Electronic Door Lock System
1st Demo Test

Team 5

강민우 201211324 ← Presenter
임동현 201211353
서동현 201211375
함진아 201211389

Contents

- DEVELOPMENT ENVIRONMENT
- MODIFICATION
- ABOUT CODE
- DEMO
- FEEDBACK

Development Environment

Development Environment

개발 환경 : **Visual Studio 2010**
컴파일러 : **MinGW**
컴파일 환경 : **Eclipse**

- 입력 관련 함수 : MinGW 컴파일에는 문제가 없지만, Eclipse 실행 시 콘솔 입력 방식 사용 불가능
- Window관련 함수 사용을 위해 먼저 Visual Studio에서 구현
- Visual Studio에서 구현한 코드를 Eclipse로 옮긴 후, MinGW를 이용 하여 컴파일
- 컴파일 한 실행 파일을 cmd를 이용하여 실행

Development Environment

개발 환경 : **Visual Studio 2010**
컴파일러 : **MinGW**
컴파일 환경 : **Eclipse**

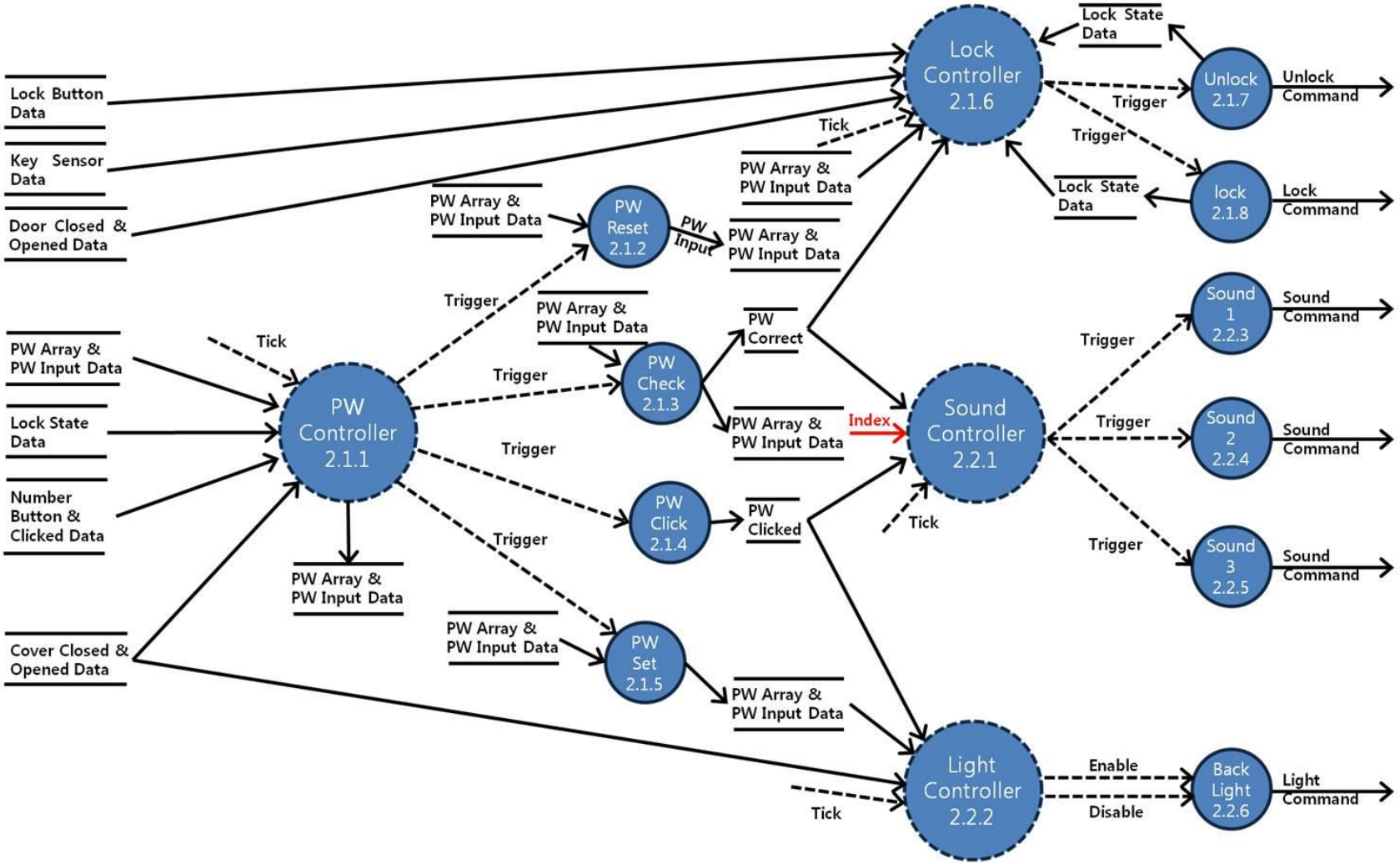
```
00:21:30 **** Incremental Build of configuration Debug for project EDLS ****
Info: Internal Builder is used for build
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\Sound_Controller.o" "..\\src\\Sound_Controller.c"
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\Light_Controller.o" "..\\src\\Light_Controller.c"
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\UI.o" "..\\src\\UI.c"
..\src\UI.c: In function 'num_pad_ui':
..\src\UI.c:34:20: warning: suggest parentheses around '&&' within '||' [-Wparentheses]
  if((CSS != FALSE) && (lflag^light_cmd) || nflag != NBS){
                    ^
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\determine_signal.o" "..\\src\\determine_signal.c"
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\pw_control2.o" "..\\src\\pw_control2.c"
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\main.o" "..\\src\\main.c"
gcc -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\Lock_Controller.o" "..\\src\\Lock_Controller.c"
gcc -o EDLS.exe "src\\pw_control2.o" "src\\main.o" "src\\determine_signal.o" "src\\UI.o" "src\\Sound_Controller.o" "src\\Lock_Controller.o" "src\\Light_Controller.o"

00:21:34 Build Finished (took 4s.732ms)
```

