Microwave Oven

T6 Extreme Cold Emerald

200911365 김건현
200911369 김동현
201011360 장윤정

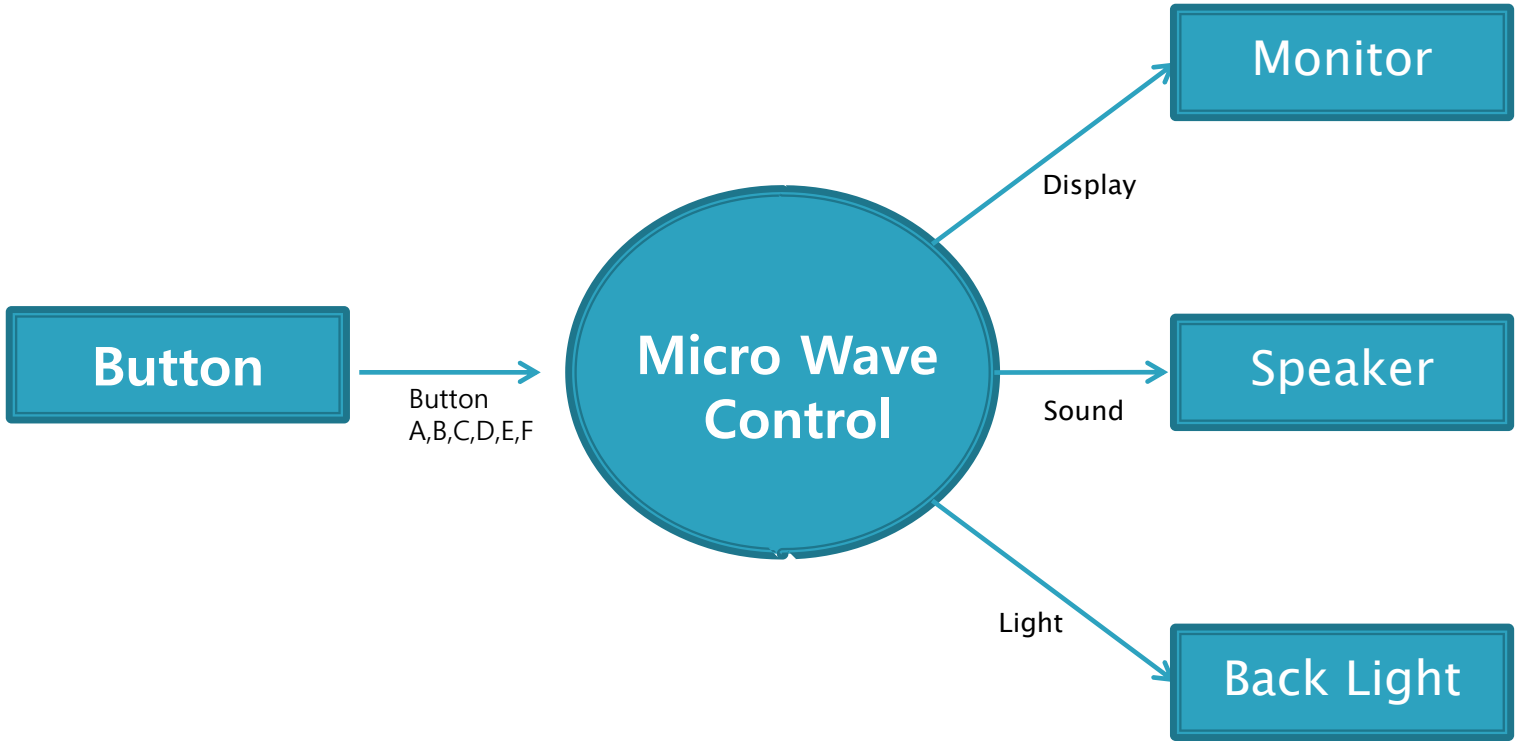
목차

- Event List
- System Contact Diagram
- Data Flow Diagram(DFD)
- Data Dictionary
- State Machine for Controller
- Process Specification
- Structured Charts

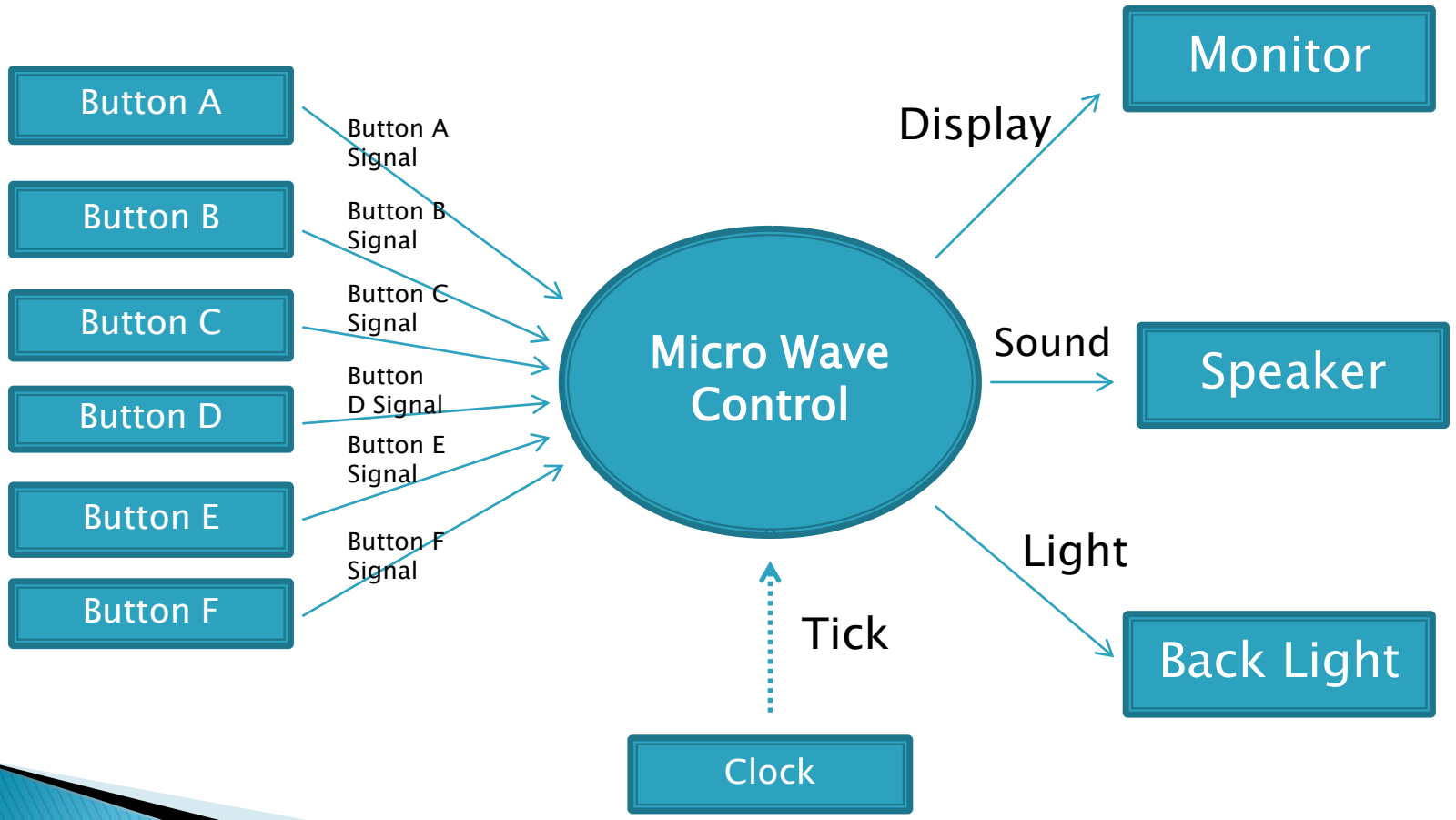
Event List

Symbol	Description	Type
A	Input : keyboard ‘a’ output : 10초 / 10도 증가	Interrupt
B	Input : keyboard ‘b’ output : 30초 / 20도 증가	Interrupt
C	Input : keyboard ‘c’ output : 시간 / 온도 모드 변경	Interrupt
D	Input : keyboard ‘d’ output : 조리 모드 선택	Interrupt
E	Input : keyboard ‘e’ output : 시작 / 취소 버튼	Interrupt
F	Input : keyboard ‘f’ output : 문 열림 감지 BackLight On/Off	Interrupt
G	Input : 현재 온도 output : 현재 온도를 1에 표시	Periodic
1	현재온도 & 설정온도 / 남은 시간 & 설정 시간 표시	Periodic
2	현재모드를 표시 (00 : 모드 사용 안 함)	Interrupt
3	Back Light 문 열림 감지 센서에 따라 작동	Interrupt
4	Beep speaker 3초 Beep 종료 알리는 소리	Periodic
Tick	프로그램 동작 중 계속해서 전달되는 전기적 신호 (1초에 한번씩 전달)	Periodic

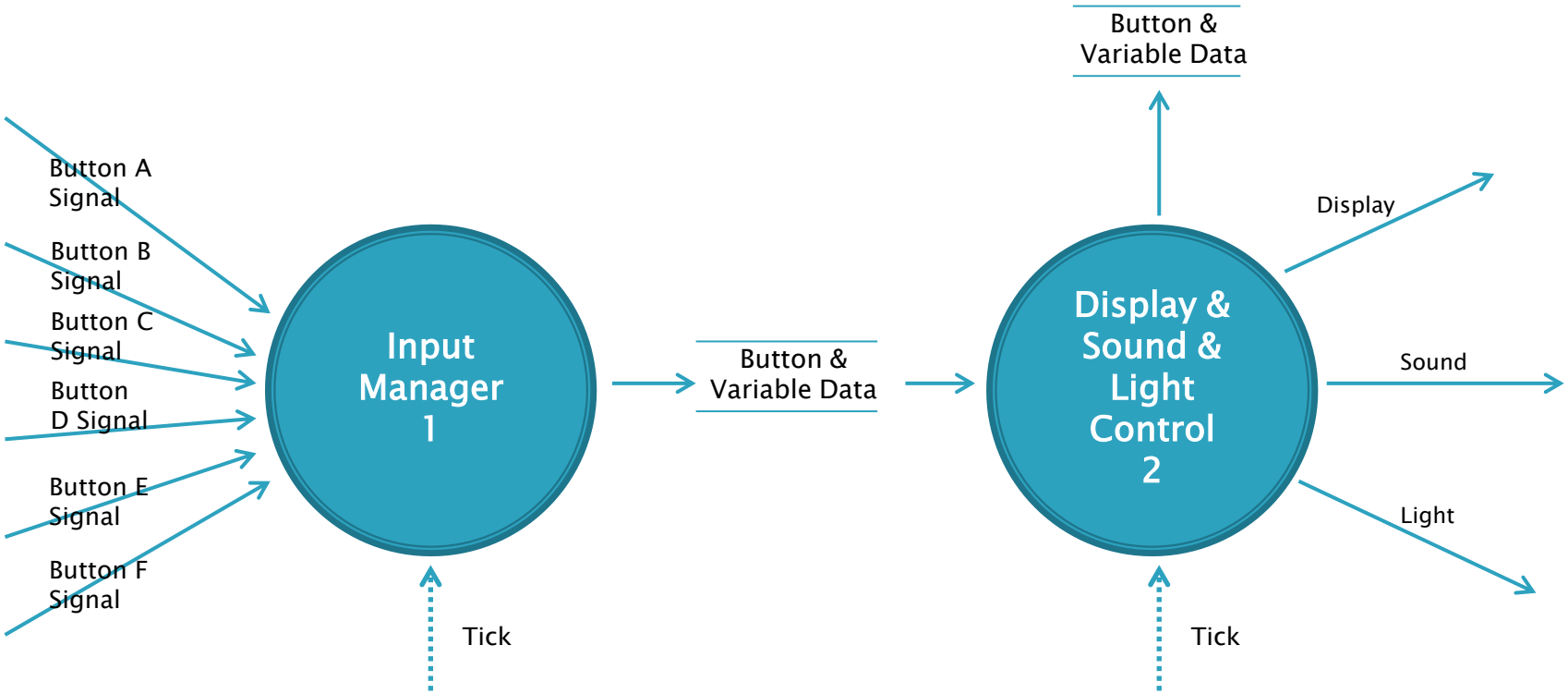
System Context Diagram



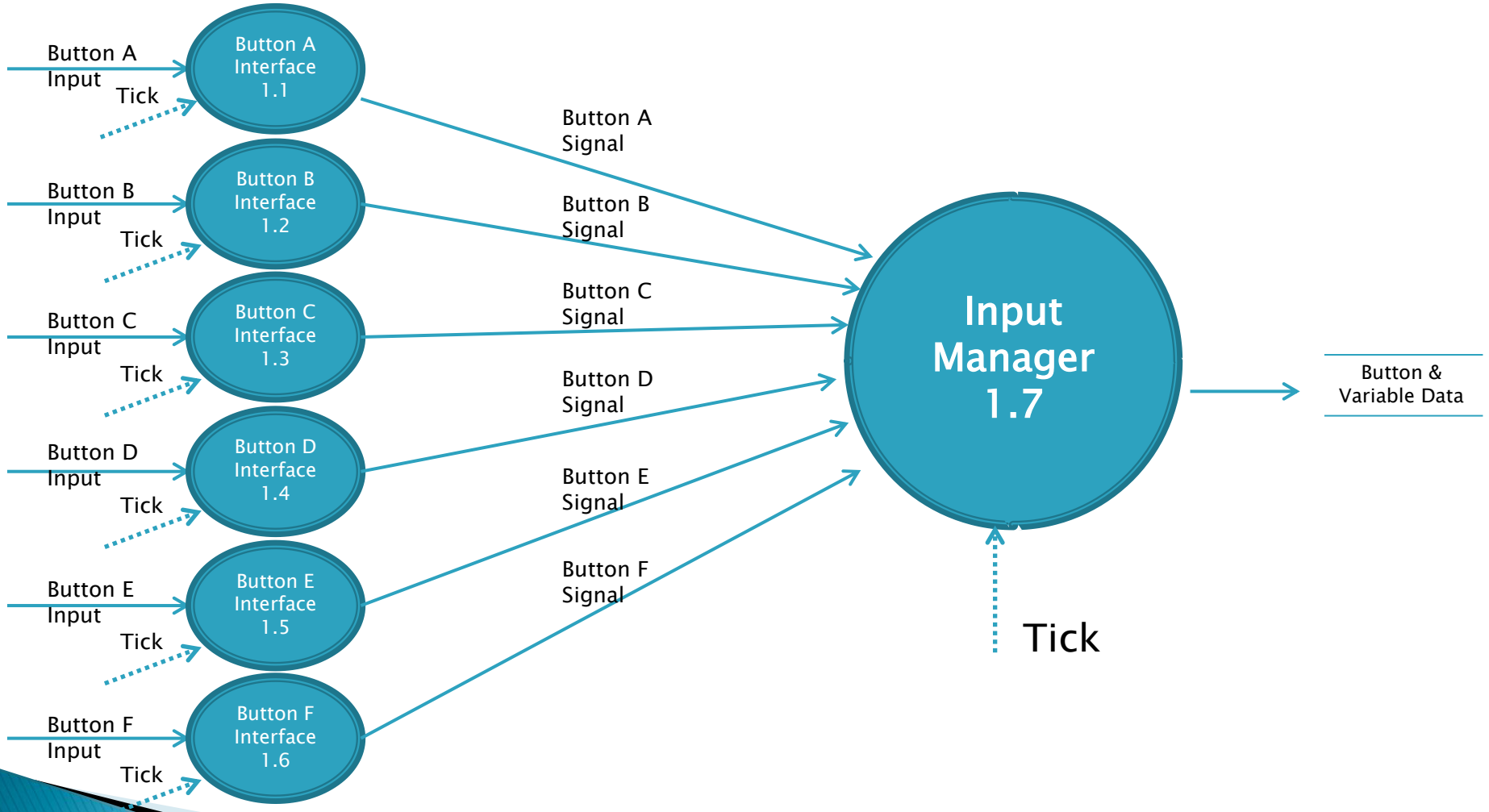
DFD (Level 0)


