Implement Report

Team 5 –Jirung

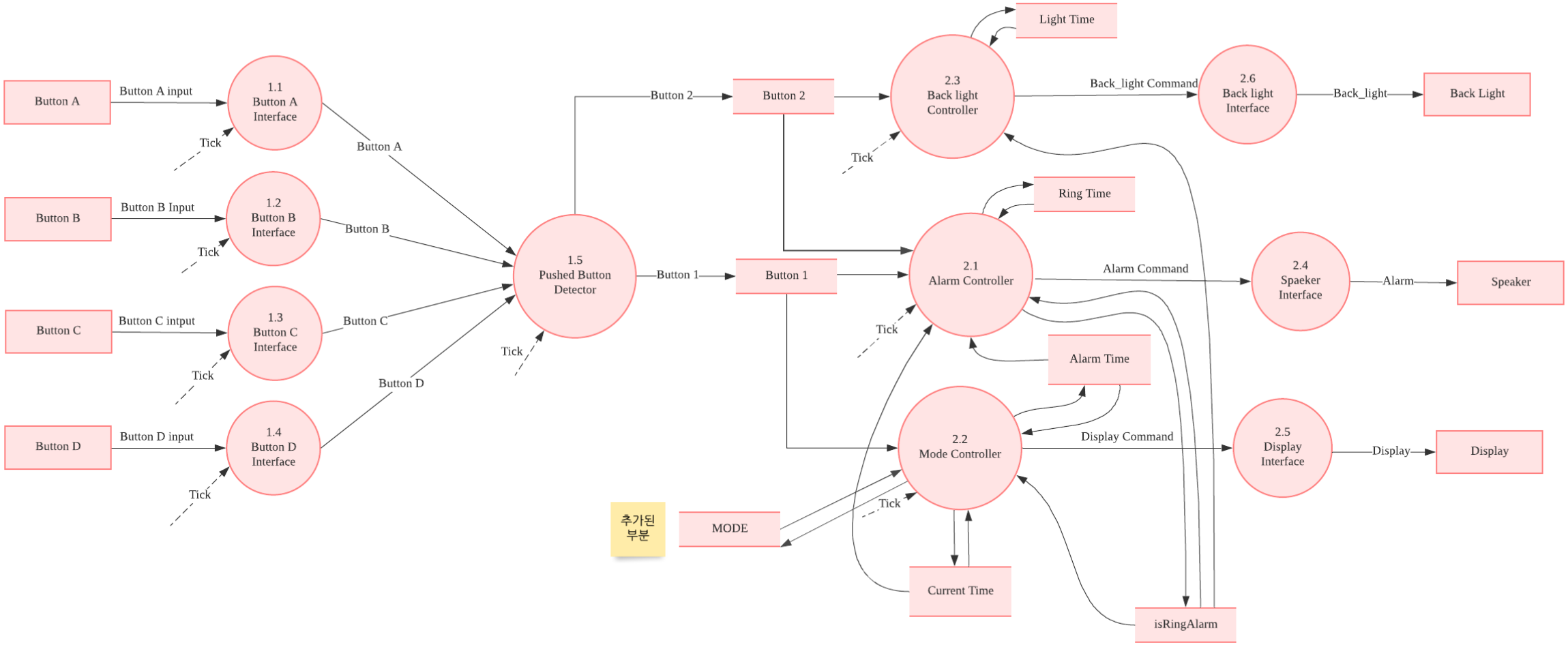
김하림201710326
박성민201811254
이동현201510240
정주원201711424

수정사항

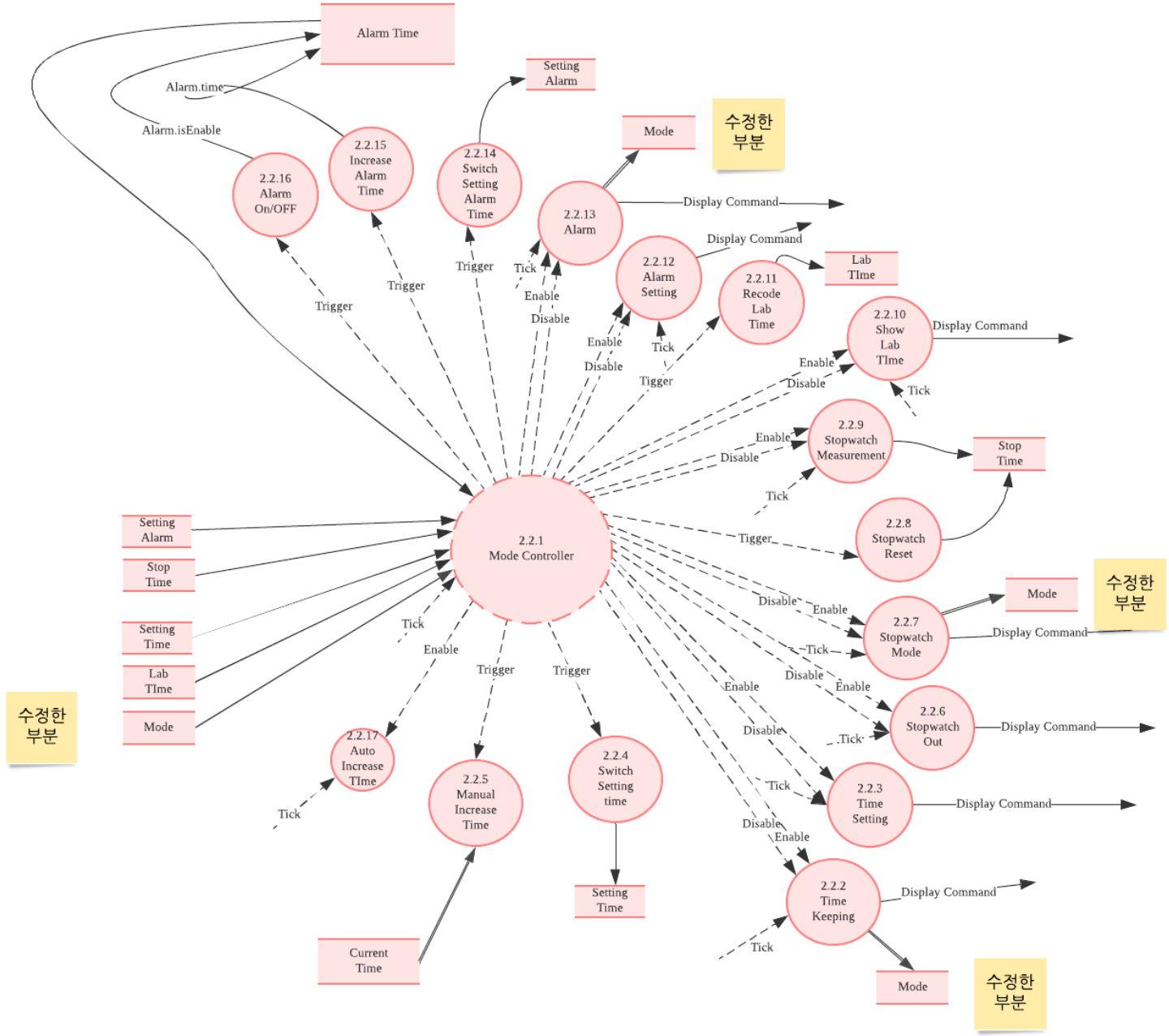
문제점 : mode를 변경할 때 현재 mode가 무엇인지 알 수 없었다.