Modification

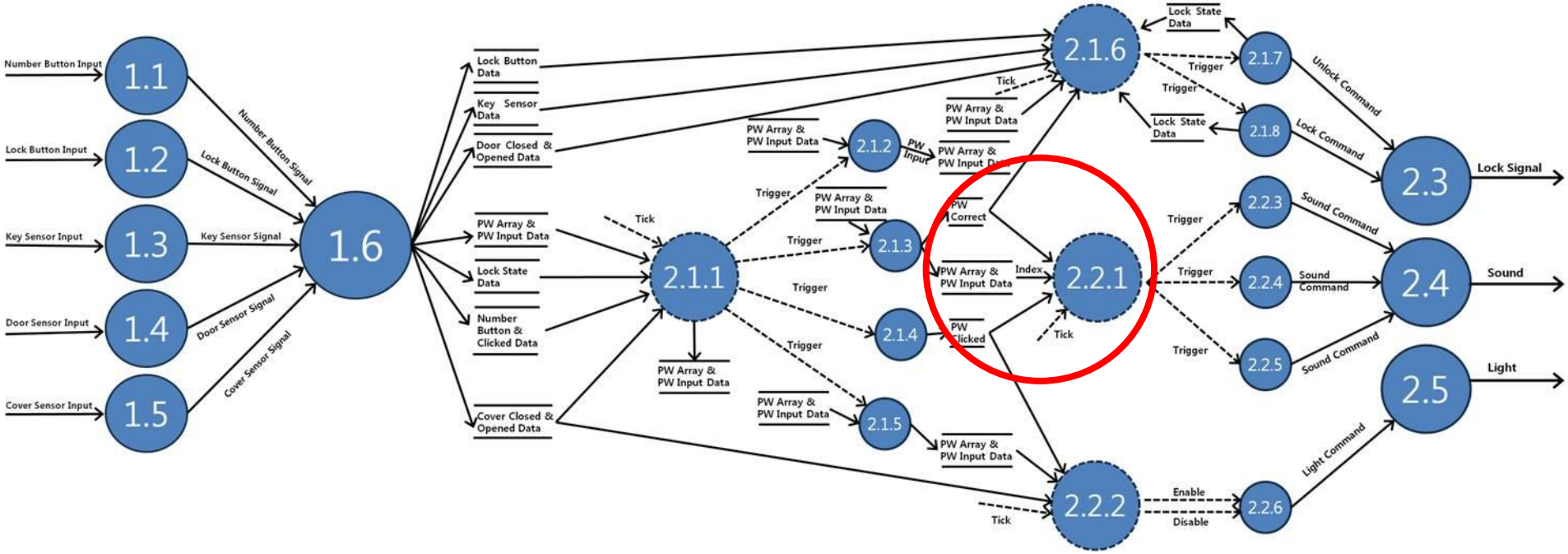
Modification

SRA 수정 사항 (1) DFD - Level 3