Button A
Button B
Button C
Button D
Button E
Button F
Button A Signal
Button B Signal
Button C Signal
Button D Signal
Button E Signal
Button F Signal
Micro Wave Control
Display
Monitor
Sound
Speaker
Light
Back Light
Tick
Clock

DFD (Level 1)


Button A Signal
Button B Signal
Button C Signal
Button D Signal
Button E Signal
Button F Signal
Input Manager 1
Tick
Button & Variable Data
Button & Variable Data
Display & Sound & Light Control 2
Tick
Display
Sound
Light

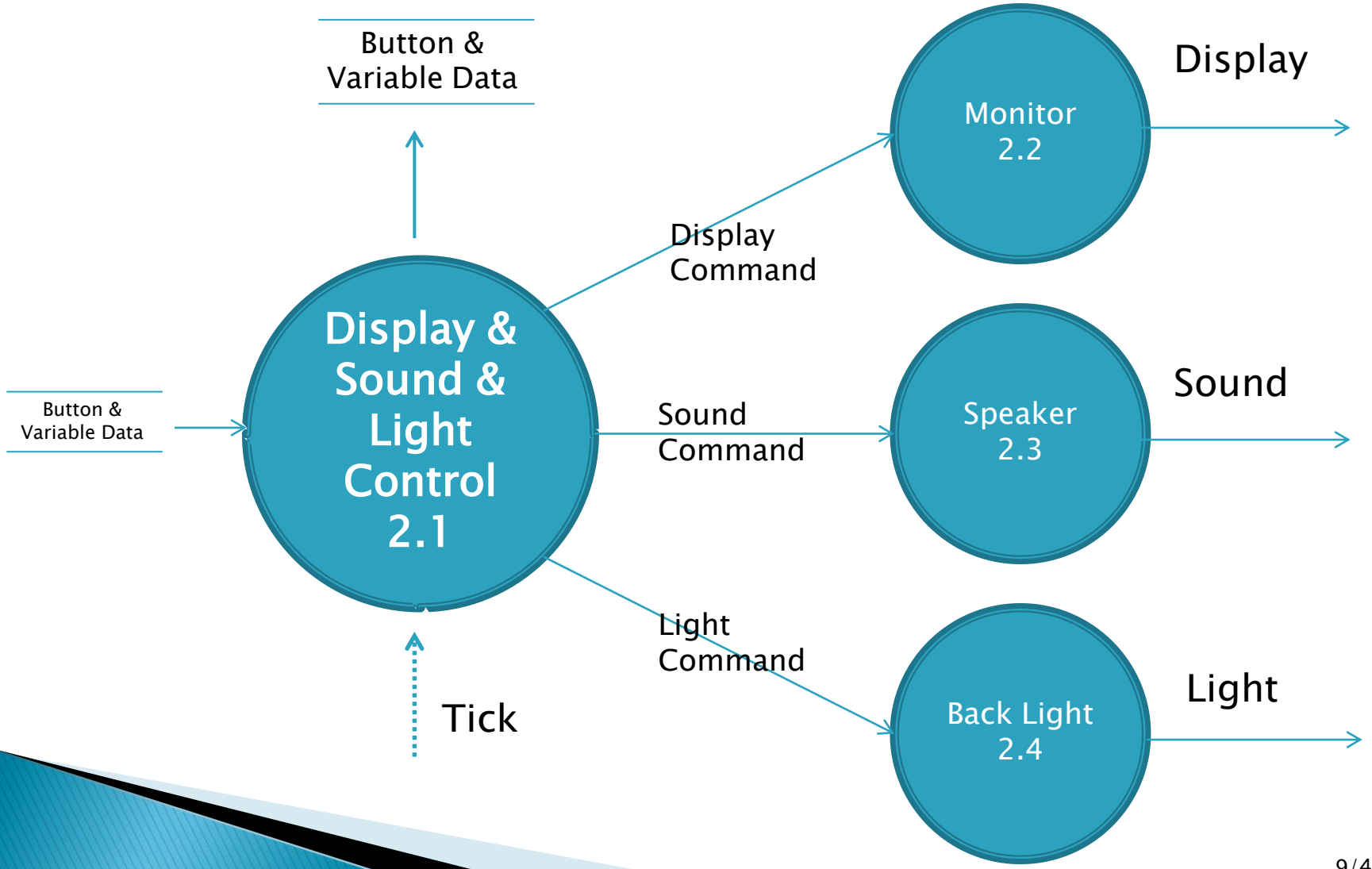
DFD (Level 2)


Button A Input
Tick
Button A Interface 1.1
Button B Input
Tick
Button B Interface 1.2
Button C Input
Tick
Button C Interface 1.3
Button D Input
Tick
Button D Interface 1.4
Button E Input
Tick
Button E Interface 1.5
Button F Input
Tick
Button F Interface 1.6
Button A Signal
Button B Signal
Button C Signal
Button D Signal
Button E Signal
Button F Signal
Input Manager 1.7
Tick
Button & Variable Data

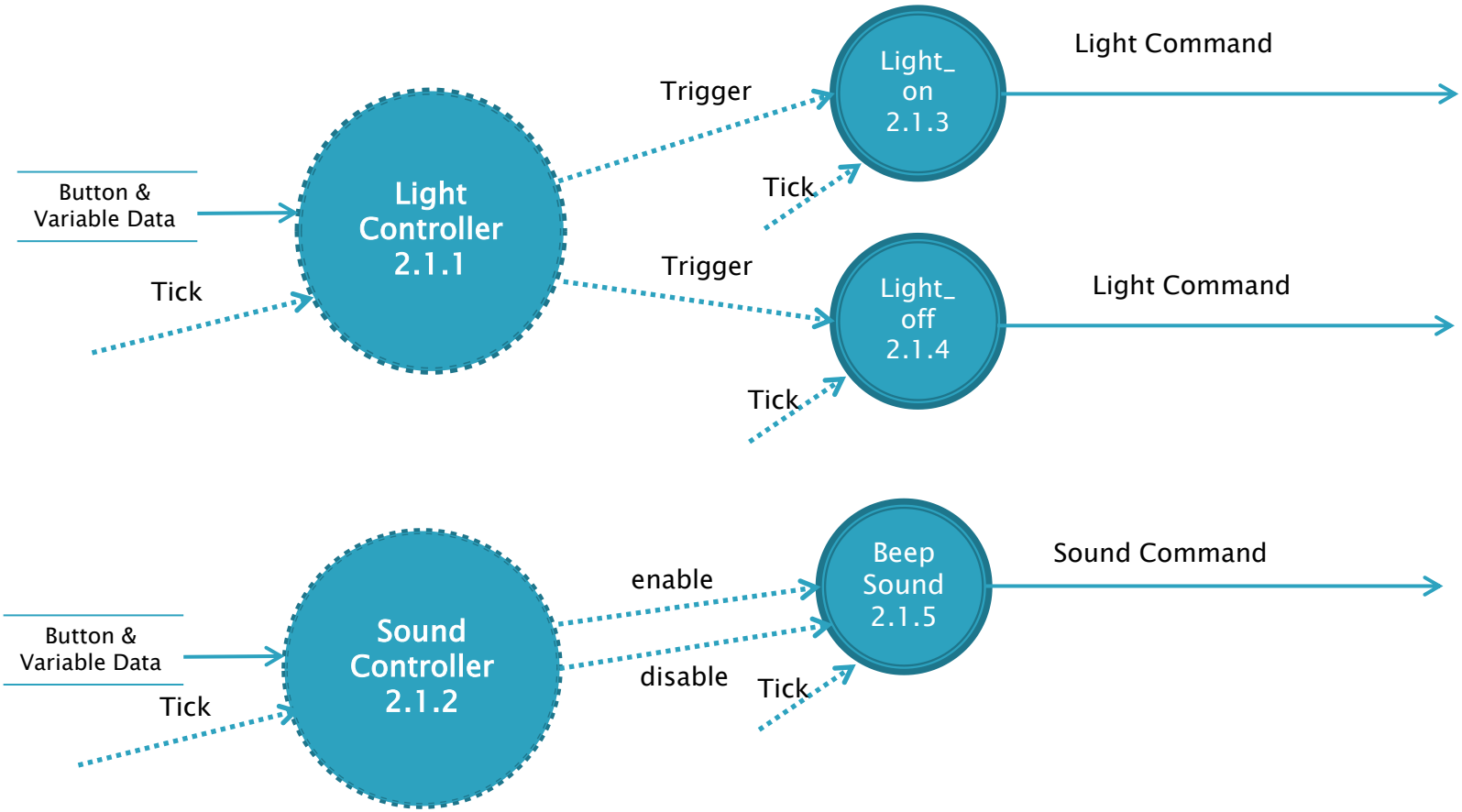
Data Dictionary

Input/Output Event	Description	Type	Format
Button A Input	Keyboard로부터 입력 값이 'a'일 경우 boolean 형 변수를 true로 변경하는 Input을 Button A Interface 로 전달	Interrupt	Boolean
Button B Input	Keyboard로부터 입력 값이 'b'일 경우 boolean 형 변수를 true로 변경하는 Input을 Button B Interface 로 전달	Interrupt	Boolean
Button C Input	Keyboard로부터 입력 값이 'c'일 경우 boolean 형 변수를 true로 변경하는 Input을 Button C Interface 로 전달	Interrupt	Boolean
Button D Input	Keyboard로부터 입력 값이 'd'일 경우 boolean 형 변수를 true로 변경하는 Input을 Button D Interface 로 전달	Interrupt	Boolean
Button E Input	Keyboard로부터 입력 값이 'e'일 경우 boolean 형 변수를 true로 변경하는 Input을 Button E Interface 로 전달	Interrupt	Boolean
Button F Input	Keyboard로부터 입력 값이 'f'일 경우 boolean 형 변수를 true로 변경하는 Input을 Button F Interface 로 전달	Interrupt	Boolean

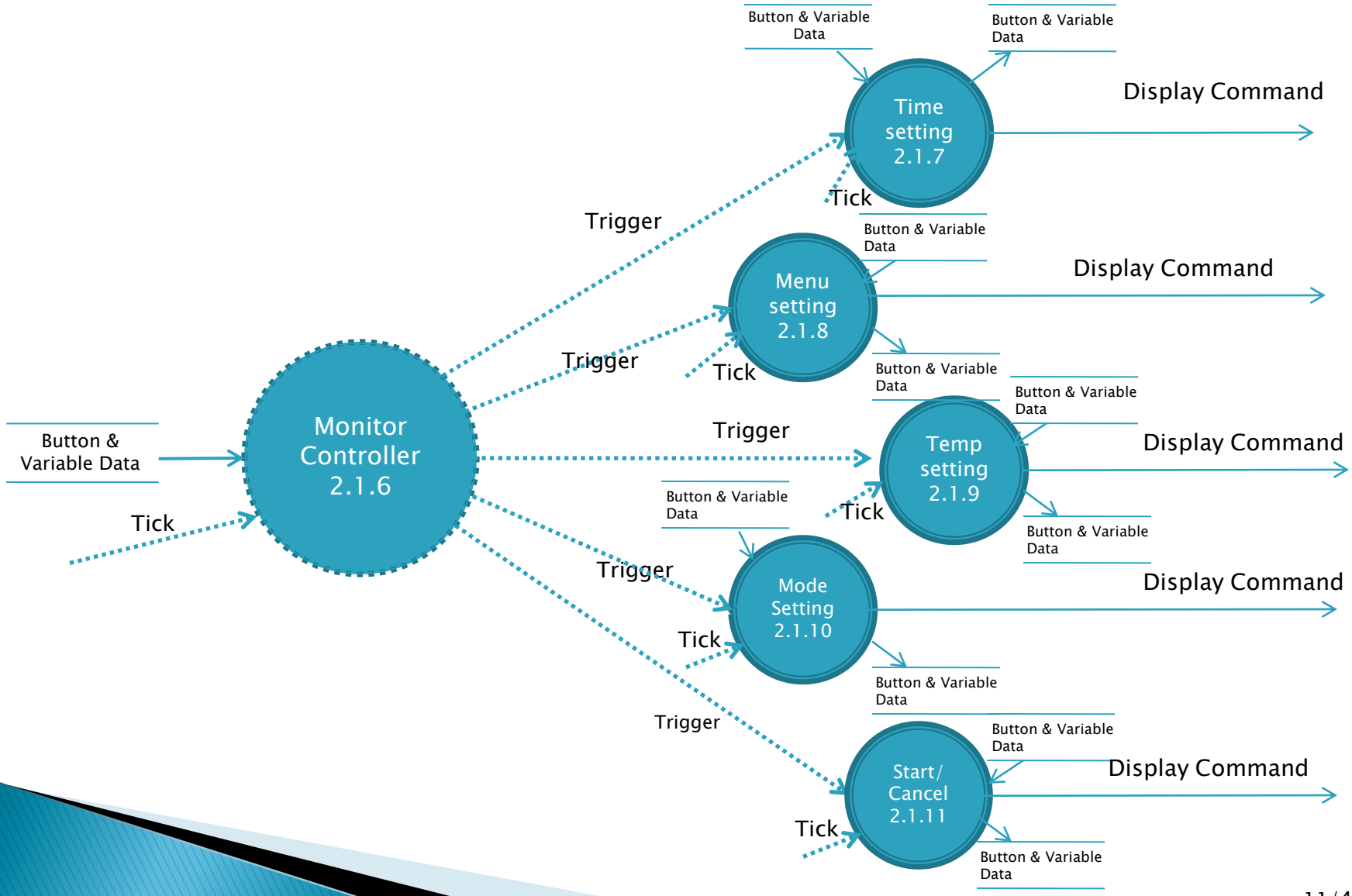
DFD Level 2


Button &
Variable Data
Monitor
2.2
Display
Display
Command
Button &
Variable Data
Display &
Sound &
Light
Control
2.1
Sound
Command
Speaker
2.3
Sound
Light
Command
Back Light
2.4
Light
Tick