수정사항 : mode를 저장할 수 있는 Storage를 추가하였다.

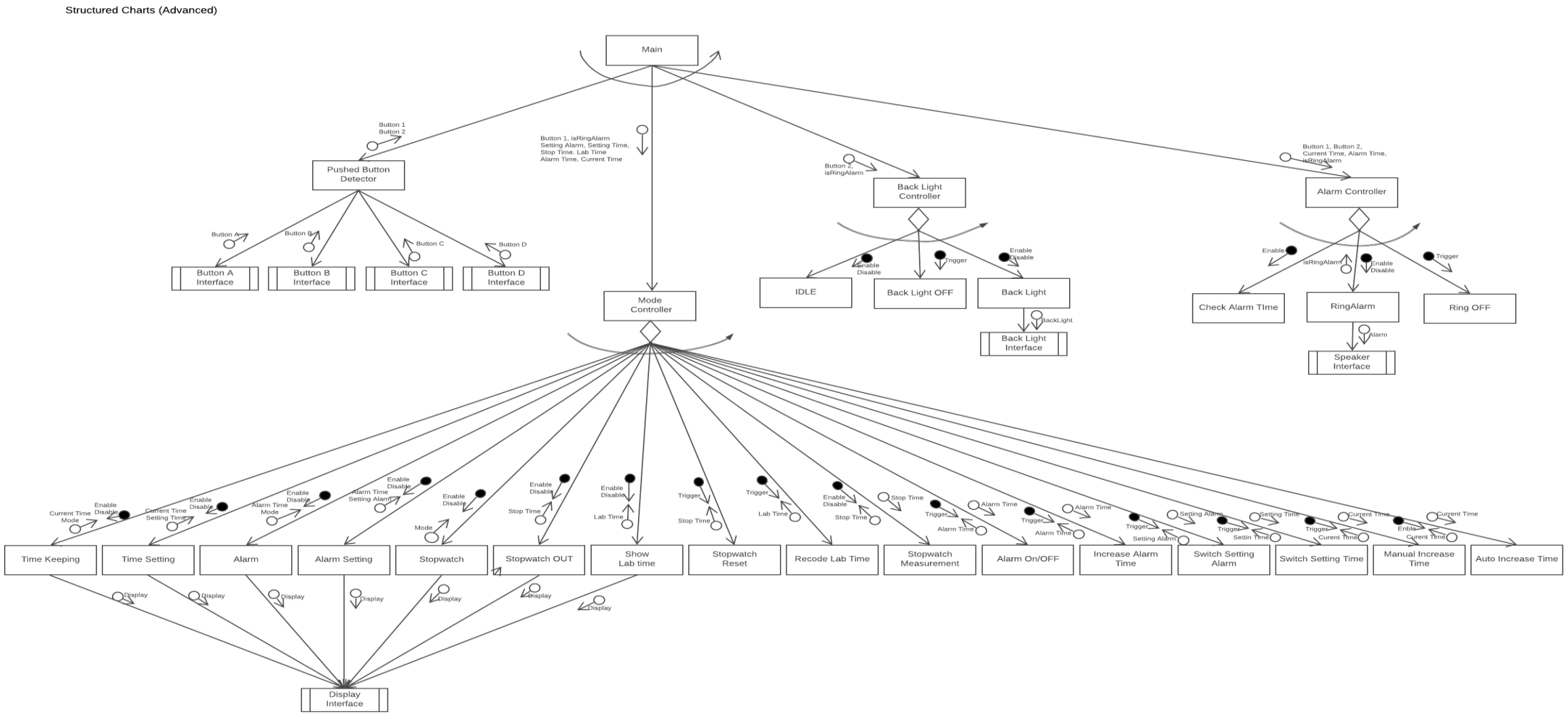
DFD L2 – v3.0


Button A
Button A input
1.1
Button A
Interface
Tick
Button A
Button B
Button B Input
1.2
Button B
Interface
Tick
Button B
Button C
Button C intput
1.3
Button C
Interface
Tick
Button C
Button D
Button D input
1.4
Button D
Interface
Tick
Button D
1.5
Pushed Button
Detector
Tick
Button 2
Button 1
Button 2
Button 1
2.3
Back light
Controller
Tick
Light Time
Back_light Command
2.6
Back light
Interface
Back_light
Back Light
2.1
Alarm Controller
Tick
Ring Time
Alarm Command
2.4
Spaeker
Interface
Alarm
Speaker
Alarm Time
2.2
Mode Controller
Tick
Display Command
2.5
Display
Interface
Display
Display
추가된
부분
MODE
Current Time
isRingAlarm

