_out(currentscreen);
        for(int i=1;i<13;i++) PORTD->PCR[i] |=0x01000000; // Clear the ISF bit
}
```

여기서 currentscreen과 curr_modenum... 등 현재상태를 변경하고 싶을 때 함수 reconfig_currstate()를 이용한다. 여기에 3개의 index값(mode, screen, interruptfunc)을 넣으면, 현재상태를 자동으로 바꾸어주며, 코드를 간결하게 하여 가독성을 높여 준다.

이렇게 내부 프로그램 측에서는 reconfig_currstate()만 사용하고, 하드웨어 입출력 관련 코드 측에서는 currentscreen만 읽어 들이면 되며, 둘 사이 코드가 서로 엮이지 않는 일종의 모듈화라고 볼 수 있다.



(3) clock_program_build()

프로그램을 실행시키기 위해서는 먼저 프로그램을 구성하는 데이터들의 구조를 잡아주어야 한다. 이를 수행하는 것이 clock_program_build()의 역할이다. 실질적으로 하는 작업은, 프로그램의 모든 데이터에 접근할 수 있는 포인터 배열 역할을 하는 total_modes에 메모리를 할당해 내부적으로 modeData와 screenData의 포인터 배열로 자리를 만든 다음, 프로그램의 재료라고 할 수 있는 전역변수로 선언된 배열과 인터럽트 작업 함수을 적절한 자리에 모두 할당하는 것이라고 할 수 있다.

(4) clock_program_run()

total_modes가 완성된 시점에서, 프로그램을 시작하는 것은 currentscreen에 적절한 포인터를 할당하는 reconfig_currstate()를 실행하고 screen_out()만 실행하면 된다.

```
void clock_program_run(){reconfig_currstate(0, 0, 0);screen_out(currentscreen);}
```

Ⅵ 설계결과 및 논의

(1) 설계 결과 체크리스트

항목		동작 여부	비고
프로그램 설계	파이썬을 이용한 LCD 한글 코드 생성 자동화	O	
	재사용 가능한 코드를 통한 확장성 확보	O	
	내부 프로그램 구동코드 및 하드웨어 작동코드 모듈화	O	
프로그램 기능	초기설정화면 구동	O	
	↳ 시, 분 설정	O	
	↳ 국가 목록 화면 구동	O	
	↳ 선택 국가 표시	O	
	메인화면 구동	△	발생문제 4)
	세계시계	X	

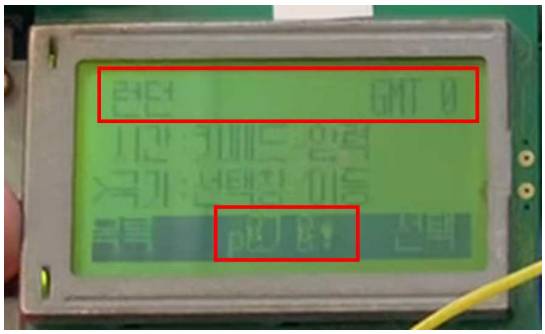
(2) 발생 문제 및 해결

프로젝트를 진행하는 과정에서 여러 가지 문제들을 마주하였다. 이 중에서 논의할 가치가 있는 몇 가지 문제와 이를 해결한 방식에 대해서 이야기 하겠다.

1) 포인터 할당 문제 : 초반에 코드를 작성하고 실험실에서 실행시킬 때 놀랍지 않게도 제대로 작동하지 않았다. 하지만 빌드를 통해서 오류가 없음을 확인했음에도 불구하고, 입출력보드에 아무 반응도 나오지 않는 경우 디버깅하기가 매우 번거로웠다. 이를 해결하기 위해 코드들을 주석처리 하며 점점 원인이 되는 코드를 확인하니, 포인터에 할당된 주소값이 잘못 되었거나, 포인터 변수에 메모리 할당이 되지 않았을 경우 일어나는 것을 확인하였다. 이로 인해 하드웨어가 없는 상태에서 코딩을 진행하기 어렵게 되었으며, 이 오류는 진행과정 중 수시로 발생하였다. 현재는 세계시계 모듈을 제외하고는 이를 해결하였다.

2) 메모리 관련 문제: 메모리 관련 문제가 처음 의심되었던 때는, 프로젝트 초반에 포인터 배열에 서로 다른 길이의 LCD 한글코드들을 할당하는 함수를 코딩하였을 때이다. 다중 포인터에 동적메모리할당을 이용하여 포인터 배열을 만들어준 뒤, 각각 다른 길이의 배열을 할당하는 코드로, 코드자체에는 문제가 없었다. 문제는 배열이 일정크기 이상 늘어나면 코드가 더 이상 작동을 안하고 멈추는 것이었다. 당시 이를 검토하는 과정에, 단순히 배열 길이와 관련된 문제임을 결론 내렸으며, 메모리에 한계가 있을 수 있다는 생각을 하게 되었다.