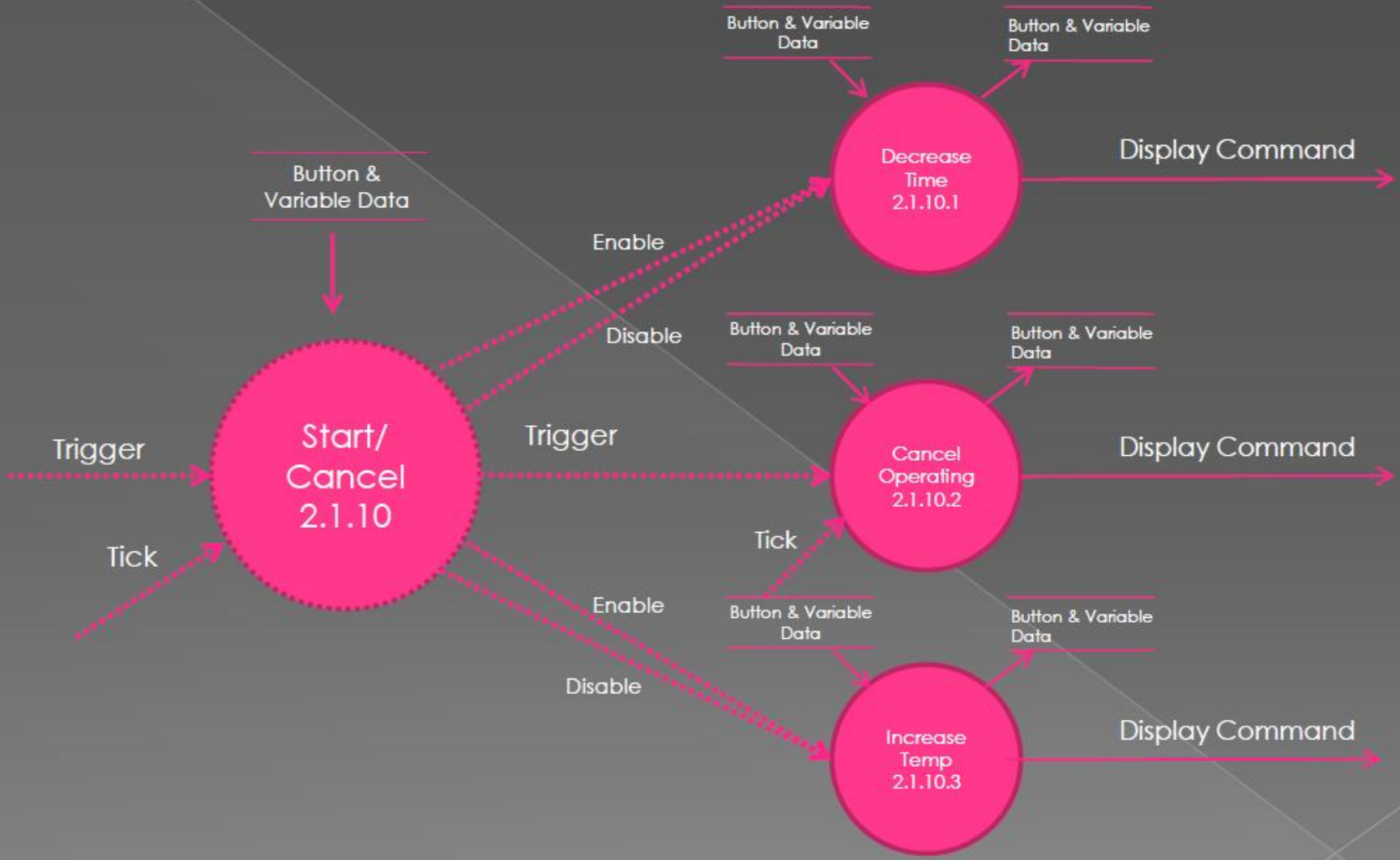
DFD (Level 3)


Button &
Variable Data
Tick
Light
Controller
2.1.1
Trigger
Tick
Light_
on
2.1.3
Light Command
Trigger
Tick
Light_
off
2.1.4
Light Command
Button &
Variable Data
Tick
Sound
Controller
2.1.2
enable
disable
Tick
Beep
Sound
2.1.5
Sound Command

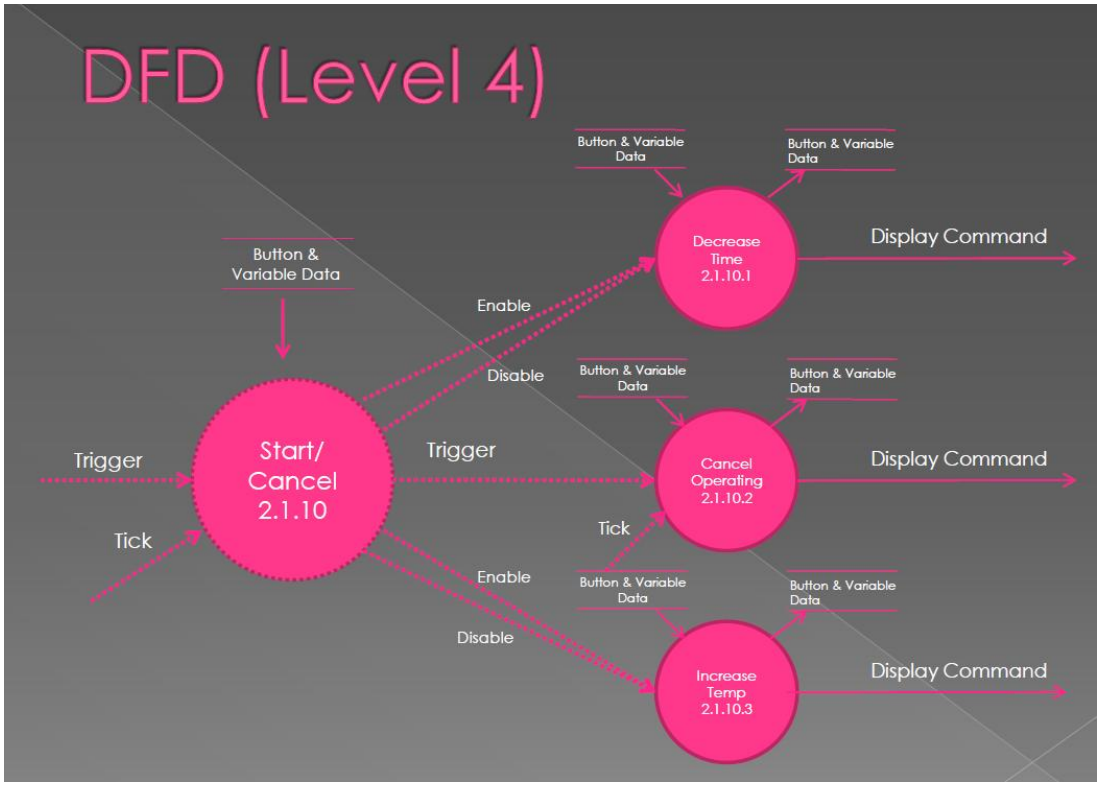
DFD (Level 3)

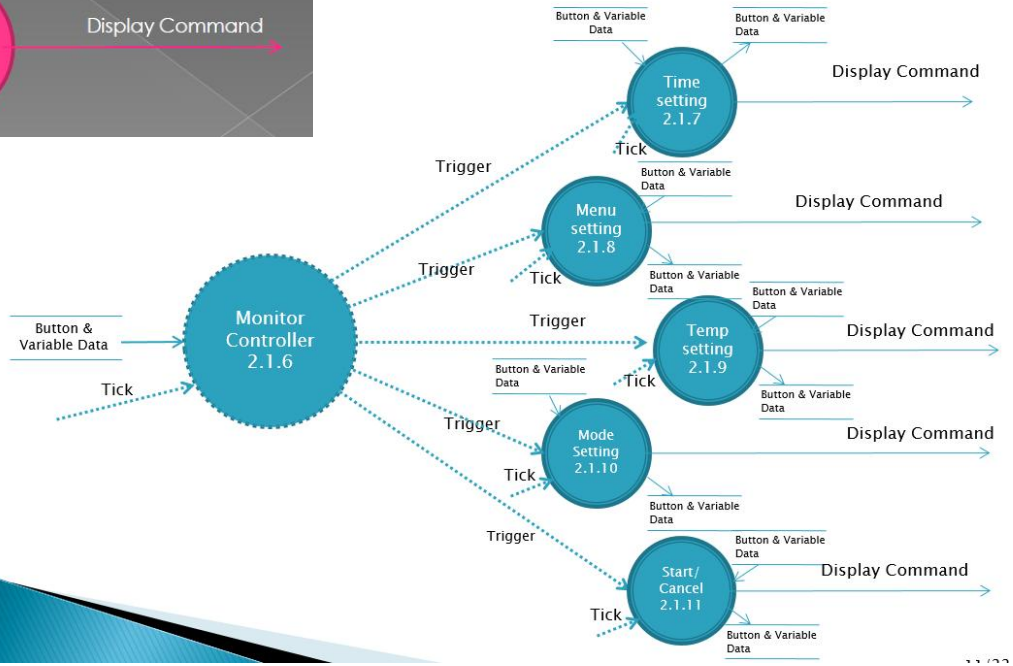

Button & Variable Data
Monitor Controller 2.1.6
Tick
Trigger
Trigger
Trigger
Trigger
Trigger
Time setting 2.1.7
Menu setting 2.1.8
Temp setting 2.1.9
Mode Setting 2.1.10
Start/ Cancel 2.1.11
Tick
Tick
Tick
Tick
Tick
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Button & Variable Data
Display Command
Display Command
Display Command
Display Command
Display Command

DFD (Level 4)


Button & Variable Data
Button & Variable Data
Decrease Time 2.1.10.1
Display Command
Button & Variable Data
Enable
Disable
Button & Variable Data
Button & Variable Data
Trigger
Start/ Cancel 2.1.10
Trigger
Cancel Operating 2.1.10.2
Display Command
Tick
Tick
Enable
Button & Variable Data
Button & Variable Data
Disable
Increase Temp 2.1.10.3
Display Command

DFD (Level 4)


Button & Variable Data
Button & Variable Data
Decrease Time 2.1.10.1
Display Command
Button & Variable Data
Enable
Disable
Button & Variable Data
Button & Variable Data
Trigger
Start/ Cancel 2.1.10
Trigger
Cancel Operating 2.1.10.2
Display Command
Tick
Tick
Enable
Button & Variable Data
Button & Variable Data
Disable
Increase Temp 2.1.10.3
Display Command


Button & Variable Data
Button & Variable Data
Time setting 2.1.7
Display Command
Tick
Trigger
Button & Variable Data
Menu setting 2.1.8
Display Command
Trigger
Tick
Button & Variable Data
Button & Variable Data
Button & Variable Data
Trigger
Temp setting 2.1.9
Display Command
Button & Variable Data
Tick
Button & Variable Data
Button & Variable Data
Trigger
Mode Setting 2.1.10
Display Command
Tick
Button & Variable Data
Trigger
Button & Variable Data
Start/ Cancel 2.1.11
Display Command
Tick
Button & Variable Data
Button & Variable Data
Monitor Controller 2.1.6
Tick

Data Dictionary

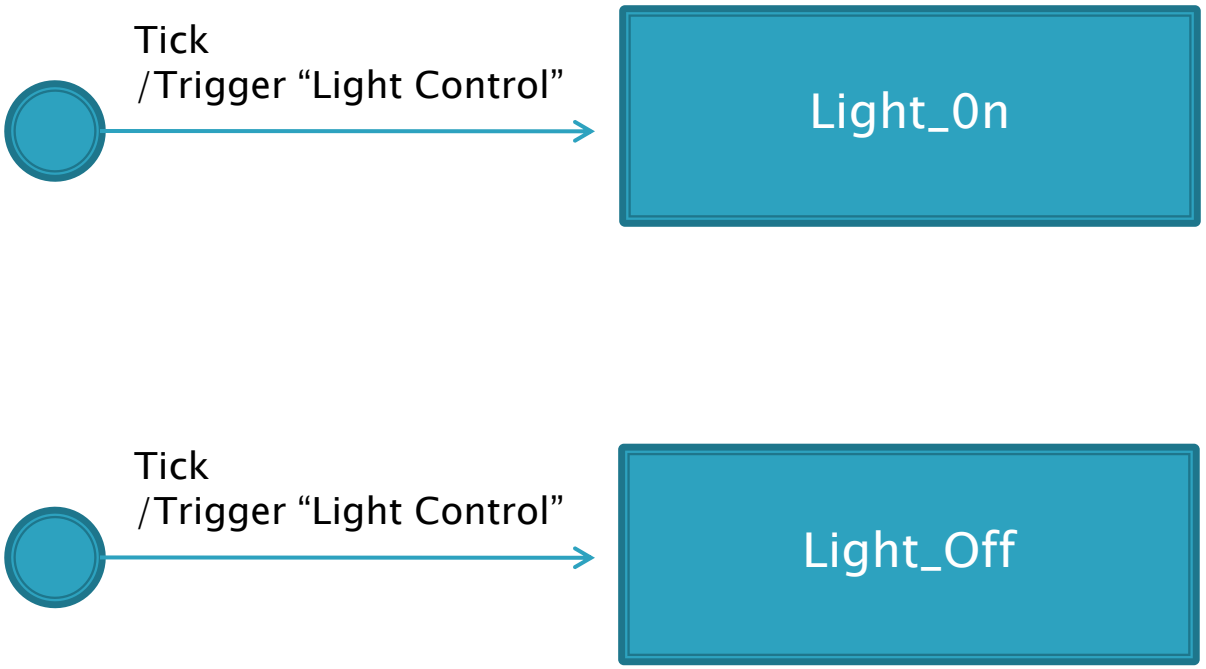
Input/Output Event	Description	Type	Format
Button A Signal	Button A Input이 true일 경우 조건에 따라 Integer형 변수 변경하는 명령을 Input Manger로 전달하는 signal	Interrupt	Integer
Button B Signal	Button A Input이 true일 경우 조건에 따라 Integer형 변수 변경하는 명령을 Input Manger로 전달하는 signal	Interrupt	Integer
Button C Signal	Button C Input이 true일 경우 Integer형 변수의 값을 변경하는 명령을 Input Manager로 전달하는 signal	Interrupt	Integer
Button D Signal	Button C Input이 true일 경우 Integer형 변수의 값을 변경하는 명령을 Input Manager로 전달하는 signal	Interrupt	Integer
Button E Signal	Button C Input이 true일 경우 Integer형 변수의 값을 변경하는 명령을 Input Manager로 전달하는 signal	Interrupt	Integer
Button F Signal	Button C Input이 true일 경우 Integer형 변수의 값을 변경하는 명령을 Input Manager로 전달하는 signal	Interrupt	Integer
Display Command	Sensor와 Button에서 받은 데이터를 바탕으로 Monitor에 명령을 전달	온도, 시간, 모드	String
Sound Command	Sensor와 Button에서 받은 데이터를 바탕으로 Speaker에 명령을 전달	Beep 소리	
Light Command	Sensor와 Button에서 받은 데이터를 바탕으로 Back Light에 명령을 전달	Back Light On/Off	
Button & Variable Data	Button A~F와 각각의 Controller에서 변경되고 입력되는 데이터를 저장하는 구조체	Interrupt	Struct