DFD L3- v3.0


Alarm Time
Alarm.time
Alarm.isEnable
Setting Alarm
Mode
수정한 부분
2.2.16 Alarm On/OFF
2.2.15 Increase Alarm Time
2.2.14 Switch Setting Alarm Time
2.2.13 Alarm
Display Command
2.2.12 Alarm Setting
2.2.11 Recode Lab Time
Lab TIme
2.2.10 Show Lab TIme
2.2.9 Stopwatch Measurement
Stop Time
2.2.8 Stopwatch Reset
2.2.7 Stopwatch Mode
2.2.6 Stopwatch Out
2.2.3 Time Setting
2.2.2 Time Keeping
2.2.1 Mode Controller
Trigger
Tick
Enable
Disable
Setting Alarm
Stop Time
Setting Time
Lab TIme
Mode
수정한 부분
2.2.17 Auto Increase TIme
2.2.5 Manual Increase Time
2.2.4 Switch Setting time
Setting Time
Current Time
Mode
수정한 부분

Structured Chart (advanced) – v2.0



Program Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <sys/types.h>
#include <pthread.h>

#include "alarmController.h"
#include "conio.h"
#include "modeController.h"
#include "backLightController.h"

/* GLOBAL */
MODE mode = TK_MODE;
BUTTON btn = NONE;
extern BOOL tk_isSetting;
extern BOOL al_isSetting;

/* main function */
int main(int argc, char *argv[]) {
    time_t init_time = 1546268400; /* 2019.01.01 00:00:00 */
    currentTime = localtime(&init_time);
    changeTime = (Time*) malloc( sizeof( Time ) );
    pthread_t alarm_ctr;

    tk_isSetting = FALSE;
    al_isSetting = FALSE;


    system("clear");
    printf("\e[?25l");
    pthread_create(&alarm_ctr, NULL, ringAlarm, NULL);
    set_conio_terminal_mode();
    while (1) {
        btn = pushedButtonDector();
//        if ( btn)
//            printf("%d\r\n", btn);
        alarmController();
        backLightController();
        modeController();

    }
}
```

main

Structure/Function	description
BUTTON btn	들어온 키 값을 저장하는 변수
Mode mode	timekeeping, alarm, stopwatch 중 어떤 mode인지 나타내는 변수

Program Code

```
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/select.h>
#include <termios.h>
/*
#include <windows.h>
*/// LINUX

typedef enum _BUTTON {
        NONE = 0, A, B, C, D
}BUTTON;

struct termios orig_termios;


void gotoxy(int, int);
void reset_terminal_mode();
void set_conio_terminal_mode();
int kbhit();
int getch();
int getKey();
BUTTON pushedButtonDector();
~
```

Conio.h

Structure/Function	description
Struct termios orig_terminos	kbhit()를 사용하기 위한 변수

Program Code

```c
#include "conio.h"

void gotoxy(int x, int y) {
        printf("\033[%d;%df",y,x);
        fflush(stdout);
}

void reset_terminal_mode() {
    tcsetattr(0, TCSANOW, &orig_termios);
}

void set_conio_terminal_mode() {
    struct termios new_termios;

    tcgetattr(0, &orig_termios);
    memcpy(&new_termios, &orig_termios, sizeof(new_termios));

    atexit(reset_terminal_mode);
    cfmakeraw(&new_termios);
    tcsetattr(0, TCSANOW, &new_termios);
}

int kbhit() {
    struct timeval tv = { 0L, 0L };
    fd_set fds;
    FD_ZERO(&fds);
    FD_SET(0, &fds);
    return select(1, &fds, NULL, NULL, &tv);
}

int getch() {
    int r;
    unsigned char c;
    if ((r = read(0, &c, sizeof(c))) < 0) {
        return r;
    } else {
        return c;
    }
}

/* getKeyCode */
int getkey() {
        int ch = 0;
        if ((ch = getch()) == 0xE0)
                ch += getch();
        return ch;
}

BUTTON pushedButtonDector() {
        int key = 0;
        BUTTON res = 0;
        if(kbhit()) {
                key = getkey();
                if ('A' <= key && key <= 'D')
                        res = key - 'A' + 1;
                else if ('a' <= key && key <= 'd')
                        res = key - 'a' + 1;
                else if (key == 27) /* ESC KEY INPUT */
                        exit(0);
        }
        return res;
}
```

1,1 All

Conio.c

Structure/Function	Description
gotoxy(int,int)	출력되는 좌표를 설정해주는 함수
reset_terminal_mode()	kbhit()를 사용하기 위한 함수
set_conio_terminal_mode()	kbhit()를 사용하기 위한 함수
kbhit()	키보드 입력이 있는지 확인하는 함수
getch()	buffer 없이 문자를 입력 받는 함수
getkey()	키보드 입력을 받는 함수
pushedButtonDetector()	입력된 키보드 입력이 무엇인지 판단하는 함수; 지정된 입력이 아니면 0 반환

Program Code

```c
#pragma once
#include <time.h>
#include <stdlib.h>
#include <pthread.h>

/* sturcture */

typedef struct tm Time;
Time* currentTime;
Time* changeTime;

typedef enum _MODE {
	TK_MODE, AL_MODE, SW_MODE
} MODE;

typedef enum alarmChange {
	AL_HOUR, AL_MIN
} AL_CH;

typedef enum watchChange {
	W_SEC, W_HOUR, W_MIN, W_YEAR, W_MONTH, W_DAY
} W_CH;

typedef enum boolean {
	FALSE, TRUE
} BOOL;

typedef struct time_sw {
	int min;
	int sec;
	int centi;	// centi-sec
} sw_Time;
```

modeController.h

Structure/Function	Description
Time* currentTime	autoIncreaseTime( ) 함수의 인자로 전달되어 시간을 증가시키고, 증가된 시간을 받는 변수

Program Code

```c
#include <stdio.h>
#include "conio.h"
#include "modeController.h"

extern MODE mode;

void modeController(  ) {

        currentTime = autoIncreaseTime(currentTime);
        if ( mode == TK_MODE )
                timekeeping_mode();
        else if ( mode == AL_MODE )
                alarm_mode();
        else
                stopwatch_mode();
}