프로그램에서 메모리가 문제가 될 수 있다는 것을 확실히 알 수 있었던 것은 화면 안정성과 관련된 문제를 다루었던 때이다. 화면과 화면 사이를 전환하거나, 단순히 버튼 인터럽트가 일어날 때도 화면에 알 수 없는 문자열이 원래 출력대상 대신 출력되었다. 검토해본 결과, clock_program_build() 내부

의 대부분 변수들은 동적 할당을 이용한 반면에, 문자열은 함수 내에서 배열 형태로 선언한 뒤, 그 주소를 포인터에 넘기는 방식으로 작동하였다. 함수 내부의 지역변수로 선언한 탓에 함수의 호출이 중단되면서 해당 메모리가 free되며 다른 문자열이 덮어 씌워지는 것을 허용하면서 발생한 문제였다. 이는 문자열을 전역변수로 선언한 후에는 더 이상 잔상이 생기지 않고 화면이 안정되면서 확실히 확인하게 되었다.



잔상이 발생한 화면

안정화 완료 후 화면

3) screen_out()과 seg_out() delay문제: seg_out()은 무한 반복문을 이용하여 지속적으로 출력하여야 하는데, interrupt에서 screen_out()이 실행되면, screen_out()이 실행되는 동안 seg_out()이 출력되지 않는다. 이를 해결하기 위해, screen_out()의 내부에서 LCD에 글자를 출력할 때 마다 seg_out()를 실행하여, screen_out() 실행 중에도 7-segment의 출력이 멈추지 않도록 할 수 있었다.

4) mode_init() 함수 동시 실행 문제: 마지막으로 다룰 문제는 프로젝트에 매우 치명적이었으며, 현재도 원인을 알지 못하는 문제이다. 본 프로그램은 내부에 3가지 mode가 있으며, 세계시계를 제외한 초기설정(현재시간설정) 및 메인메뉴는 각각 실행하였을 때 정상적으로 작동함을 확인할 수 있었다. 하지만 이 2개의 모드를 동시에 실행하게 되면, 프로그램은 시작을 하지 않고 아무런 출력도 나오지 않는다. 그저 같이 실행하기만 하면 작동이 되지 않는다. 이 양상은 앞의 2)번 문제와 매우 비슷한 양상을 가지고 있기 때문에, 메모리 관련 문제로 인한 것이라는 추측만 할 수 있었다.
S32K144의 내부 용량을 초과한 것일지 모른다는 판단을 하였지만, 코드 실행 결과의 크기가 훨씬 작았으므로 이는 틀린 추측이었으며, 현재는 2)번 문제와 비슷한 메모리 충돌과 관련된 문제로 생각하고 있다.

(3) 프로젝트 자체 평가

초기 구상 단계에서 계획한 기능들 중 일부가 구현되었으나, 몇 가지 기능들은 구현하지 못하였다. 이에 대한 2가지 정도의 면에서 프로젝트 진행을 평가해 보았다.

1 첫째, 초기 구상 단계에서 주어진 시간 안에 완료하기 어려운 프로젝트를 계획하였다. 초기에는 알람, 스톱워치, 타이머 등 더 다양한 시계 기능들을 구현하고자 하였다. 그러나 본 프로젝트는 시간과 자원의 활용이 유한한, 다량의 코딩이 필요한 프로젝트와는 맞지 않는 환경에서 진행되었다. 이로 인해 구상했던 다른 기능들은 구현이 되지 못했고, 세계시계의 구조만 작성 및 일부 구현되었다. 프로젝트 초기 구성 단계에서 주어진 환경과 맞는 주제 및 내용을 정해야 한다는 점을 느끼게 되었다.

2 둘째, 첫째에서 언급했듯이 프로그램을 실행하기 위해서는 입출력회로보드, S32K144 등 장비가 필수적이었고, 이를 활용할 수 있는 시간은 제한적이었다. 프로그램에 다양한 기능을 추가하고 싶어 큰 규모로 시작한 프로젝트는, 시간적 요소를 고려하지 못한 채 진행되었다. 작성한 코드의 오류를 해결하는 것에 많은 시간이 투자되었고, 결국 구상했던 기능들이 모두 구현되지 못하였다. 시간적 요소를 고려하여 프로젝트의 개발일정을 짜야한다는 것을 느끼게 되었다.