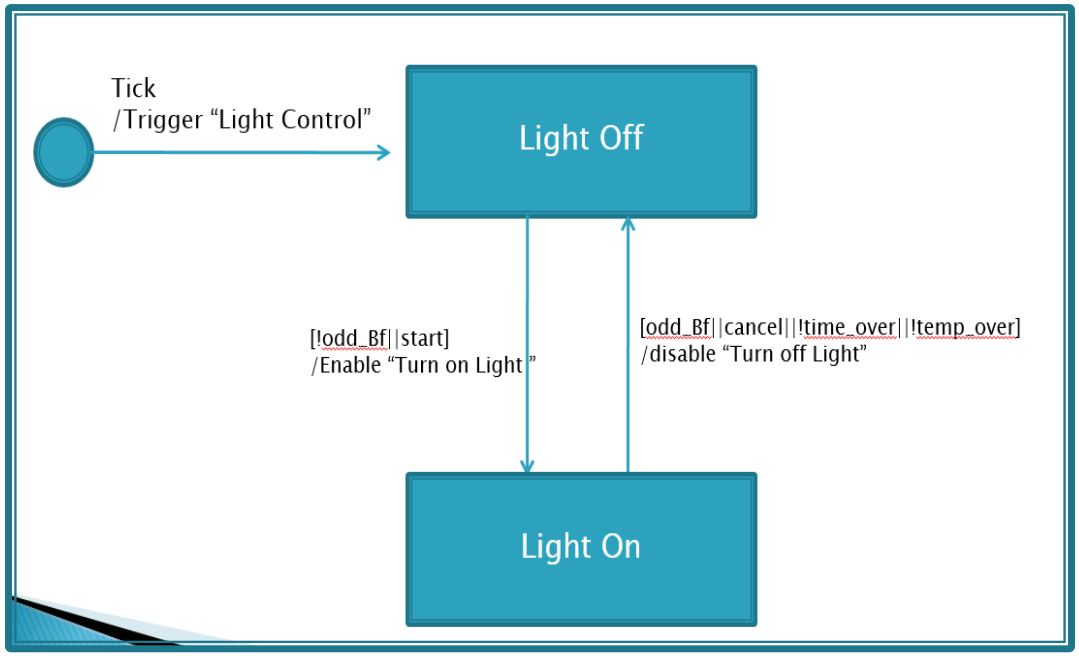
Data Dictionary (Button & Variable Data)

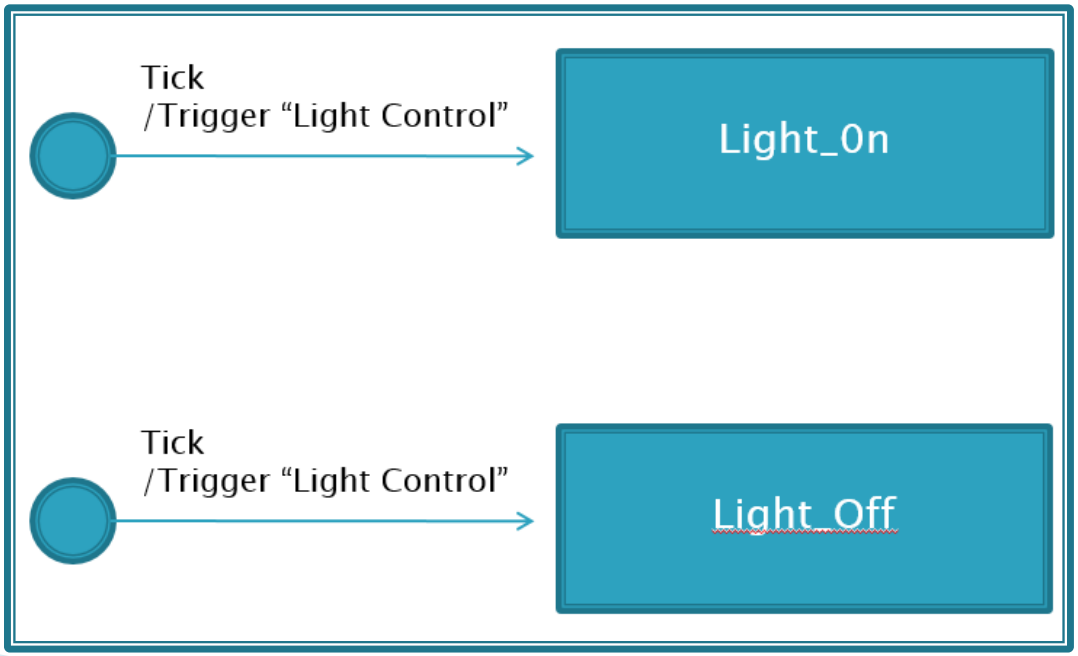
Value	Description	type	Format
c_Value	Button C가 눌리면 값이 0일 때는 1로 변경하고 1일 때는 0으로 변경하는 데이터, 초기값은 0	Interrupt	Integer
d_Value	Button D가 눌리면 값이 1씩 증가하는 변수, 초기값은 0, 변수 값이 5라면 다시 0으로 돌아간다.	Interrupt	Integer
e_Value	Button E가 눌리면 다른 값들과의 조건이 충족되면 0을 1로 1은 0으로 변경한다.	Interrupt	Integer
f_Value	Button F가 눌리면 값을 1씩 증가하는 변수, 초기값은 10, 변수 값이 10이면 다시 1로 돌아간다	Interrupt	Integer
button_A	Keyborad 에서 'a' 를 입력 받으면 true로 변경되는 변수, 초기값은 false	Interrupt	Boolean
button_B	Keyborad 에서 'b' 를 입력 받으면 true로 변경되는 변수, 초기값은 false	Interrupt	Boolean
button_C	Keyborad 에서 'c' 를 입력 받으면 true로 변경되는 변수, 초기값은 false	Interrupt	Boolean
button_D	Keyborad 에서 'd' 를 입력 받으면 true로 변경되는 변수, 초기값은 false	Interrupt	Boolean
button_E	Keyborad 에서 'e' 를 입력 받으면 true로 변경되는 변수, 초기값은 false	Interrupt	Boolean
button_F	Keyborad 에서 'f' 를 입력 받으면 true로 변경되는 변수, 초기값은 false	Interrupt	Boolean
time	C_value의 값이 0일 경우 Button A가 눌리면 10이 증가되고 Button B가 눌리면 30이 증가되는 변수	Interrupt	Integer
temp	C_value의 값이 1일 경우 Button A가 눌리면 10이 증가되고 Button B가 눌리면 20이 증가되는 변수	Interrupt	Integer
temp_s	초기 온도 값을 저장하는 Integer형 변수, 초기값은 20	Interrupt	Integer
sound			

DFD (Level 4)

State Transition Diagram for Light Controller 2.1.1

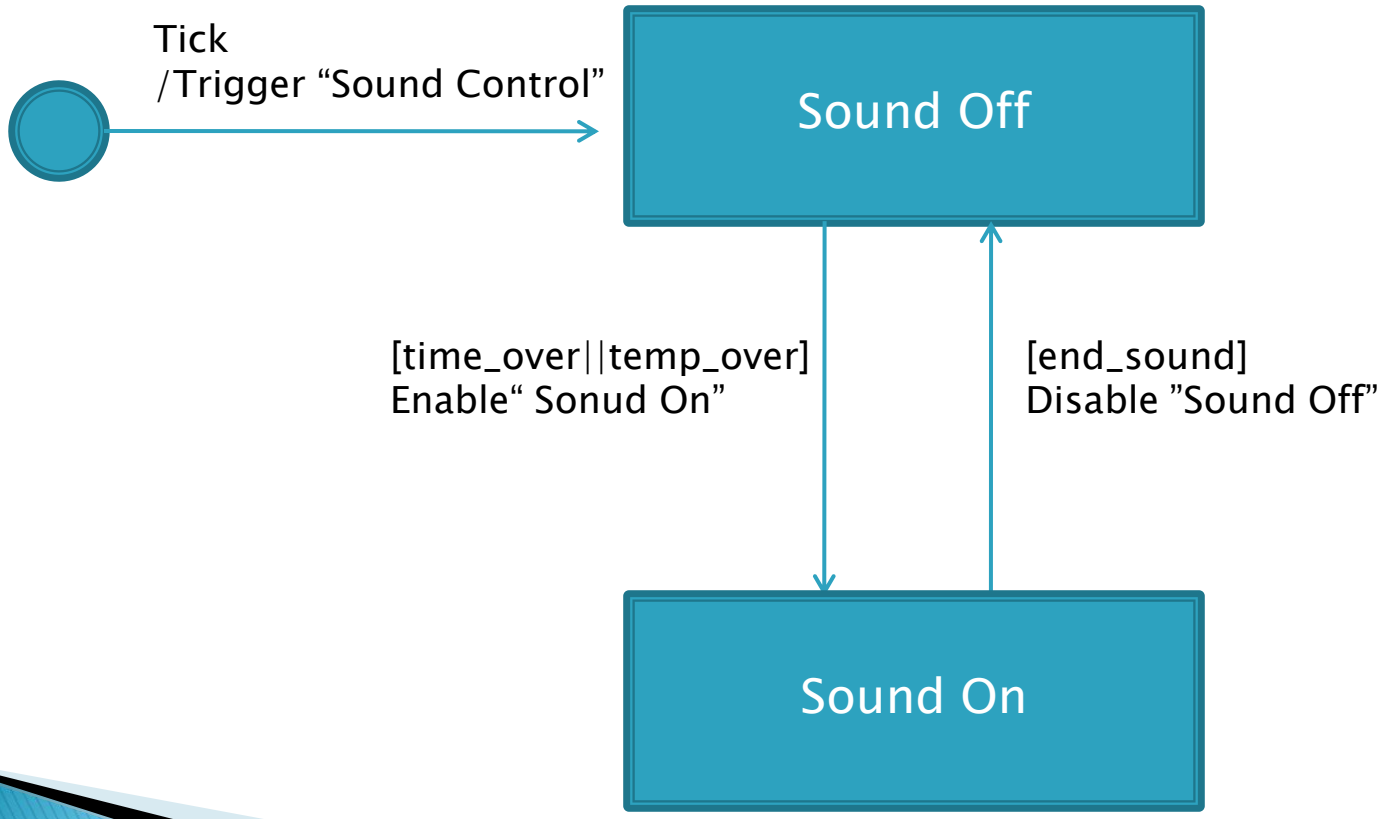



Tick
/Trigger “Light Control”
Light Off
[!odd_Bf||start]
/Enable “Turn on Light”
[odd_Bf||cancel||!time_over||!temp_over]
/disable “Turn off Light”
Light On


Tick
/Trigger “Light Control”
Light_On
Tick
/Trigger “Light Control”
Light_Off

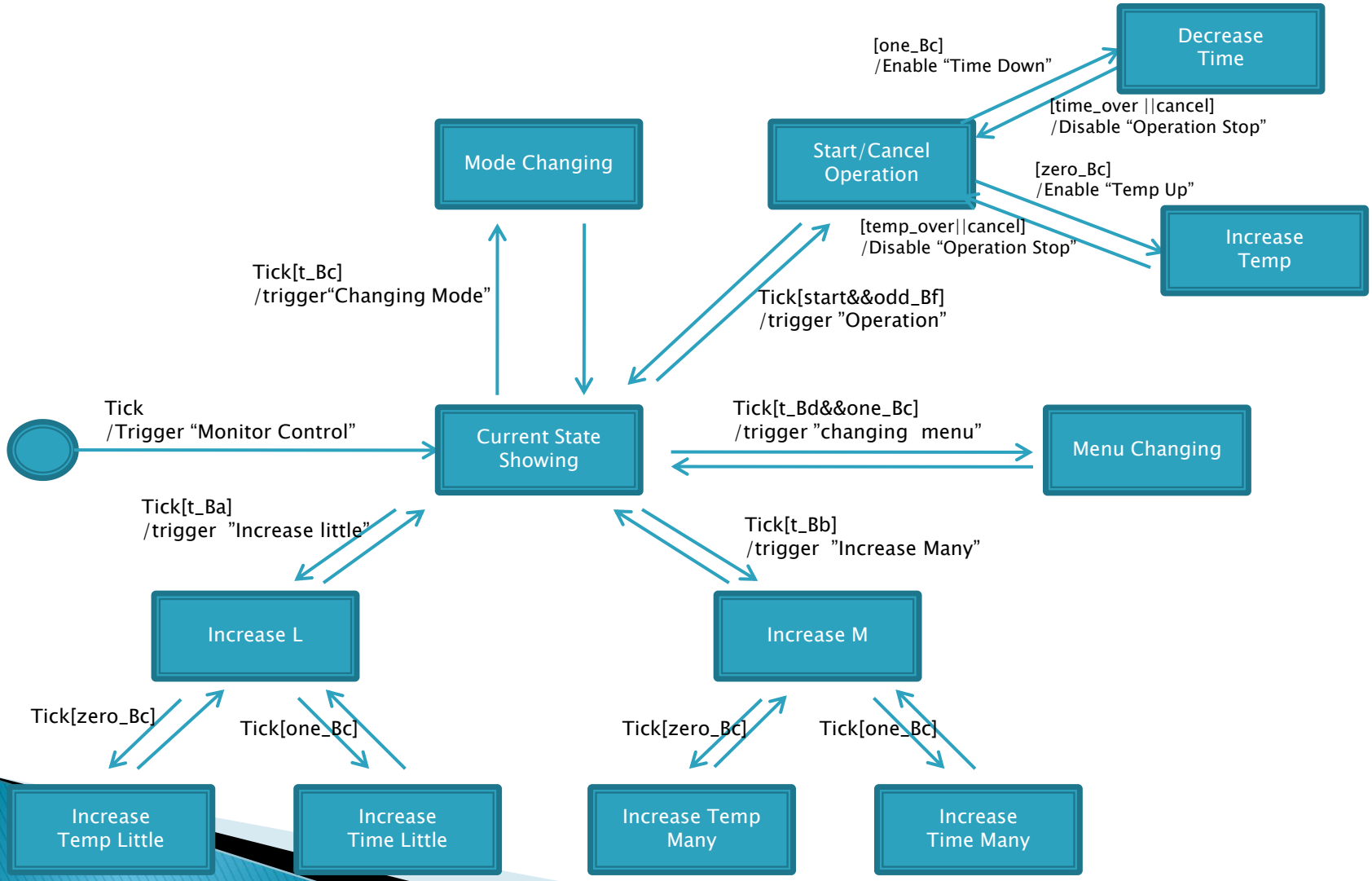
DFD (Level 4)