Time* autoIncreaseTime(Time* currentTime) {

//      static clock_t p_clock = 0;
//      clock_t c_clock = clock();
        static struct timespec p_clock;
        struct timespec c_clock = { 0, 0 };

        static int start = 0;
        if ( start == 0 ) {
                clock_gettime( CLOCK_MONOTONIC, &p_clock );
        start = 1;
        }
        clock_gettime( CLOCK_MONOTONIC, &c_clock );

        time_t t = mktime(currentTime);

//      if (c_clock - p_clock >= CLOCKS_PER_SEC) {
        if ( ( c_clock.tv_sec * 1000000000 + c_clock.tv_nsec ) - ( p_clock.tv_sec * 1000000000 + p_clock.tv_nsec
 ) >= 1000000000 ) {
                /* In One Second */
                if (t == 4102412399)
                    t = 1546268399;
                t++; /* Increae one second */
//              p_clock = clock(); /* Assign current clock to p_clock */
                clock_gettime( CLOCK_MONOTONIC, &p_clock );

/*XXX*///gotoxy(0, 0);
/*XXX*///printf("%lu", t);
/*XXX*/
                /*printf("    \033[4m%04d\033[0m, %02d, %02d, %02d:%02d:%02d   %llu\r\n",
                                currentTime->tm_year + 1900,
                                currentTime->tm_mon + 1,
                                currentTime->tm_mday,
                                currentTime->tm_hour,
                                currentTime->tm_min,
                                currentTime->tm_sec,
                                t
                                );*/


        }

        return localtime(&t);
}
```

modeController.c

Structure/Function	Description
void modeController()	mode 값에 따라 timekeeping_mode(), alarm_mode(), stopwatch_mode() 중 하나를 호출하는 함수
Time* autoIncreaseTime(Time* currentTime)	1초가 지날 때마다 currentTime에 저장된 시간을 1초씩 늘려주는 함수

Program Code

```c
#include <stdio.h>
#include <time.h>

#pragma once

int isRingAlarm;


void ringoff();

void *ringAlarm(void* args);

int check_Alarm_Time();

int alarmController( );
~
~
```

alarmController.h

Structure/Function	Description
isRIngAlarm	알람이 울리는 지에 대한 정보를 저장하는 변수.

Program Code

```c
#include <unistd.h>
#include <pthread.h>
#include "conio.h"
#include "modeController.h"
#include "alarmController.h"

extern BUTTON btn;
extern Time al_time;                    // global for display
extern BOOL al_isSet;                 // global for display


void ringOff() {
    isRingAlarm = 0;
}

void *ringAlarm(void* args) {
    struct timespec now, from;
    /*XXX*///printf("ringAlarm() : is Created!\r\n");
    while (1) {
        clock_gettime( CLOCK_MONOTONIC, &from);
        if (isRingAlarm)
            system("echo -e '\a'");
        while (isRingAlarm) {
            clock_gettime( CLOCK_MONOTONIC, &now );
            if ( ( now.tv_sec * 1000000000 + now.tv_nsec )
                 - ( from.tv_sec * 1000000000 + from.tv_nsec ) >= 1000000000 ) {
                system("echo -e '\a'");
                /*XXX*///printf("ringAlarm() : ring\r\n");
                clock_gettime( CLOCK_MONOTONIC, &from );
            }
        }
    }
}

int check_Alarm_Time(){

    if (al_isSet == TRUE
        && (al_time.tm_min== currentTime->tm_min && al_time.tm_hour== currentTime->tm_hour && 0 == currentTime->
tm_sec ))
        return 1;
    else
        return 0;
}

int alarmController( ) {

    if (!isRingAlarm && check_Alarm_Time()) {
        /*XXX*///printf("alarmController() : change isRingAlarm%d\r\n", isRingAlarm);
        isRingAlarm = 1;
    }


    if( isRingAlarm && ((btn ==1 || btn == 2 || btn == 3 || btn == 4)||(currentTime->tm_sec >= 5))) {
        ringoff();
        isRingAlarm = 0;
        btn = NONE;
    }

}
~
```

alarmController.c

Structure/Function	Description
void ringOff( )	isRingAlarm을 0으로 설정해주는 함수
int alarmController( )	check_Alarm_Time()을 호출해 alarm 시간이 되었는지 검사하며 alarm을 켜고 alarm이 켜져 있을 때 btn이 들어오면 alarm을 끄는 함수
void* ringAlarm(Void* args)	alarm beep 음을 출력하는 thread 함수
int check_Alarm_Time( )	currentTime과 al_time이 일치하는지 검사하는 함수

Program Code

```h
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

struct tm BackLightTime;

void backLightController(int btn);

void backLight();

void lightoff();
```

```c
#include "modeController.h"
#include "alarmController.h"
#include "backLightController.h"


int back_lighting = 0;

void backLightController(int btn){

        if(isRingAlarm == 0 && btn ==4){
        back_lighting = 1;
                BackLightTime.tm_sec = currentTime->tm_sec;
                backLight();//FIXME: ▒~]대 | ~G寬~L ▒~U~H ▒~U|  ▒~H~X ▒~W~F▒~]~D源~L?
        }

        if(back_lighting == 1 && currentTime->tm_sec - BackLightTime.tm_sec == 2){
                lightoff();
        back_lighting = 0;
        }
}

void backLight(){
    back_lighting = 1;
}

void lightoff(){
    back_lighting = 0;
}
```

backLightController.c

Structure/Function	Description
int back_lighting	backlight를 켤지 끌지 설정하는 변수
void backLightController()	btn값에 따라 backLight() 또는 lightOff()를 호출하는 함수
void backLight()	back_lighting을 1로 설정하는 함수
void lightOff()	back_lighting을 0로 설정하는 함수

Program Code