Ⅶ 소스코드 : 첨부파일 참조

<참고문헌>

[1] 마이크로프로세서설계실험 1~11주 부록 및 강의자료

[2] https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=gauya&logNo=220129979980

[3] S32K1xx Series Cookbook Software examples and startup code to exercise microcontroller features(AN5413)

[부록1] 프로그램 구조체 및 함수 기능

1) 구조체

1. screenData : 화면을 구동하는 데 필요한 정보가 포함된 구조체

변수선언	기 능
char msg[3][23]	표시부 출력 데이터
char switch_config[23]	조작부 기능 표시 데이터
int seg_num	시간 관련 출력(7-segment)
void (**interruptfuncs)()	interrupt handler에서 실행될 '인터럽트 작업 함수' 배열
int current_interrupt	현재화면에서 이용하는 interrupt 작업함수 index
int screenNum	modeData 내에서 선언된 screen 배열 중 해당 screen index
void* others	기타 필요한 포인터 변수를 위한 void 포인터(screentype_listData 등)

2. modeData : 각 모드가 가진 화면에 대한 screenData*을 배열로 가지는 구조체

변수선언	기 능
screenData** screens	모드가 가진 screen들의 배열
int totalscreennum	모드가 가진 screen들의 개수
int modeNum	total_modes 내에서 선언된 modeData 배열 중 해당 모드의 index

3. screenType_listData : 화면 형식 중 list를 동작/출력하기 위해 필요한 정보를 가진 구조체

변수선언	기 능
char** list	화면에 표시될 list의 데이터가 저장된 2차원 문자 배열
char* title	표시부 상단에 표시될 제목 데이터
char* switch_config	조작부 기능표시에 표시될 데이터
int size	list의 크기(line 개수)
int height	화면에 표시되는 line 높이
int cursor_loc	화면상 커서가 있는 위치
int top_num	화면 최상단에 위치한 list line의 index
int has_title	제목 존재 여부
int selected	현재 커서가 가르키고 있는 list상 line의 index

(테스트/사용 안됨, 코드 상에서만 존재)

4. timeFormat : 현재 시간, 날짜 정보가 들어가게 되는 구조체

변수선언	기 능
int timescales[6]	~~year, month, day, hour, minute, second~~

5. timeSession : 시간, 날짜 정보와 이를 구동하기 위한 timer interrupt 들어가게 되는 구조체

변수선언	기 능
timeFormat*	~~year, month, day, hour, minute, second~~

void (*time_interrupt)()	시간 구동 함수(timer interrupt 내부 함수, 시계용)
int timenum	total_times 내에서 선언된 timeSession 배열 중 해당 모드의 index

6. interruptData : 기존 인터럽트 작업함수 할당에 비해 개선된 방식(테스트 못해봄)

변수선언	기 능
void (*interruptfinc)()	인터럽트 작동 함수 포인터
char* switch_config	인터럽트 작동 함수에 대응하는 switch_config

2) **도구 함수**(필요할 때 수시로 사용하는 함수들)

함 수	기 능
screen	
screen_out(screenData*)	screenData 내부의 화면표시 배열을 LCD에 출력
seg_out(int)	정수 4자리를 7-segment에 출력
reconfig_currstate(mode_num, screen_num, interrupt_num**)**	currentscreen과 current indices와 관련된 전역변수를 변경하는 함수
screenType_listData	
screenType_listData* **set_screenType_listData(** int size, height has_title, char *title, char *switch_config)	screenType_listData*를 생성한 뒤 입력된 정보로 초기화 뒤 반환
screentype_list(screenData*,screenType_listData*)	list에 저장된 데이터에 맞춰 screendata 내 화면배열 내용을 변경
screenType_list_operation_UP(screenType_listData*)	리스트를 위로 올리도록 listData 내 변수를 변경하는 작업을 수행
screenType_list_operation_DOWN(screenType_listData*)	리스트를 아래로 내리도록 listData 내 변수를 변경하는 작업을 수행
int **screenType_list_ rewriteLine**(screenType_listData* list, char* line, int lineNum)	리스트의 원하는 line의 내용을 변경 입력된 lineNum이 list size보다 클 경우 0 반환 그 외 작업을 수행하고 1반환
timeSession : 사용하지 않음	
timeFormat* **set_timeFormat**(int year, month, day, hour, min, sec)	timeFormat*를 생성한 뒤 입력된 정보로 초기화 뒤 반환
timeInt2timeFormat(timeFormat* tmp, int timeInt)	hour와 minute가 각각 2자리씩 있는 4자리 정수로 timeFormat 내부 시, 분을 변경
int **timeFormat2timeInt**(timeFormat* tmp)	timeFormat 내부 시, 분을 seg_out에 적절한 int값으로 변형하여 반환