State Transition Diagram for Sound Controller 2.1.1



DFD (Level 4)

State Transition Diagram for Monitor Controller 2.1.5



Data Dictionary

Input/Output Event	Description	Type	Format
t_Ba	Button_Data.button_A == true	Interrupt	Boolean
t_Bb	Button_Data.button_B == true	Interrupt	Boolean
t_Bc	Button_Data.button_C == true	Interrupt	Boolean
t_Bd	Button_Data.button_D == true	Interrupt	Boolean
t_Be	Button_Data.button_E == true	Interrupt	Boolean
one_Bc	Variable_Data.c_Value == 1	Interrupt	Integer
zero_Bc	Variable_Data.c_Value == 0	Interrupt	Integer
start	Variable_Data.e_Value == 1	Interrupt	Integer
cancel	Variable_Data.e_Value == 0	Interrupt	Integer
odd_Bf	(Variable_Data.f_Value)%2==1	Interrupt	Integer
time_over	Variable_Data.time == 0	Periodic	Integer
temp_over	Variable_Data.temp – Variable_Data.temp_s == 0	Periodic	Integer
end_sound	Variable_sound == 3	Periodic	Integer

Process Specification

Number	1.1
Name	Button A Interface
Input	Button A Input, tick
Output	Button A Signal
Description	입력 받은 Button A Input의 값을 판단하여 true일 경우 정해진 Integer형 변수의 값을 변경하는 명령을 전달하는 Button A Signal을 출력

Number	1.2
Name	Button B Interface
Input	Button B Input, tick
Output	Button B Signal
Description	입력 받은 Button B Input의 값을 판단하여 true일 경우 정해진 Integer형 변수의 값을 변경하는 명령을 전달하는 Button B Signal을 출력

Process Specification

Number	1.3
Name	Button C Interface
Input	Button C Input, tick
Output	Button C Signal
Description	입력 받은 Button C Input의 값을 판단하여 true일 경우 정해진 Integer형 변수의 값을 변경하는 명령을 전달하는 Button C Signal을 출력

Number	1.4
Name	Button D Interface
Input	Button D Input, tick
Output	Button D Signal
Description	입력 받은 Button D Input의 값을 판단하여 true일 경우 정해진 Integer형 변수의 값을 변경하는 명령을 전달하는 Button D Signal을 출력

Process Specification

Number	1.5
Name	Button E Interface
Input	Button E Input, tick
Output	Button E Signal
Description	입력 받은 Button E Input의 값을 판단하여 true일 경우 정해진 Integer형 변수의 값을 변경하는 명령을 전달하는 Button E Signal을 출력

Number	1.6
Name	Button F Interface
Input	Button F Input, tick
Output	Button F Signal
Description	입력 받은 Button F Input의 값을 판단하여 true일 경우 정해진 Integer형 변수의 값을 변경하는 명령을 전달하는 Button F Signal을 출력

Process Specification

Number	1.7
Name	Input manager
Input	Button A,B,C,D,E,F Signal, Tick
Output	Button Data & Variable Data
Description	입력된 Button Signal에 따라서 Button Data와 Variable Data내의 변수 값을 변경해 준 뒤 다시 저장해주는 출력을 생성한다.

Number	2.1.1
Name	Light Controller
Input	Button Data & Variable Data, Tick
Output	Enable, Disable
Description	입력된 Button Data와 Variable Data에 따라서 Enable과 Disable을 출력해준다.

Process Specification

Number	2.1.2
Name	Sound Controller
Input	Button Data & Variable Data, Tick
Output	Enable, Disable
Description	입력된 Button Data와 Variable Data에 따라서 Enable과 Disable을 출력해준다.

Number	2.1.3
Name	Light_on
Input	Trigger,Tick
Output	Light Command
Description	입력된 Button Data와 Variable Data에 따라서 조건이 만족하면 Light on함수를 실행한다.

Process Specification

Number	2.1.4
Name	Light_off
Input	Trigger,Tick
Output	Light Command
Description	입력된 Button Data와 Variable Data에 따라서 조건이 만족하면 Light off함수를 실행한다.

Number	2.1.5
Name	Beep Sound
Input	Enable, Disable
Output	Sound Command
Description	Enable이 전달되면 Sound on, Disable이 전달되면 Sound off의 동작을 취한다

Process Specification

Number	2.1.6
Name	Display Controller
Input	Button Data & Variable Data, Tick
Output	Trigger
Description	Button Data & Variable Data의 상태를 체크하여 5가지 프로세스 중 적합한 곳으로 trigger를 출력한다.

Number	2.1.7
Name	Time Setting
Input	Trigger, Tick, Button Data & Variable Data
Output	Button Data & Variable Data, Display Command
Description	Trigger가 입력되면 Button Data & Variable Data를 체크하여 시간 모드인지 온도모드인지에 대한 display command를 보낸 후 변경된 Button Data & Variable Data를 저장하는 출력을 생성한다.

Process Specification

Number	2.1.8
Name	Menu Setting
Input	Trigger, Tick, Button Data & Variable Data
Output	Button Data & Variable Data, Display Command
Description	Trigger가 입력되면 Button Data & Variable Data를 체크하여 메뉴에 대한 display command를 보낸 후 변경된 Button Data & Variable Data를 저장하는 출력을 생성한다.

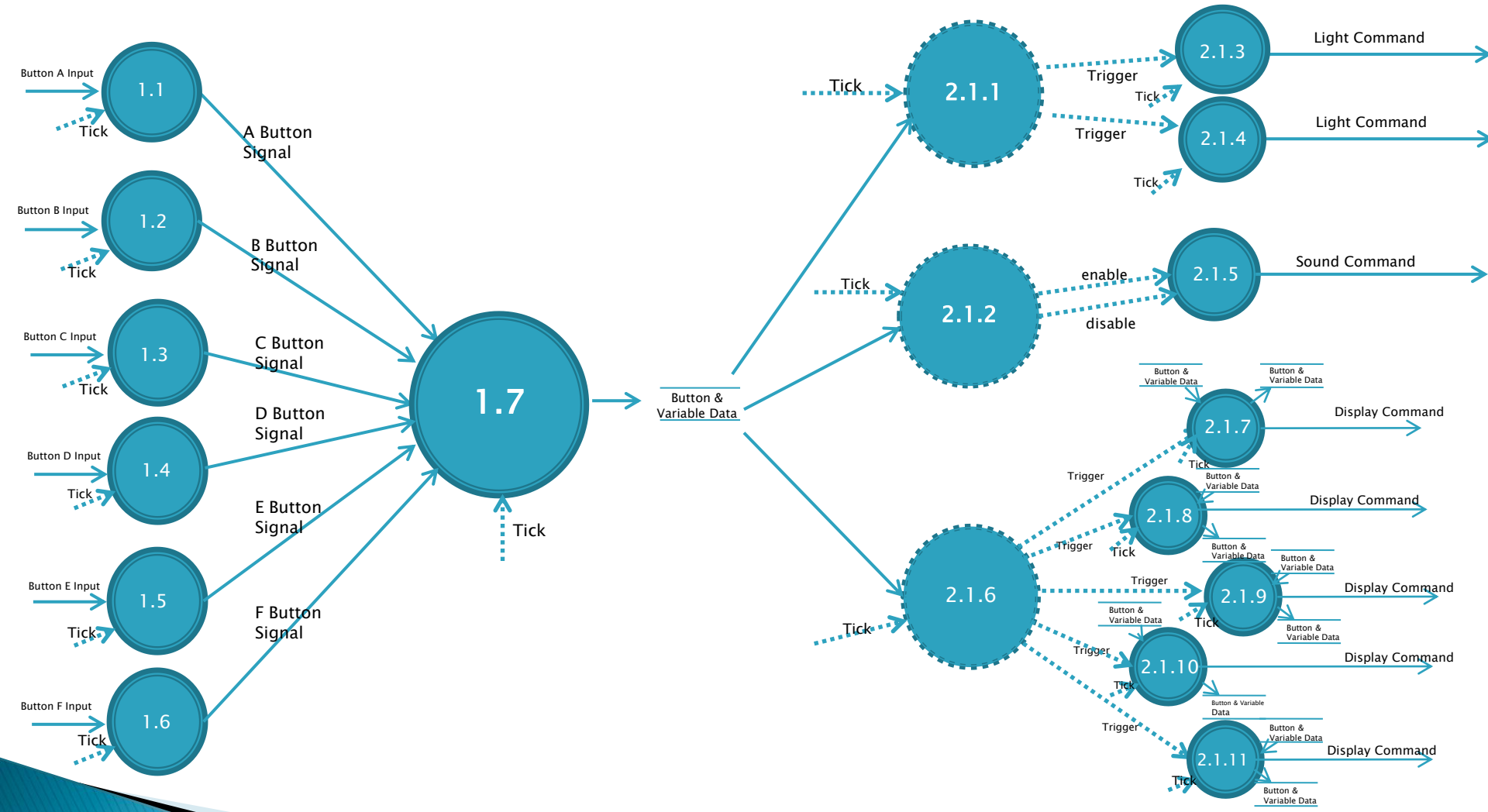
Number	2.1.9
Name	Temp Setting
Input	Trigger, Tick, Button Data & Variable Data
Output	Button Data & Variable Data, Display Command
Description	Trigger가 입력되면 Button Data & Variable Data를 체크하여 시간 모드인지 온도모드인지에 대한 display command를 보낸 후 변경된 Button Data & Variable Data를 저장하는 출력을 생성한다.

Process Specification

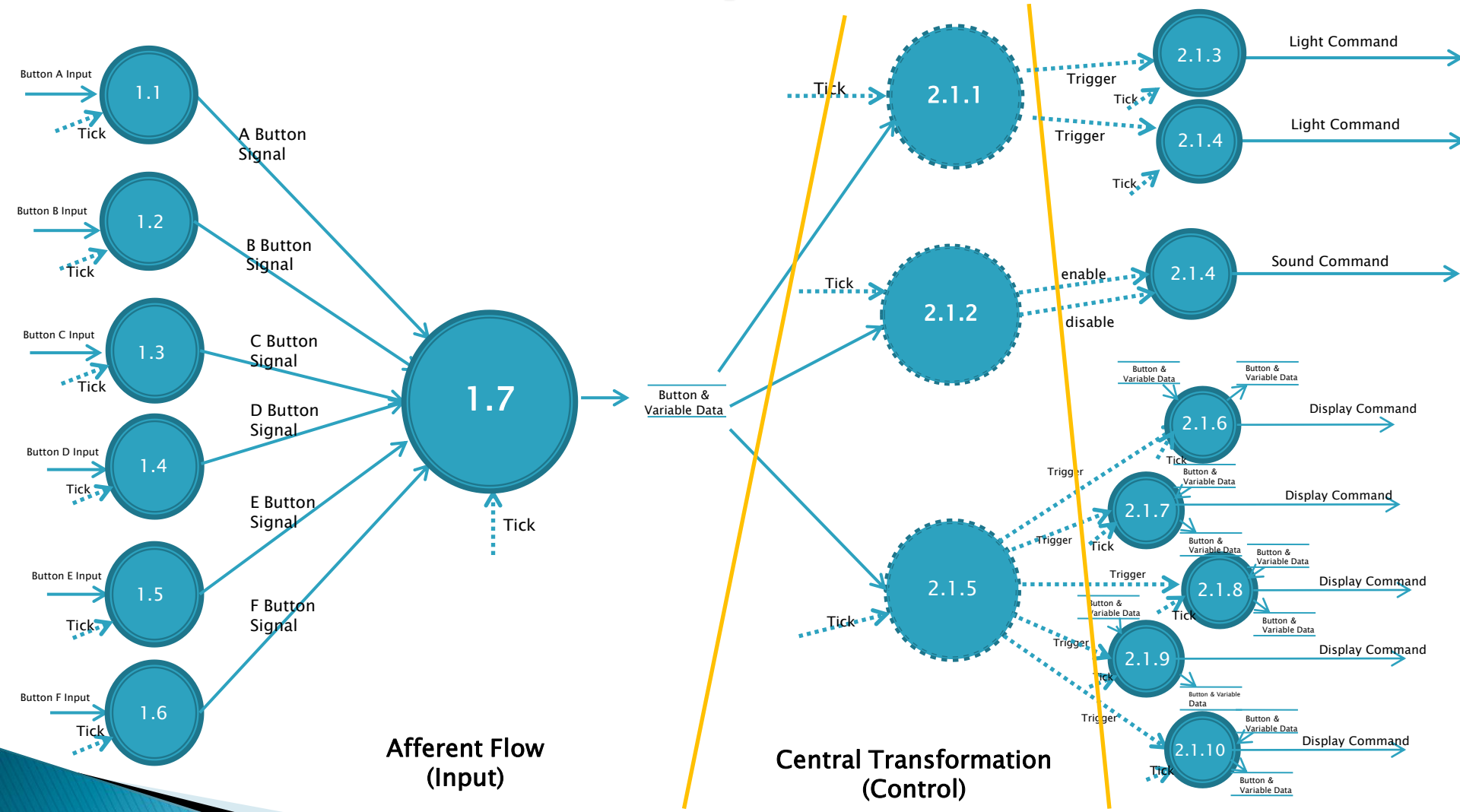
Number	2.1.10
Name	Mode Setting
Input	Trigger, Tick, Button Data & Variable Data
Output	Button Data & Variable Data, Display Command
Description	Trigger가 입력되면 Button Data & Variable Data를 체크하여 시간 모드인지 온도모드인지에 대한 display command를 보낸 후 변경된 Button Data & Variable Data를 저장하는 출력을 생성한다.

Number	2.1.11
Name	Start/Cancel
Input	Trigger, Tick, Button Data & Variable Data
Output	Button Data & Variable Data, Display Command
Description	Trigger가 입력되면 Button Data & Variable Data를 체크하여 시간 모드인지 온도모드인지에 대한 display command를 보낸 후 변경된 Button Data & Variable Data를 저장하는 출력을 생성한다.