```c
#include "conio.h"
#include "modeController.h"

extern MODE mode;
extern BUTTON btn;
extern Time* currentTime;
extern BOOL al_isSet; // Add for Display by harheem
W_CH tk_toChange;       // global for display
BOOL tk_isSetting;      // global for display
extern Time* changeTime;
char day[][3] = { "SU", "MO", "TU", "WE", "TH", "FR", "SA" };
extern int back_lighting;
extern int light;

void switch_setting_time( ) {
        // toChange: select what to change
        // Second -> Hour -> Minute -> Year -> Month -> Day -> Second

        tk_toChange =  ( tk_toChange + 1 ) % 6;
/*XXX*/ //printf( "switch_setting_time(): tochange=%d\r\n", tk_toChange );
}
```

mode_tk.c

Structure/Function	Description
W_CH tk_toChange	time-setting mode에서 어떤 값을 증가시킬 것인가를 나타내는 변수
BOOL tk_isSetting	time-keeping mode인지 time-setting mode인지 나타내는 변수
void switch_setting_time()	tk_toChange를 1 증가시키는 함수; 5보다 커지면 0으로 초기화

Program Code

```
void manual_incease_time( ) {
	// currentTime: Current Time tm STORAGE
	// toChange: what to change

	int year = 0;	// for OPT
	int day = 0;	// for day++

	switch ( tk_toChange ) {
		case W_SEC:
			changeTime->tm_sec = ( changeTime->tm_sec + 1 ) % 60;
			break;
		case W_HOUR:
			changeTime->tm_hour = ( changeTime->tm_hour + 1 ) % 24;
			break;
		case W_MIN:
			changeTime->tm_min = ( changeTime->tm_min + 1 ) % 60;
			break;
		case W_YEAR:
			changeTime->tm_year++;
			if ( changeTime->tm_year > 199 )	// year > 2099
				changeTime->tm_year = 119;	// year := 2019
			break;
		case W_MONTH:
			changeTime->tm_mon = ( changeTime->tm_mon + 1 ) % 12;
			break;
		case W_DAY:
			day = changeTime->tm_mday + 1;

			switch( changeTime->tm_mon ) {
				case 0:
				case 2:
				case 4:
				case 6:
				case 7:
				case 9:
				case 11:
					if ( day > 31 )
						changeTime->tm_mday = 1;
					else
						changeTime->tm_mday = day;
					break;
				case 1:
					year = changeTime->tm_year;
					if ( ( !( year % 4 ) && ( year % 100 ) ) || !( year % 400 ) ) {
						if ( day > 29 )
							changeTime->tm_mday = 1;
						else
							changeTime->tm_mday = day;
					}
					else {
						if ( day > 28 )
							changeTime->tm_mday = 1;
						else
							changeTime->tm_mday = day;
					}
					break;
				default:
					if ( day > 30 )
						changeTime->tm_mday = 1;
					else
						changeTime->tm_mday = day;
			}
	}
/*XXX*/ //printf( "manual_increase_time(): Time=%d/%d/%d %d:%d:%d\r\n", changeTime->tm_year+1900, changeTime->tm_mon+1, changeTime->tm_mday, changeTime->tm_hour, changeTime->tm_min, changeTime->tm_sec );
}
```

mode_tk.c

Structure/Function	Description
void manual_increase_time()	currentTime에서 tk_toChange가 나타내는 값을 1 증가시키는 함수; 각각의 최대치를 넘어서면 0으로 초기화

Program Code

mode_tk.c

Structure/Function	Description
void display_tk(int month, int date, int hour, int minute, int second)	currentTime의 시간을 인자로 받아 화면에 출력하는 함수

```
void display_tk(int type) {
    Time* time = changeTime;
    if (type)
        time = currentTime;

        gotoxy(0, 0);
        printf("   -----------\r\n");
        if (tk_isSetting) {
                if (tk_toChange == W_MONTH) {
                        printf("    %c[%d;24m%s %c[%d;4m%02d%c[%d;24m-%02d%c[0;24m \r\n", 27, light, day[currentTime->tm_wday], 27, light, time->tm_mon + 1, 27, light, time->tm_mday, 27);
                }
                else if(tk_toChange == W_DAY) printf("    %c[%d;24m%s %02d-%c[%d;4m%02d%c[0;24m \r\n", 27, light, day[currentTime->tm_wday], time->tm_mon + 1, 27, light, time->tm_mday, 27);
                else printf("    %c[%dm%s %02d-%02d%c[0m \r\n", 27, light, day[currentTime->tm_wday], time->tm_mon + 1, time->tm_mday, 27);
        }
        else printf("    %c[%dm%s %02d-%02d%c[0m \r\n", 27, light, day[currentTime->tm_wday], time->tm_mon + 1, time->tm_mday, 27);
        printf("  ------------\r\n\r\n");
        if (al_isSet) {
                if (tk_isSetting) {
                        if (tk_toChange == W_SEC) {
                                printf("  *  %c[%d;24m%02d:%02d %c[%d;4m%02d%c[0;24m \r\n\r\n", 27, light, time->tm_hour, time->tm_min, 27, light, time->tm_sec, 27);
                        }
                        else if (tk_toChange == W_HOUR) {
                                printf("  *  %c[%d;4m%02d%c[%d;24m:%02d %02d%c[0;24m \r\n\r\n", 27, light, time->tm_hour, 27, light, time->tm_min, time->tm_sec, 27);
                        }
                        else if (tk_toChange == W_MIN) {
                                printf("  *  %c[%d;24m%02d:%c[%d;4m%02d%c[%d;24m %02d%c[0;24m \r\n\r\n", 27, light, time->tm_hour, 27, light, time->tm_min, 27, light, time->tm_sec, 27);
                        }
                        else printf("  *  %c[%dm%02d:%02d %02d%c[0m \r\n\r\n", 27, light, time->tm_hour, time->tm_min, time->tm_sec, 27);
                }
                else printf("  *  %c[%dm%02d:%02d %02d%c[0m \r\n\r\n", 27, light, time->tm_hour, time->tm_min, time->tm_sec, 27);
        }
```

Program Code