int **timedelay_sec**(timeFormat* time, int delay)	timeform의 시간으로부터 delay초 후 시간을 계산
int **timedelay_sec_reverse**(timeFormat* time, int delay)	timeform의 시간으로부터 delay초 전 시간을 계산
currenttime_run()	timeform의 시간이 timer interrupt 1초마다 증가하는 시계구동함수

2) 구성 함수(프로그램 구성 함수, _init()계열 → 모두 clock_program_builder.h 파일에 존재)

1. mode_init() : modeData를 구성하는 함수

함 수	기 능
clock_program_build()	total_modes에 modeData 포인터 배열 동적할당
mode_currtimesetting_init()	현재시간설정(초기설정)에 필요한 screenData 메모리 할당
mode_mainscreen_init()	메인화면에 필요한 screenData 메모리 할당
mode_worldtime_init()	세계시계에 필요한 screenData 메모리 할당

2. Screen_init() : screenData를 구성하는 함수

함 수	기 능
mode_currtimesetting	
Screen_CTS_mainsetlist_init()	현재시간설정(초기설정)의 메인리스트 화면의 데이터 할당
screenType_listData_CTS_mainsetlist_listalloc_init(screenType_listData* temp)	현재시간설정의 메인리스트 화면의 screenType_listData에 list를 추가하는 함수
Screen_CTS_countrylist_init()	현재시간설정(초기설정)의 국가목록 화면의 데이터 할당
screenType_listData_CTS_countrylist_listalloc_init(screenType_listData* temp)	현재시간설정의 국가목록 화면의 screenType_listData에 list를 추가하는 함수
mode_mainscreen	
Screen_MS_mainlist_init()	메인메뉴 화면의 데이터 할당
screenType_listData_MS_mainlist_listalloc_init(screenType_listData* temp)	메인메뉴 화면의 screenType_listData에 list를 추가하는 함수
mode_worldtime	
Screen_WT_mainlist_init()	세계시계 화면의 데이터 할당
screenType_listData_WT_mainlist_listalloc_init	세계시계 화면의 screenType_listData에 list를 추가하는 함수

3. time_init() : 프로그램 내부 시계를 구성하는 함수

함 수	기 능
times_init(int n)	timeSession을 생성하여, 0으로 초기화하여 전역 포인터 변수 times에 할당

3) 구조 함수(프로그램 구조로 할당되며, 구성함수에 의해 total_modes 내에 배치되는 문자열/인터럽트 작업 함수 등, 모두 clock_program_builder.h 파일에 존재)

1. 문자열

변수명	기 능
mode_currtimesetting	
list1_currtimeset2_hour	시간설정 화면에서 변경버튼을 누를 때 hour 설정 모드로 변경될 때 list 문자열
list1_currtimeset2_minuit	시간설정 화면에서 변경버튼을 누를 때 minute 설정 모드로 변경될 때 list 문자열
list1_currtimeset1	메인리스트로 돌아갈 때 list 문자열
switch_config_currtimeset_currtimeset1	현재시간설정1 인터럽트 작업 함수 스위치 문자열
switch_config_currtimeset_currtimeset2	현재시간설정2 인터럽트 작업 함수 스위치 문자열
switch_config_currtimeset_countryset1	국가설정1 인터럽트 작업함수 스위치 문자열
switch_config_currtimeset_countryset2	국가설정2 인터럽트 작업함수 스위치 문자열

initial_setting_title	초기설정 화면 제목
currtimeset_title	현재시간설정 화면 제목
currtimeset_mainlist_list	현재시간설정 메인리스트 리스트 문자열
currtimeset_mainlist_country_list	현재시간설정 메인리스트 국가선택 후 리스트 문자열
currtimeset_countrylist_list	현재시간설정 국가목록 리스트 문자열
switch_config_countrylist	국가목록 인터럽트 작업함수 스위치 문자열
mode_mainscreen	
mainscreen_mainlist_list	메인화면 리스트 문자열
mainscreen_mainlist_title	메인화면 제목
switch_config_mainscreen_mainlist_countrylist	메인화면 인터럽트 작업함수 스위치 문자열
mode_worldtime	
~~worldtime_mainlist_list~~	~~세계시계 리스트 문자열~~
~~worldtime_mainlist_title~~	~~세계시계 제목~~
~~switch_config_worldtime_mainlist_countrylist~~	~~세계시계 인터럽트 작업함수 스위치 문자열~~

2. 인터럽트 함수

변수명	기 능
mode_currtimesetting	
inturrupt_currtimeset1()	현재시간설정1 인터럽트 작업 함수
inturrupt_currtimeset2()	현재시간설정2 인터럽트 작업 함수
inturrupt_countryset1()	국가설정1 인터럽트 작업 함수
inturrupt_countryset2()	국가설정2 인터럽트 작업 함수
inturrupt_countrylist()	국가목록 인터럽트 작업 함수
mode_mainscreen	
inturrupt_mainlist()	메인화면 인터럽트 작업 함수
mode_worldtime	
~~inturrupt_worldtime()~~	~~세계시계 인터럽트 작업 함수~~
~~time_worldtimes_run()~~	~~세계시계 작동 timer 인터럽트 함수~~