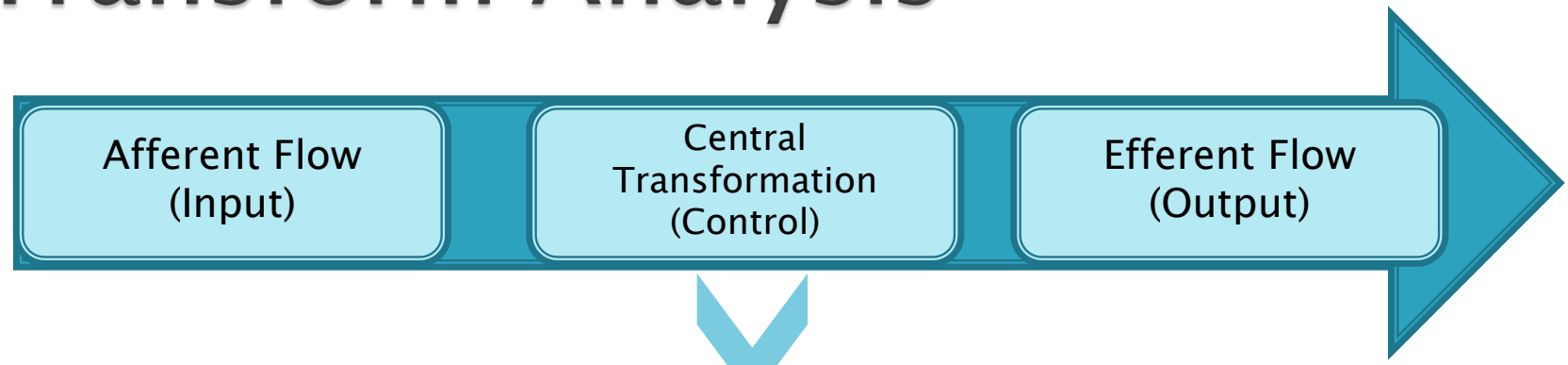
Overall

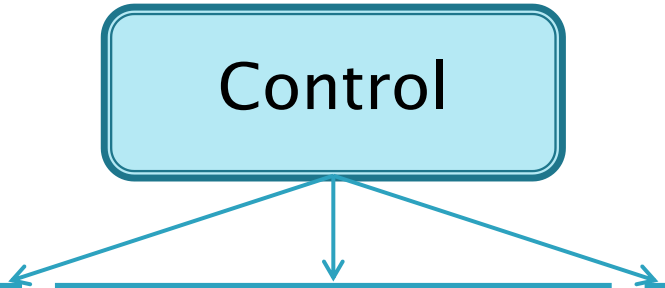

Button A Input
1.1
Tick
A Button Signal
Button B Input
1.2
Tick
B Button Signal
Button C Input
1.3
Tick
C Button Signal
Button D Input
1.4
Tick
D Button Signal
Button E Input
1.5
Tick
E Button Signal
Button F Input
1.6
Tick
F Button Signal
1.7
Tick
Button & Variable Data
Tick
2.1.1
Trigger
Tick
2.1.3
Light Command
Trigger
Tick
2.1.4
Light Command
Tick
Tick
2.1.2
enable
2.1.5
Sound Command
disable
Tick
2.1.6
Button & Variable Data
2.1.7
Display Command
Trigger
Tick
2.1.8
Display Command
Trigger
Tick
Trigger
2.1.9
Display Command
Tick
Trigger
2.1.10
Display Command
Tick
Trigger
2.1.11
Display Command
Tick

Transform Analysis



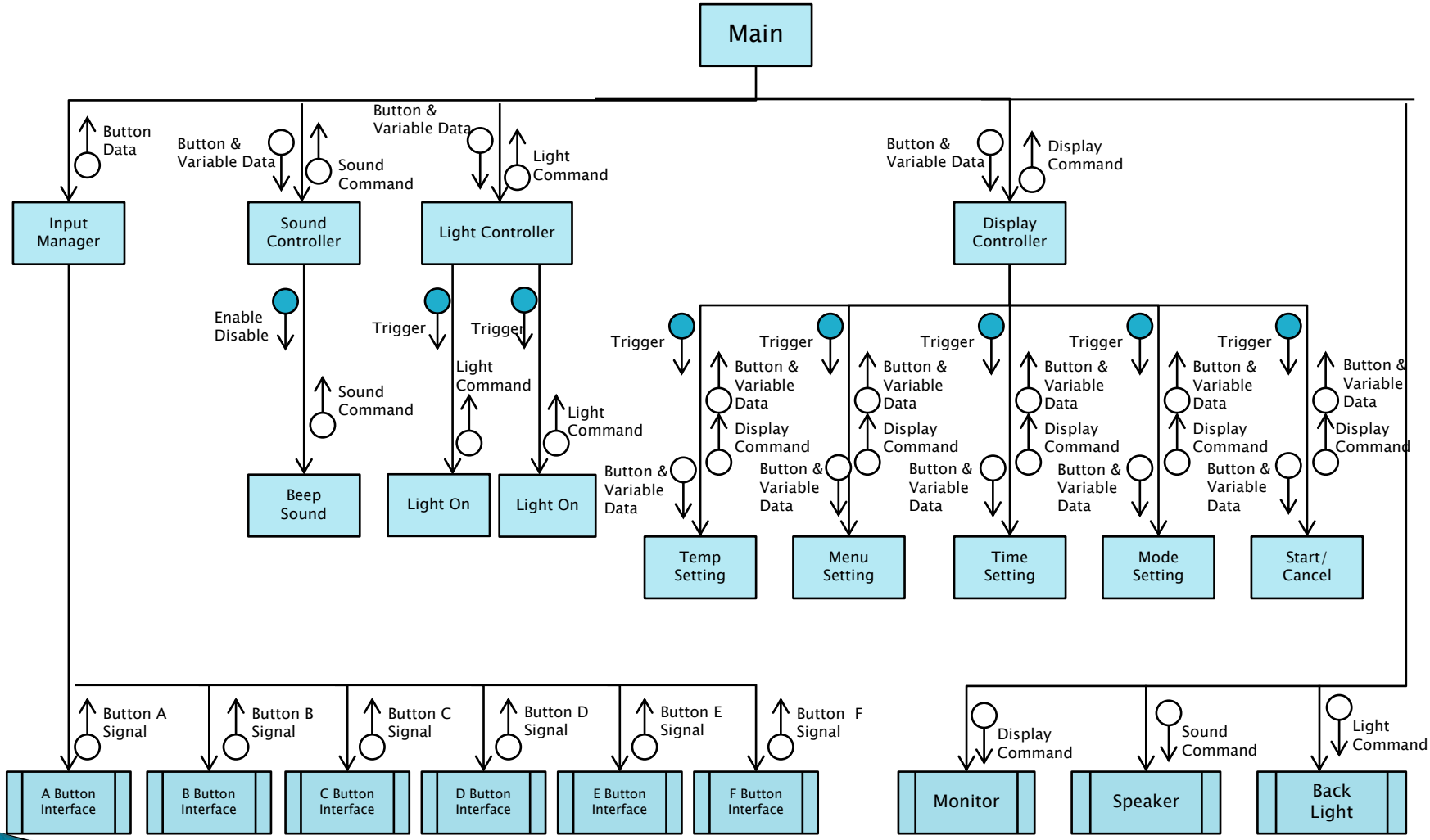
Transform Analysis



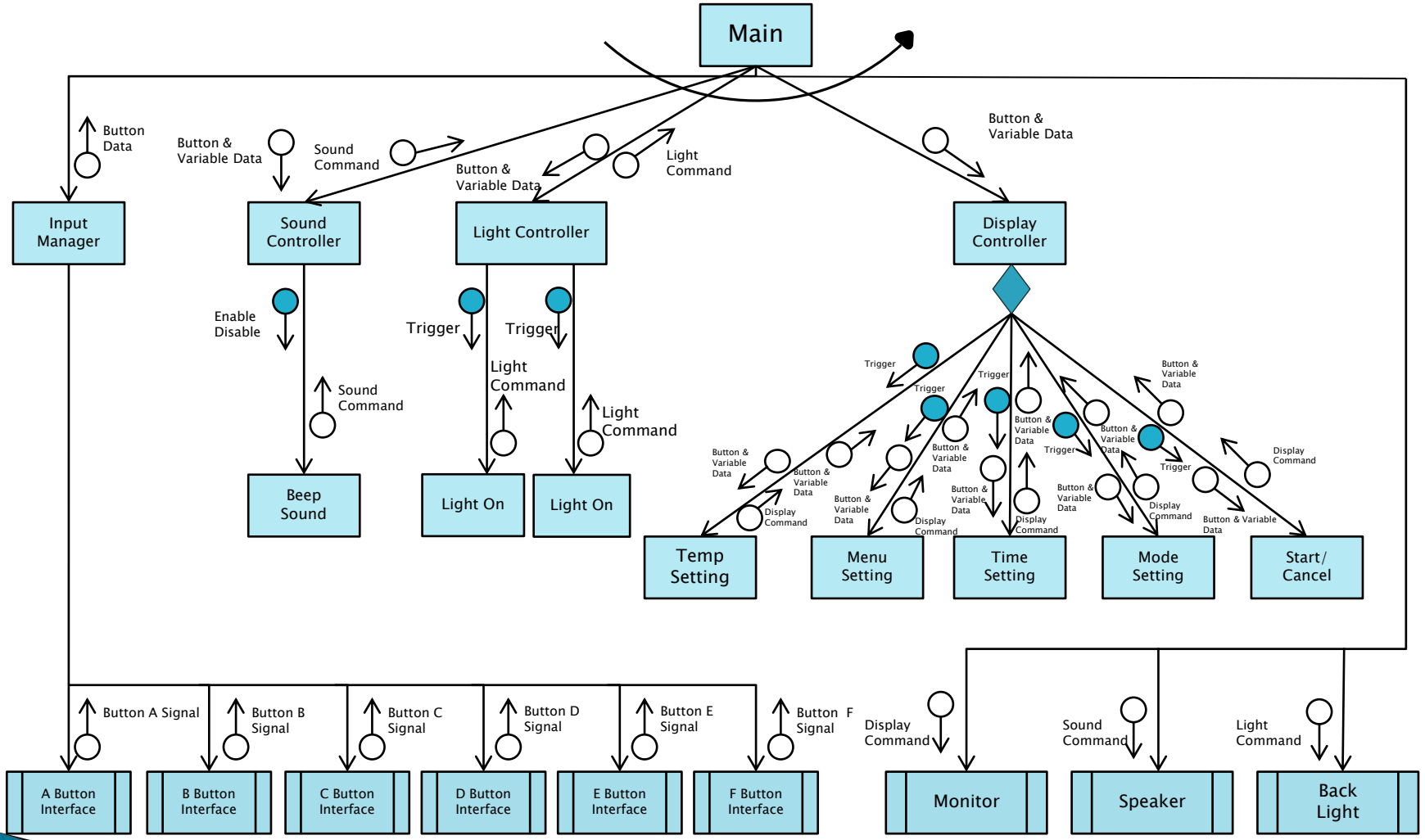


Input	Process	Output
⊙ Button A~F ⊙ Tick ⊙ Button & Variable Data	⊙ Sound Controller ⊙ Display Controller ⊙ Light Controller	⊙ Button & Variable Data ⊙ Enable/Disable ⊙ Trigger

Structured Charts – MOS (Basic)



Structured Charts – MOS (Advanced)



Implementation

My_lib.h

```c
#include <stdio.h>
#include <windows.h>
#include <conio.h>

#define MAX_X 75
#define MAX_Y 25
#define MIN_X 2
#define MIN_Y 1
#define A 65
#define B 66
#define C 67
#define D 68
#define E 69
#define F 70
#define BLACK 0
#define BLUE 9
#define GREEN 10
#define SKY 11
#define RED 12
#define PINK 13
#define YELLOW 14
#define WHITE 15
#define delay(n) Sleep(n)
```

```c
int button_A_interface();
int button_B_interface();
int button_C_interface();
int button_D_interface();
int button_E_interface();
int button_F_interface();
void Input_manager(int key);
void Monitor(void);
void Monitor_controller(void);
void Temp_setting(void);
void Time_setting(void);
void Menu_setting(void);
void Mode_setting(void);
void gotoxy(int x, int y);
void set_color(int val);
void Start_cancel(void);
void Sound_controller(void);
void Beep_sound(void);
void Light_on(void);
void Light_off(void);
void Light_controller(void);
```

```c
struct variable_data
{
    int c_value;
    int d_value;
    int e_value;
    int f_value;
    int time;
    int temp;
    int sound;
    int temp_s;
};
```

Variable_data 구조체

```c
struct button_data
{
    boolean b_A;
    boolean b_B;
    boolean b_C;
    boolean b_D;
    boolean b_E;
    boolean b_F;
```

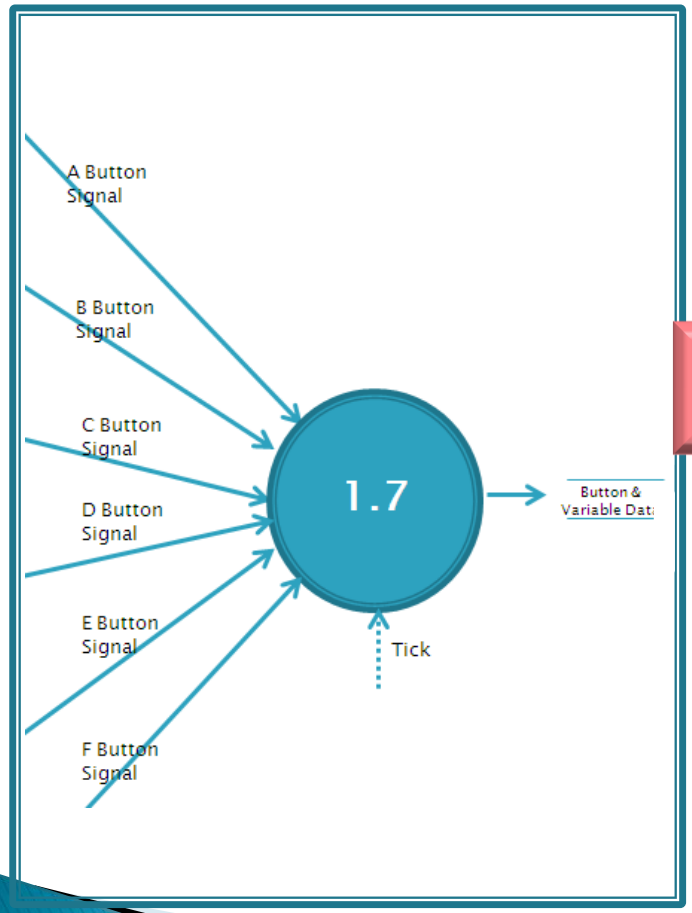
Button_data 구조체

Implementation

Main.c

```
#include "my_lib.h"

struct button_data button_Data={0,0,0,0,0,0};
struct variable_data variable_Data={1,0,0,0,0,20,0,20};

int main(void)
{
    char key;
    Monitor();
    while(1)
    {
        if(kbhit())
        {
            key = getch();
            if('a'==key)
                button_Data.b_A = 1;
            if('b'==key)
                button_Data.b_B = 1;
            if('c'==key)
                button_Data.b_C = 1;
            if('d'==key)
                button_Data.b_D = 1;
            if('e'==key)
                button_Data.b_E = 1;
            if('f'==key)
                button_Data.b_F = 1;
        }
        if(1 == button_Data.b_A)
            Input_manager(button_A_interface());
        if(1 == button_Data.b_B)
            Input_manager(button_B_interface());
        if(1 == button_Data.b_C)
            Input_manager(button_C_interface());
        if(1 == button_Data.b_D)
            Input_manager(button_D_interface());
        if(1 == button_Data.b_E)
            Input_manager(button_E_interface());
        if(1 == button_Data.b_F)
            Input_manager(button_F_interface());

        Monitor_controller();
        Sound_controller();
        Light_controller();
        delay(70);
    }

    return 0;
```