```
        else {
                if (tk_isSetting) {
                        if (tk_toChange == W_SEC) {
                                printf("     %c[%d;24m%02d:%02d %c[%d;4m%02d%c[0;24m \r\n\r\n", 27, light, time->tm_hour, time->tm_min, 27, light, time->tm_sec, 27);
                        }
                        else if (tk_toChange == W_HOUR) {
                                printf("     %c[%d;4m%02d%c[%d;24m:%02d %02d%c[0;24m \r\n\r\n", 27, light, time->tm_hour, 27, light, time->tm_min, time->tm_sec, 27);
                        }
                        else if (tk_toChange == W_MIN) {
                                printf("     %c[%d;24m%02d:%c[%d;4m%02d%c[%d;24m %02d%c[0;24m \r\n\r\n", 27, light, time->tm_hour, 27, light, time->tm_min, 27, light, time->tm_sec, 27);
                        }
                        else printf("     %c[%dm%02d:%02d %02d%c[0m \r\n\r\n", 27, light, time->tm_hour, time->tm_min, time->tm_sec, 27);
                }
                else printf("     %c[%dm%02d:%02d %02d%c[0m \r\n\r\n", 27, light, time->tm_hour, time->tm_min, time->tm_sec, 27);
        }
        printf("   ----------\r\n");
}
```

Program Code

```
void timekeeping_mode( ) {
	// currentTime: Current Time Storage

	if ( mode != TK_MODE ) {
/*XXX*/ //printf( "timekeeping_mode(): Not Timekeeping Mode - RETURN\r\n" );
		return;
	}
	if (back_lighting == 1) light = 33;
	else light = 0;
	if (tk_isSetting == TRUE) display_tk(changeTime->tm_mon + 1, changeTime->tm_mday, changeTime->tm_hour, changeTime->tm_min, changeTime->tm_sec);
	else display_tk(currentTime->tm_mon + 1, currentTime->tm_mday, currentTime->tm_hour, currentTime->tm_min, currentTime->tm_sec);


	if ( tk_isSetting == FALSE ) {
		// if BUTTON-C pressed in TIME KEEPING MODE,	goto ALARM MODE
		if ( btn == C ) {
			mode = ( mode + 1 ) % 3;
/*XXX*/ //printf( "timekeeping_mode(): Mode Change - AL; RETURN\r\n" );
			btn = NONE;
			return;
		}

		// if BUTTON-B pressed in TIME KEEPING MODE, DO NOTHING

		// if BUTTON-A pressed in TIME KEEPING MODE,	become TIME SETTING MODE
		if ( btn == A ) {
			time_switch( changeTime, currentTime );
			tk_isSetting = ( tk_isSetting + 1 ) % 2;
			tk_toChange = W_SEC;
/*XXX*/ //printf( "timekeeping_mode(): TIME SETTING MODE BEGIN; toChange=%d\r\n", tk_toChange );
		}
		// XXX DISPLAY - PROCESS 2.2.3: struct tm Setting ( toChange )
	}

	/***********************
	 * [~V] Time Keeping Mode *
	 * [~V] Time Setting Mode *
	 ***********************/

	else {
		// if BUTTON-C pressed in TIME SETTING MODE,	XXX PROCESS 2.2.4: Switch Setting Time XXX
		if ( btn == C ) {
			switch_setting_time();
		}

		// if BUTTON-B pressed in TIME SETTING MODE,	XXX PROCESS 2.2.5: Manual Increase Time XXX
		if ( btn == B ) {
			manual_increase_time();
			time_switch(currentTime, changeTime);
		}

		// if BUTTON-A pressed in TIME SETTING MODE,	return to TIME KEEPING MODE
		if ( btn == A ) {
			time_switch( currentTime, changeTime );
			tk_isSetting = ( tk_isSetting + 1 ) % 2;
/*XXX*/ //printf( "timekeeping_mode(): TIME SETTING MODE END; toChange=%d\r\n", tk_toChange );
			// XXX DISPLAY - PROCESS 2.2.2: Time Keeping ( currentTime )
		}
	}

	btn = NONE;
}


void time_switch( Time* dst, Time* src ) {
	memcpy( dst, src, sizeof( Time ) );
}
```

mode_tk.c

Structure/Function	Description
void timekeeping_mode()	btn값에 따라 timekeeping mode에서의 process를 수행하는 함수
void time_switch(Time* dst, Time* src)	시간 변수 dst에 시간 변수 src의 값을 넣어주는 함수

Program Code

```c
#include "conio.h"
#include "modeController.h"

/* GLOBAL */

pthread_t sw_thread;            // global for thread
sw_Time sw_time;                // global for thread
BOOL sw_isWork = FALSE;         // global for thread
pthread_mutex_t sw_mtx = PTHREAD_MUTEX_INITIALIZER;
sw_Time sw_lap;                 // global for display
BOOL sw_isLap = FALSE;          // global for display
extern MODE mode;
extern BUTTON btn;
extern BOOL al_isSet; // Add for Display by harheem
extern Time* changeTime;
extern int back_lighting;
char light = 0;

/* FUNCTION */

void *sw_increase( void* arg ) {
        // thread handler
        stopwatch_reset();

        struct timespec from;
        struct timespec now;

        clock_gettime( CLOCK_MONOTONIC, &from );

/*XXX*/ //printf( "sw_increase(): INCREASING START\r\n" );
        while ( 1 ) {
                if ( sw_isWork == TRUE ) {
                        // XXX PROCESS 2.2.9: Stopwatch Measurement XXX
                        // TODO FIXME increase
                        clock_gettime( CLOCK_MONOTONIC, &now );
                        if ( ( now.tv_sec * 1000000000 + now.tv_nsec ) - ( from.tv_sec * 1000000000 + from.tv_nsec ) >= 10000000 ) {
                                pthread_mutex_lock( &sw_mtx );
                                sw_time.centi++;
/*XXX*/ //printf( "sw_increase(): %d:%d:%d\r\n", sw_time.min, sw_time.sec, sw_time.centi );

                                if ( sw_time.centi >= 100 ) {
                                        sw_time.centi -= 100;
                                        sw_time.sec++;