Implementation

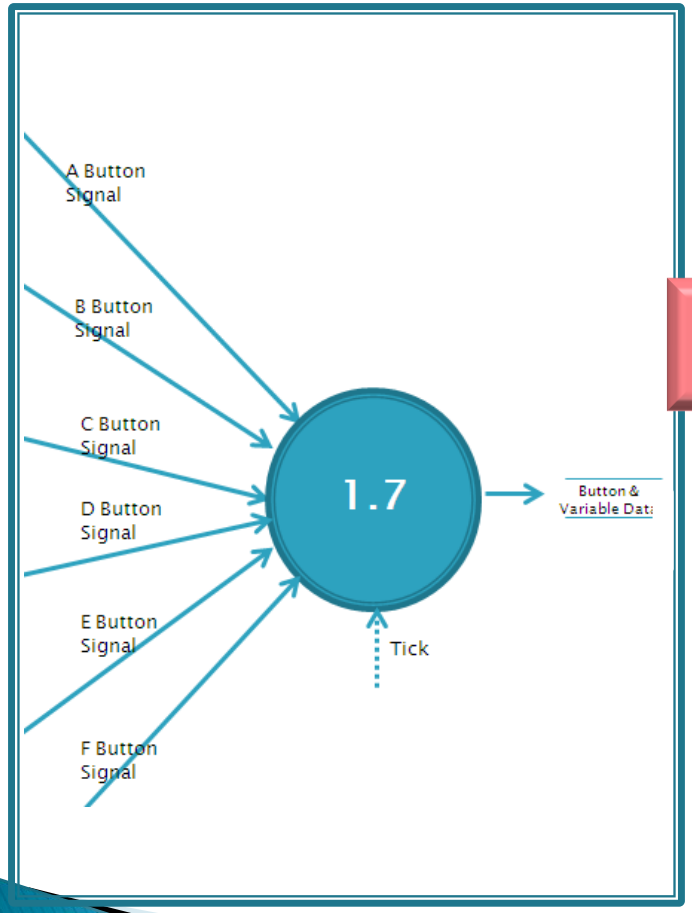


Input Manager

```
void Input_manager(int key)
{
    if(1==key)
    {
        if(1==variable_Data.c_value && 0==variable_Data.d_value)
        {
            if(variable_Data.time==590)
                variable_Data.time=600;
            else if(variable_Data.time>=600)
                variable_Data.time=10;
            else
                variable_Data.time += 10;
        }
        else if(0==variable_Data.c_value && 0==variable_Data.d_value)
        {
            if(variable_Data.temp == 80)
                variable_Data.temp = 90;
            else if(variable_Data.temp >=90)
                variable_Data.temp = 30;
            else
                variable_Data.temp +=10;
        }
    }
    else if(2==key)
    {
        if(1==variable_Data.c_value && 0==variable_Data.d_value)
        {
            if(variable_Data.time>=570 &&variable_Data.time <600)
                variable_Data.time=600;
            else if(variable_Data.time==600)
                variable_Data.time=30;
            else
                variable_Data.time+=30;
        }
        else if(0==variable_Data.c_value && 0==variable_Data.d_value)
        {
            if(variable_Data.temp >=70 && variable_Data.temp <90)
                variable_Data.temp = 90;
            else if(variable_Data.temp>=90)
                variable_Data.temp = 40;
            else
                variable_Data.temp +=20;
        }
    }
```

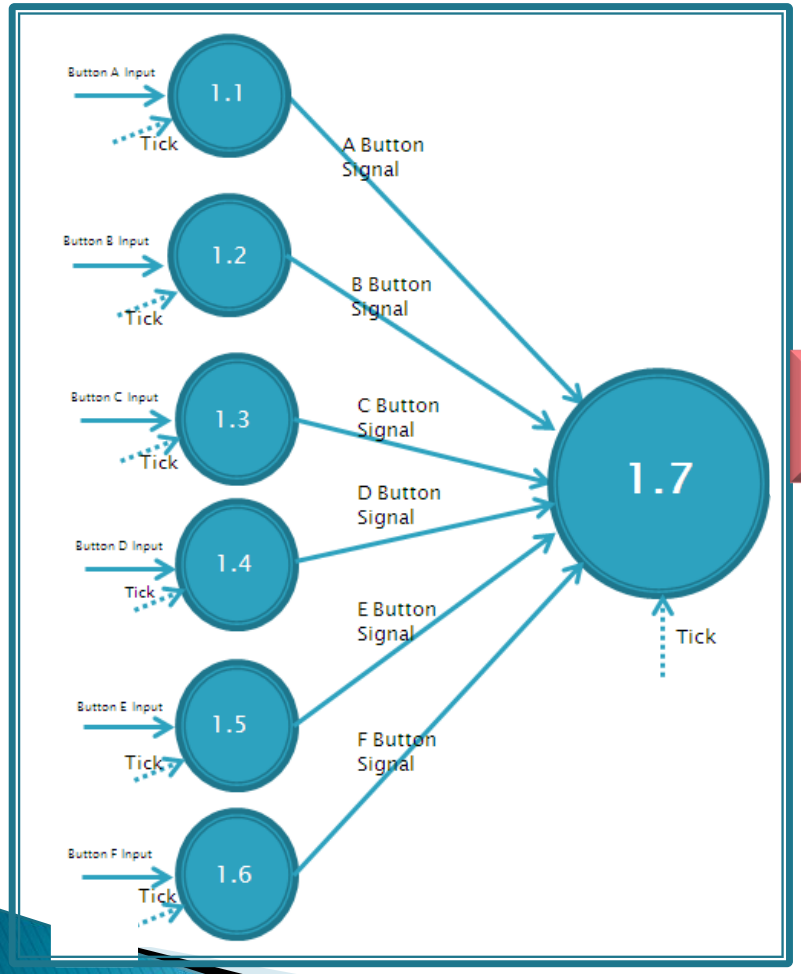
Implementation

Input Manager




```
else if(4==key)
{
    if(variable_Data.c_value==1)
    {
        if(variable_Data.d_value==5)
            variable_Data.d_value = 0;
        else
            variable_Data.d_value +=1;

        switch(variable_Data.d_value)
        {
            case 0:
                {
                    variable_Data.time = 0;
                    break;
                }
            case 1:
                {
                    variable_Data.time = 60;
                    break;
                }
            case 2:
                {
                    variable_Data.time = 90;
                    break;
                }
            case 3:
                {
                    variable_Data.time = 120;
                    break;
                }
            case 4:
                {
                    variable_Data.time = 300;
                    break;
                }
            case 5:
                {
                    variable_Data.time = 120;
                    break;
                }
        }
    }
}
else if(5==key)
{
```

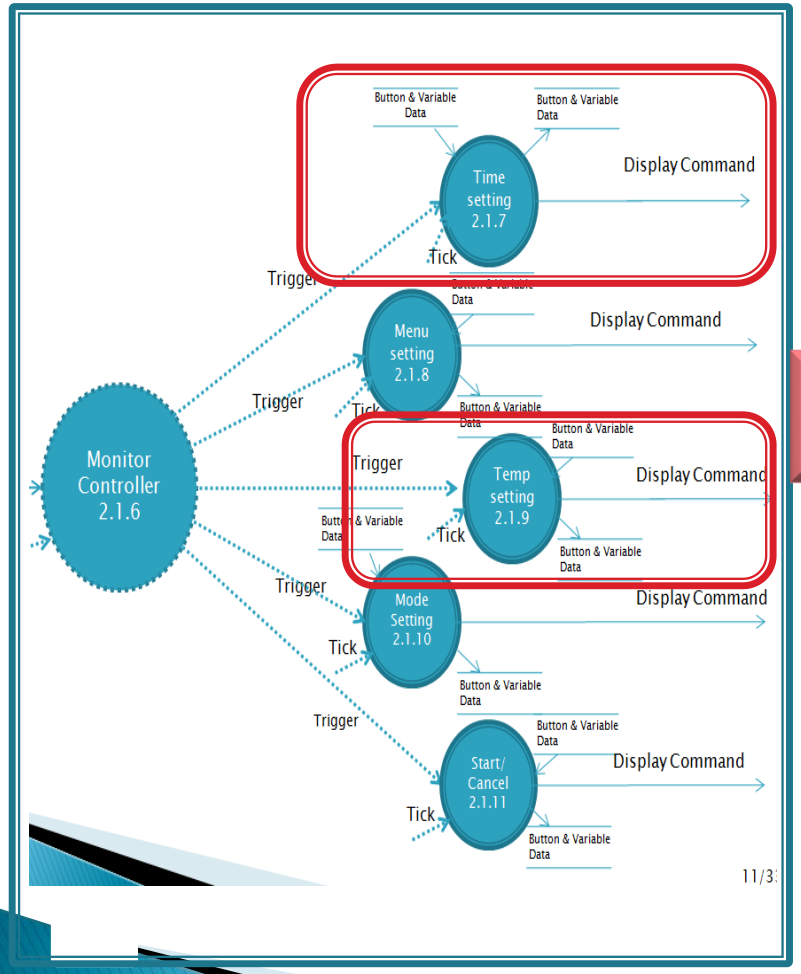
Implementation



Interface

```
#include "my_lib.h"

extern struct button_data button_Data;
extern  struct variable_data variable_Data;

int button_A_interface()
{
    return 1;
}
int button_B_interface()
{
    return 2;
}
int button_C_interface()
{
    return 3;
}
int button_D_interface()
{
    return 4;
}
int button_E_interface()
{
    return 5;
}
int button_F_interface()
{
    return 6;
}
```

Implementation

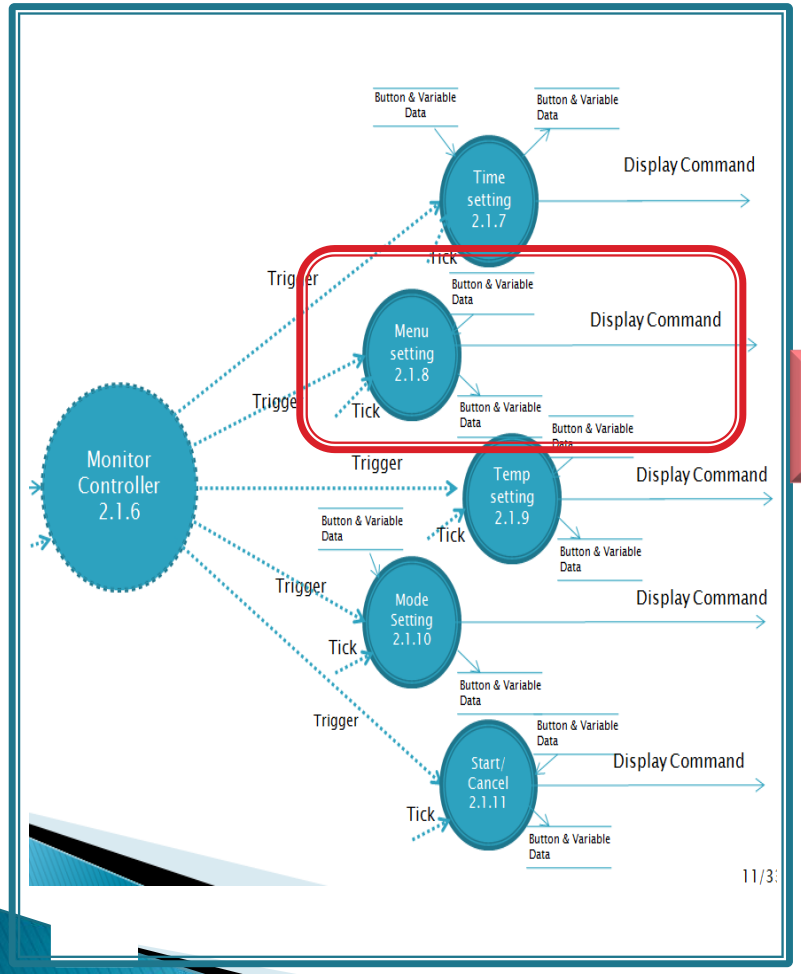


Monitor_controller

```
#include "my_lib.h"

extern struct button_data button_Data;
extern  struct variable_data variable_Data;

void Monitor_controller()
{
    if((button_Data.b_A==1 || button_Data.b_B==1)&&variable_Data.c_value==1&&variable_Data.d_value==0)
        Time_setting();
    if((button_Data.b_A==1 || button_Data.b_B==1)&&variable_Data.c_value==0&&variable_Data.d_value==0)
        Temp_setting();
    if(variable_Data.c_value==1&&button_Data.b_D==1)
        Menu_setting();
    if(button_Data.b_C==1&&variable_Data.d_value==0)
        Mode_setting();
    if(button_Data.b_E==1)
        Start_cancel();
}
void Temp_setting(void)
{
    gotoxy(54,6);
    printf("    %d℃ / %d℃\n",variable_Data.temp_s,variable_Data.temp);
    button_Data.b_A=0;
    button_Data.b_B=0;
    Monitor();
}
void Time_setting(void)
{
    gotoxy(54,4);
    printf("      %d : %d \n",variable_Data.time/60,variable_Data.time%60);
    button_Data.b_A=0;
    button_Data.b_B=0;
    Monitor();
}
```