                                        if ( sw_time.sec >= 60 ) {
                                                sw_time.sec -= 60;
                                                sw_time.min++;
                                        }
                                }
                                pthread_mutex_unlock( &sw_mtx );
                                clock_gettime( CLOCK_MONOTONIC, &from );
                        }
                }
                if ( mode != SW_MODE )
                        break;
        }
/*XXX*/ //printf( "sw_increase(): INCREASING FINISH\r\n" );
}
```

mode_sw.c

Structure/Function	Description
sw_Time sw_time	stopwatch의 시간을 저장하는 변수
BOOL sw_isWork	stopwatch 작동 여부를 저장하는 변수
pthread_mutex_t sw_mtx	stopwatch thread의 thread id
sw_Time sw_lap	laptime의 시간을 저장하는 변수
char light	backlight가 켜져 있는지 저장하는 변수
void *sw_increase(void* arg)	sw_isWork가 TRUE일 때 sw_time의 시간을 0.01초씩 증가시키는 thread 함수

Program Code

```
void record_laptime( ) {
        // sw_time: increasing stopwatch time
        // sw_lap: laptime for display
        // sw_isLap: is laptime on display?

        sw_isLap = TRUE;
        pthread_mutex_lock( &sw_mtx );
        memcpy( &sw_lap, &sw_time, sizeof( sw_Time ) );
        pthread_mutex_unlock( &sw_mtx );
/*XXX*/// printf( "record_laptime(): Lap=%d:%d:%d\r\n", sw_lap.min, sw_lap.sec, sw_lap.centi );
}

void stopwatch_reset( ) {
        // sw_time: increasing stopwatch time TO RESET

        pthread_mutex_lock( &sw_mtx );
        sw_time.min = 0;
        sw_time.sec = 0;
        sw_time.centi = 0;
                //Add LabTime Initialization by harheem
                sw_lap.centi = 0;
                sw_lap.min = 0;
                sw_lap.sec = 0;
        pthread_mutex_unlock( &sw_mtx );
/*XXX*/ //printf( "stopwatch_reset()\r\n" );
}


void display_sw(int hour, int minute, int sw_minute, int sw_second, int sw_microsecond) {
        gotoxy(0, 0);
        printf("   -----------\r\n");
        printf("    %c[%dmST %02d:%02d%c[0;0m \r\n", 27, light, hour, minute, 27);
        printf("  -------------\r\n\r\n");
        if (al_isSet) printf("  *  %c[%dm%02d'%02d\"%02d%c[0;0m \r\n\r\n", 27, light, sw_minute, sw_second, sw_microsecond, 27);
        else printf("     %c[%dm%02d'%02d\"%02d%c[0;0m \r\n\r\n", 27, light, sw_minute, sw_second, sw_microsecond, 27);
        printf("   -----------\r\n");
}
```

mode_sw.c

Structure/Function	Description
void record_laptime()	sw_time의 시간을 sw_lap에 복사하는 함수
void stopwatch_reset()	sw_time의 시간을 초기화하는 함수
void display_sw(int hour, int minute, int sw_minute, int sw_second, int sw_microsecond)	sw_time 의 시간을 인자로 받아 화면에 출력하는 함수

Program Code

```
void stopwatch_mode( ) {
        // sw_time: increasing stopwatch time
        // sw_lap: laptime for display
        // sw_isLap: is laptime on display? -> global for display
        // sw_isWork: is stopwatch work?
        // sw_thread: thread for increment

        if ( mode != SW_MODE ) {
/*XXX*/ //printf( "stopwatch_mode(): Not Stopwatch Mode - RETURN\r\n" );
        return;
        }

        //if LabTime is Not 00'00"00 -> Display LabTime by harheem
        if (back_lighting == 1) light = 33;
        else light = 0;
        if (sw_isLap == TRUE) display_sw(currentTime->tm_hour, currentTime->tm_min, sw_lap.min, sw_lap.sec, sw_lap.centi);
        else display_sw(currentTime->tm_hour, currentTime->tm_min, sw_time.min, sw_time.sec, sw_time.centi);

        // if BUTTON-C pressed && !sw_isWork,   goto TIME KEEPING MODE
        if ( btn == C && sw_isWork == FALSE ) {
/*XXX*/ //printf( "stopwatch_mode(): Mode Change - TK; RETURN\r\n" );
                mode = ( mode + 1 ) % 3;
                return;
        }

        // if BUTTON-C pressed && sw_isWork,    DO NOTHING

        if ( btn == B ) {
                if ( sw_isLap == FALSE ) {
                        // if BUTTON-B pressed && !sw_isLab     sw_isWork: T -> F -> T
                        sw_isWork = ( sw_isWork + 1 ) % 2;
/*XXX*/ //printf( "stopwatch_mode(): isWork=%d\r\n", sw_isWork );
                }
                else {
                        // if BUTTON-B pressed && sw_isLab      show sw_time
                        sw_isLap = FALSE;
/*XXX*/ //printf( "stopwatch_mode(): isWork=%d; isLap=%d\r\n", sw_isWork, sw_isLap );
                }
        }

        if ( btn == A ) {
                if ( sw_isWork == TRUE ) {
                        // if BUTTON-A pressed && sw_isWork,    XXX PROCESS 2.2.11: Record Laptime XXX
                        record_laptime( );
                }
                else {
                        // if BUTTON-A pressed in STOPWATCH MODE when ( sw_work == FALSE ),     XXX PROCESS 2.2.8: Stopwatch Reset XXX
                        stopwatch_reset( );
                }
        }

        btn = NONE;
}
```

mode_sw.c

Structure/Function	Description
void stopwatch_mode()	btn값에 따라 stopwatch mode에서의 process를 수행하는 함수

Program Code

```c
#include "conio.h"
#include "modeController.h"

/* GLOBAL */

extern MODE mode;
extern BUTTON btn;
extern pthread_t sw_thread;
Time al_time;                   // global for display
BOOL al_isSet;          // global for display
AL_CH al_toChange;              // global for display
BOOL al_isSetting;              // global for display
extern int back_lighting;
extern int light;

/* FUNCTION */

void alarm_onoff() {
        // al_isSet: is alarm setted?