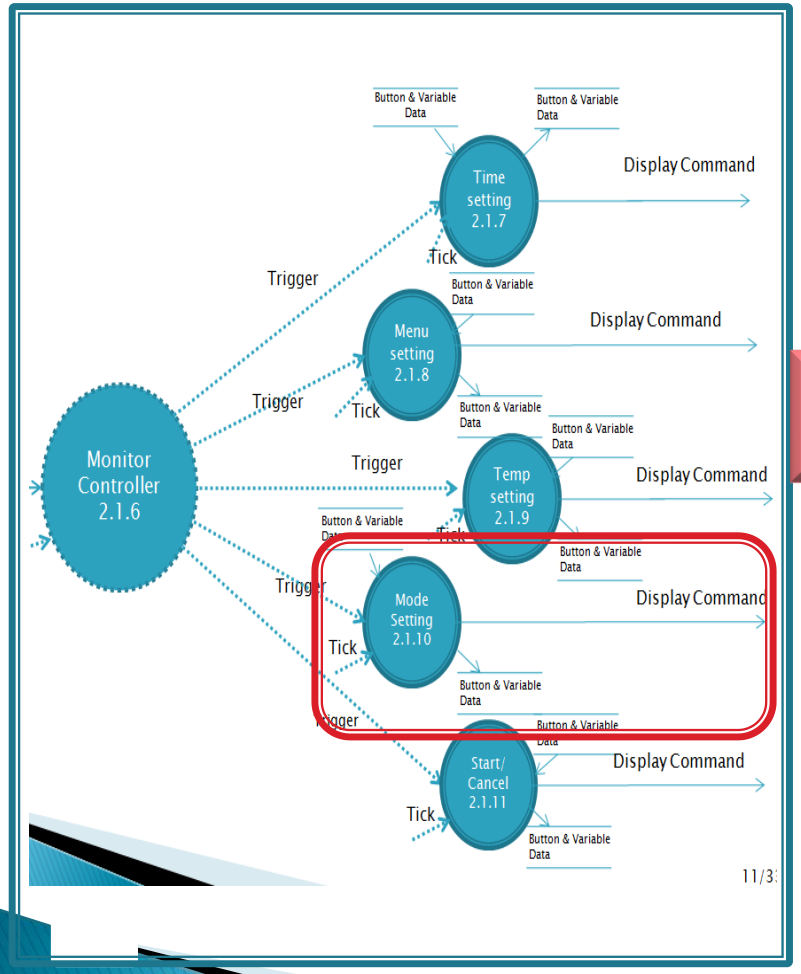
Implementation

Monitor_controller




```
void Menu_setting(void)
{
    gotoxy(54,8);
    switch(variable_Data.d_value)
    {
        case 0:
            {
                printf("   00 : manual\n");
                break;
            }
        case 1:
            {
                printf("   01 :   떡  \n");
                break;
            }
        case 2:
            {
                printf("   02 :   죽  \n");
                break;
            }
        case 3:
            {
                printf("   03 :   밥  \n");
                break;
            }
        case 4:
            {
                printf("   04 : 국/찌개\n");
                break;
            }
        case 5:
            {
                printf("   05 :  피자  \n");
                break;
            }
    }
    button_Data.b_D=0;
    Monitor();

}
```

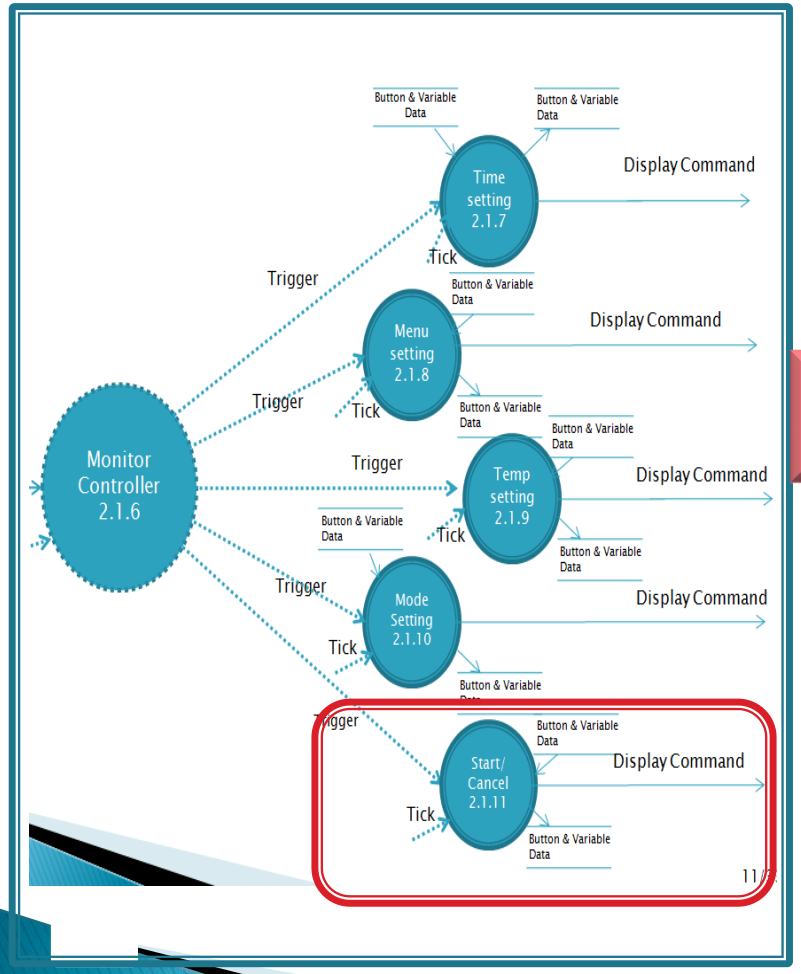
Implementation



Monitor_controller

```
void Mode_setting(void)
{
    gotoxy(54,10);
    if(variable_Data.c_value==0)
        printf("       Temp\n");
    if(variable_Data.c_value==1)
        printf("       Time\n");
    button_Data.b_C=0;
    Monitor();
}
```

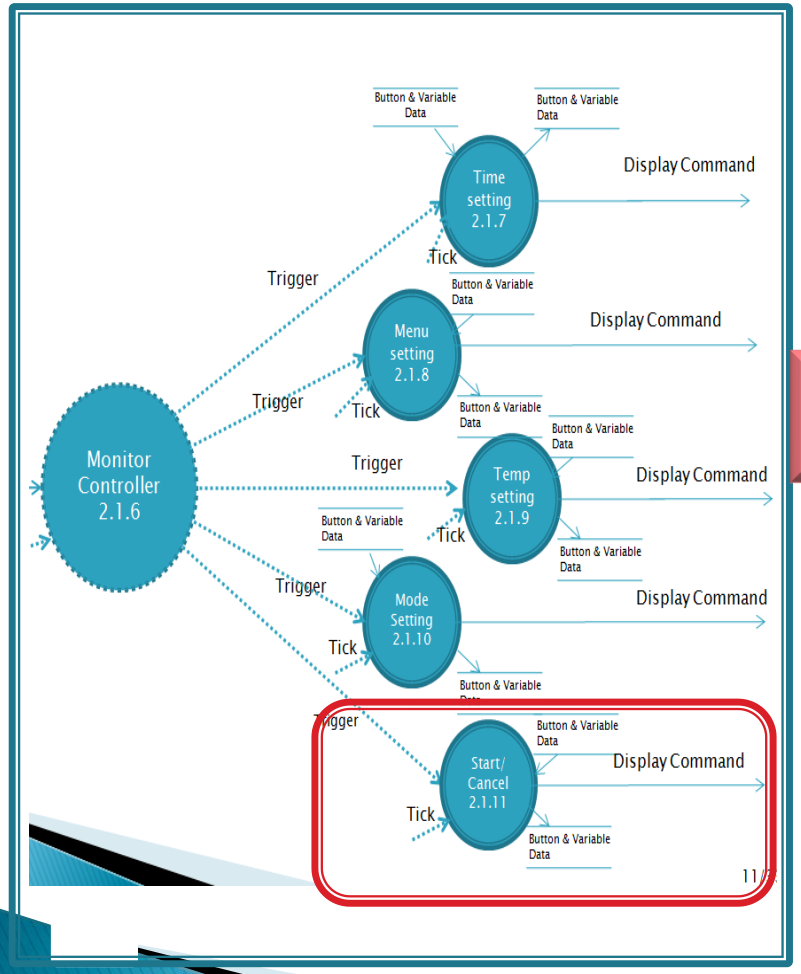
Implementation



Monitor_controller

```
void Start_cancel(void)
{
    char ch=0;
    int i=10;
    if(variable_Data.e_value==1)
    {
        button_Data.b_E=0;
        if(variable_Data.c_value==0)
        {
            while(variable_Data.temp>variable_Data.temp_s)
            {
                delay(800);
                if(kbhit())
                {
                    ch=getch();
                    if(ch=='e')
                    {
                        variable_Data.temp=20;
                        variable_Data.temp_s=20;
                        variable_Data.d_value=0;
                        Monitor();
                        break;
                    }
                }
                variable_Data.temp_s += 1;
                Monitor();
            }
            if(ch!='e')
            {
                variable_Data.sound=1;
                variable_Data.f_value=1;
            }
            variable_Data.temp=20;
            variable_Data.temp_s=20;
            variable_Data.d_value=0;
            Monitor();
        }
```

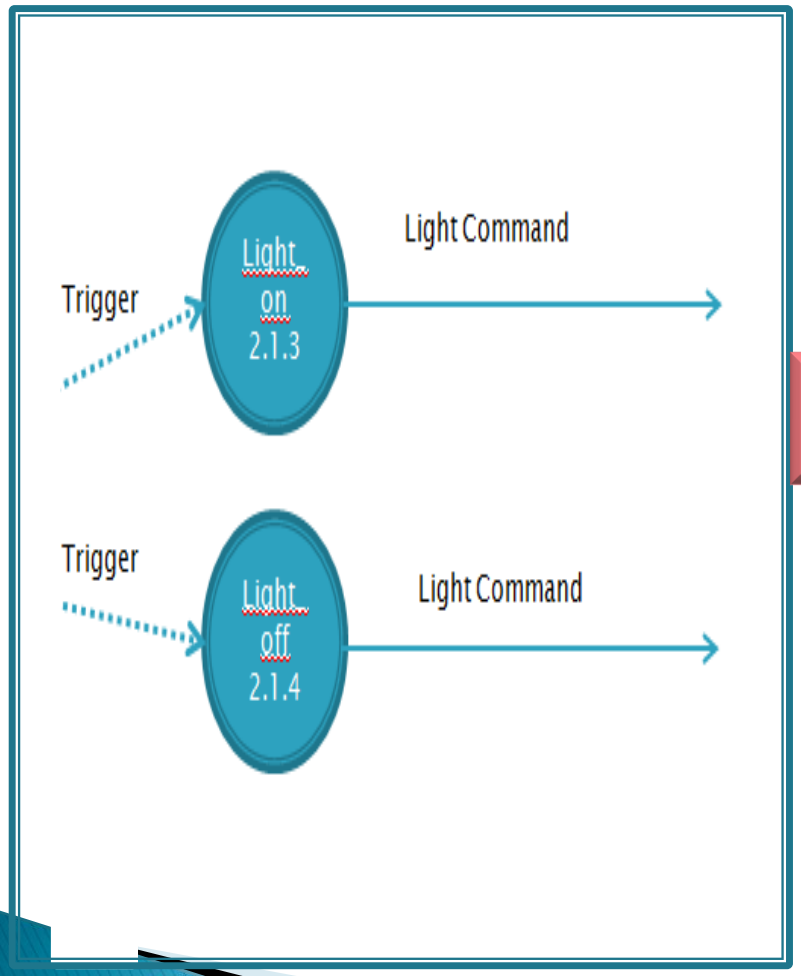
Implementation



Monitor_controller

```
if(variable_Data.c_value==1)
{
    while(variable_Data.time>0)
    {
        delay(800);
        if(kbhit())
        {
            ch=getch();
            if(ch=='e')
            {
                variable_Data.time=0;
                variable_Data.d_value=0;
                Monitor();
                break;
            }
        }
        variable_Data.time -= 1;
        Monitor();
    }
    if(ch!='e')
    {
        variable_Data.sound=1;
        variable_Data.f_value=1;
    }
    variable_Data.time=0;
    variable_Data.d_value=0;

}
}
```

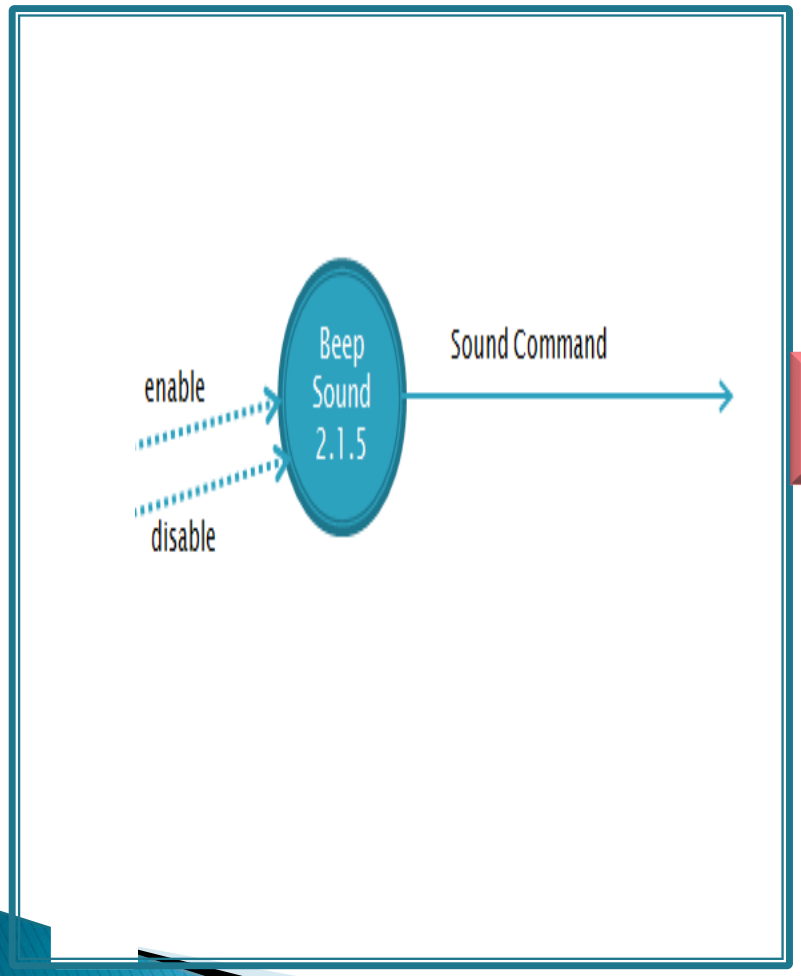
Implementation



Light_controller.c

```
#include "my_lib.h"

extern struct button_data button_Data;
extern struct variable_data variable_Data;

void Light_controller(void)
{
    if(variable_Data.f_value%2==1)
        Light_on();
    if(variable_Data.f_value%2==0)
        Light_off();
}
void Light_on(void)
{
    set_color(14);
    gotoxy(63,12);printf("°\n");
    gotoxy(58,13);printf("°\n");
    gotoxy(63,13);printf("°\n");
    gotoxy(68,13);printf("°\n");
    gotoxy(60,14);printf("°\n");
    gotoxy(66,14);printf("°\n");
    gotoxy(63,15);printf("◎\n");
    set_color(15);
}
void Light_off(void)
{
    set_color(0);
    gotoxy(63,12);printf("°\n");
    gotoxy(58,13);printf("°\n");
    gotoxy(63,13);printf("°\n");
    gotoxy(68,13);printf("°\n");
    gotoxy(60,14);printf("°\n");
    gotoxy(66,14);printf("°\n");
    gotoxy(63,15);printf("◎\n");
    set_color(15);
}
```

Implementation



Sound_controller.c

```
#include "my_lib.h"

extern struct button_data button_Data;
extern struct variable_data variable_Data;

void Sound_controller(void)
{
    if(variable_Data.sound==1)
        Beep_sound();
}
void Beep_sound(void)
{
    int i=10;
    while(i>0)
    {
        printf("\a");
        i--;
    }
    variable_Data.sound=0;
}
```

Anyone have question?

The End……^^