        al_isSet = (al_isSet + 1 ) % 2;
/*XXX*/ //printf( "alarm_onoff(): al_isSet=%d\r\n", al_isSet );
}

void switch_setting_alarm_time() {
        // toChange: select what to change
        // Hour -> Minute -> Hour
        // return: selected


        al_toChange = ( al_toChange + 1 ) % 2;
/*XXX*/ //printf( "switch_setting_alarm_time(): tochange=%d\r\n", al_toChange );
}

void increase_alarm_time() {
        // alarm: Alarm Time STORAGE
        // toChange: what to change

        switch( al_toChange ) {
                case AL_HOUR:
                        al_time.tm_hour = ( al_time.tm_hour + 1 ) % 24;
                        break;
                case AL_MIN:
                        al_time.tm_min = ( al_time.tm_min + 1 ) % 60;
                        break;
        }
/*XXX*/ //printf( "increase_alarm_time(): Alarm=%d:%d\r\n", al_time.tm_hour, al_time.tm_min );
}
```

mode_al.c

Structure/Function	Description
Time al_time	alarm이 설정된 시간을 저장하는 변수
BOOL al_isSet	alarm 설정 여부를 저장하는 변수
AL_CH al_toChange	alarm-setting mode에서 어떤 값을 증가시킬 것인가를 나타내는 변수
BOOL al_isSetting	alarm mode인지 alarm-setting mode인지 나타내는 변수
void alarm_onoff()	btn 입력에 따라 alarm을 켜고 끄는 함수
void switch_setting_alarm_time()	tk_toChange를 1 증가시키는 함수; 1보다 커지면 0으로 초기화
void increase_alarm_time()	al_time의 시간을 인자로 받아 화면에 출력하는 함수

Program Code

mode_al.c

Structure/Function	Description
void display_al(int month, int date, int hour, int minute)	al_time의 시간을 인자로 받아 화면에 출력하는 함수
void alarm_mode()	btn값에 따라 alarm mode에서의 process를 수행하는 함수

```
void display_al() {
        gotoxy(0, 0);
        printf("   -----------\r\n");
        printf("    %c[%dmAL %02d-%02d%c[0m \r\n", 27, light, currentTime->tm_mon + 1, currentTime->tm_mday, 27);
        printf("  -------------\r\n\r\n");
        if (al_isSet) {
                if (al_isSetting) {
                        if (al_toChange == AL_HOUR) {
                                printf("  *    %c[%d;4m%02d%c[%d;24m:%02d%c[0;0m     \r\n\r\n", 27, light, al_time.tm_hour, 27, light, al_time.tm_min, 27);
                        }
                        else printf("  *    %c[%d;24m%02d:%c[%d;4m%02d%c[0;0m     \r\n\r\n", 27, light, al_time.tm_hour, 27, light, al_time.tm_min, 27);
                }
                else printf("  *    %c[%dm%02d:%02d%c[0m     \r\n\r\n", 27, light, al_time.tm_hour, al_time.tm_min, 27);
}
        else {
                if (al_isSetting) {
                        if (al_toChange == AL_HOUR) {
                                printf("       %c[%d;4m%02d%c[%d;24m:%02d%c[0;0m     \r\n\r\n", 27, light, al_time.tm_hour, 27, light, al_time.tm_min, 27);
                        }
                        else printf("       %c[%d;24m%02d:%c[%d;4m%02d%c[0;0m     \r\n\r\n", 27, light, al_time.tm_hour, 27, light, al_time.tm_min, 27);
                }
                else printf("       %c[%dm%02d:%02d%c[0m     \r\n\r\n", 27, light, al_time.tm_hour, al_time.tm_min, 27);
        }
        printf("   -----------\r\n");
}
```

Program Code

```
void alarm_mode() {
        // alarm: Alarm Time STORAGE
        // al_isset: is alarm setted?

        if ( mode != AL_MODE ) {
/*XXX*/ //printf( "alarm_mode(): Not Alarm Mode - RETURN\r\n" );
                return;
        }

        if (back_lighting == 1) light = 33;
        else light = 0;
        display_al(currentTime->tm_mon + 1, currentTime->tm_mday, al_time.tm_hour, al_time.tm_min);

        if ( al_isSetting == FALSE ) {
                // if BUTTON-C pressed in ALARM MODE,   goto STOPWATCH MODE
                if ( btn == C ) {
                        mode = ( mode + 1 ) % 3;
                        pthread_create( &sw_thread, NULL, sw_increase, NULL );
/*XXX*/ //printf( "alarm_mode(): Mode Change - SW; RETURN\r\n" );
                        btn = NONE;
                        return;
                }

                // if BUTTON-B pressed in ALARM MODE,   XXX PROCESS 2.2.16: Alarm On/Off XXX
                if ( btn == B ) {
                        alarm_onoff();
                }

                // if BUTTON-A pressed in ALARM MODE,   become ALARM SETTING MODE
                if ( btn == A ) {
                        al_isSetting = ( al_isSetting + 1 ) % 2;
                        al_toChange = AL_HOUR;
/*XXX*/ //printf( "alarm_mode(): ALARM SETTING MODE BEGIN; toChange=%d\r\n", al_toChange );
                }
                // XXX DISPLAY - PROCESS 2.2.12: Alarm Setting ( toChange )
        }

        /************************
         * [~v] Alarm Mode         *
         * [~v] Alarm Setting Mode *
         ************************/

        else {
                // if BUTTON-C pressed in ALARM SETTING MODE,   XXX PROCESS 2.2.14: Switch Setting Alarm Time XXX
                if ( btn == C ) {
                        switch_setting_alarm_time();
                }

                // if BUTTON-B pressed in ALARM SETTING MODE,   XXX PROCESS 2.2.15: Increase Alarm Time XXX
                if ( btn == B ) {
                        increase_alarm_time();
                }

                // if BUTTON-A pressed in ALARM SETTING MODE,   become ALARM MODE
                if ( btn == A ) {
                        al_isSetting = ( al_isSetting + 1 ) % 2;
                        // XXX DISPLAY PROCESS 2.2.13: Alarm
/*XXX*/ //printf( "alarm_mode(): ALARM MODE BEGIN\r\n" );
                }
        }

        btn = NONE;
}
```

mode_al.c

Structure/Function	Description
void alarm_mode()	btn값에 따라 alarm mode에서의 process를 수행하는 함수

Thank you