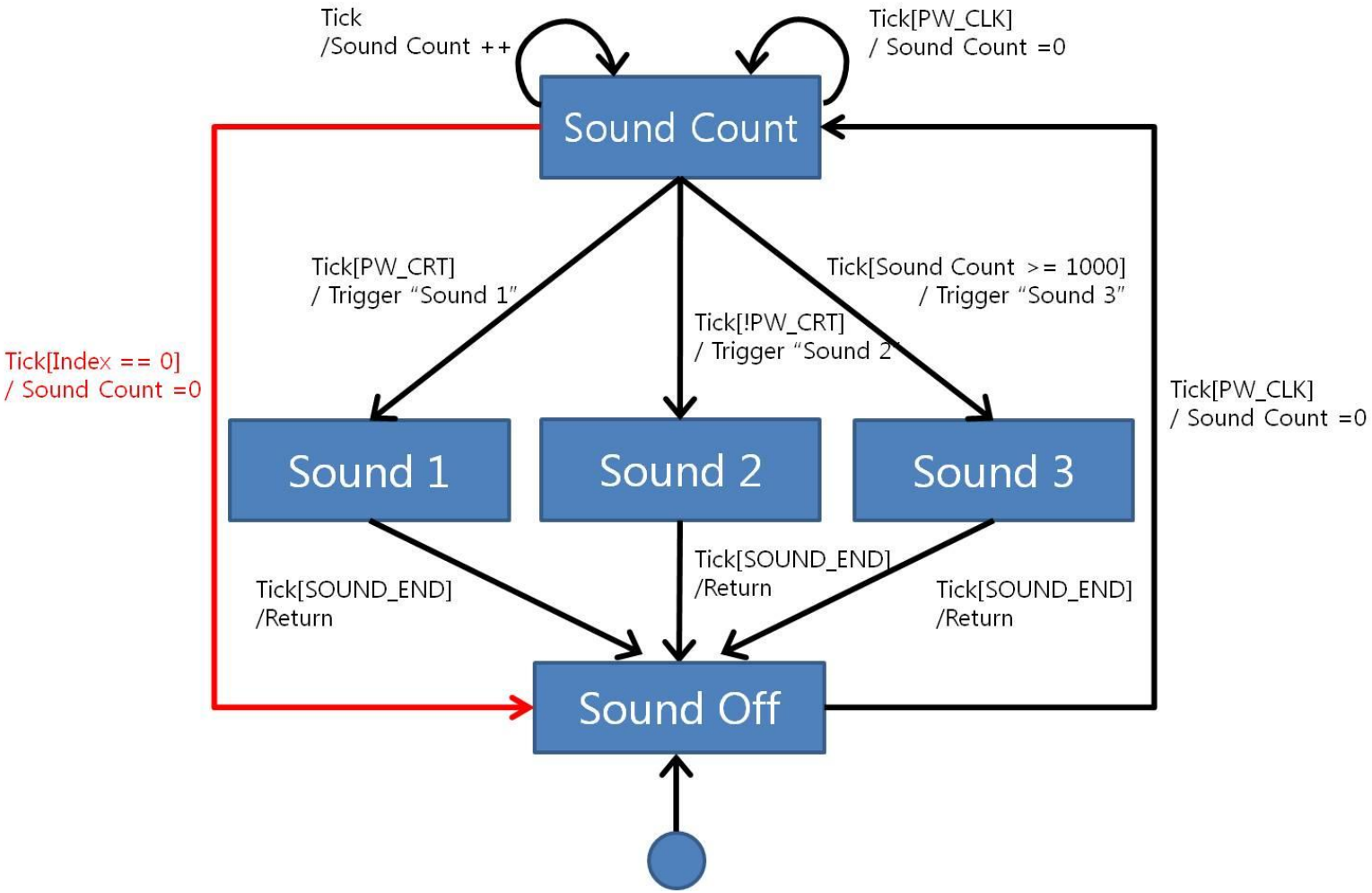
Modification

SRA 수정 사항 (1) DFD - overall



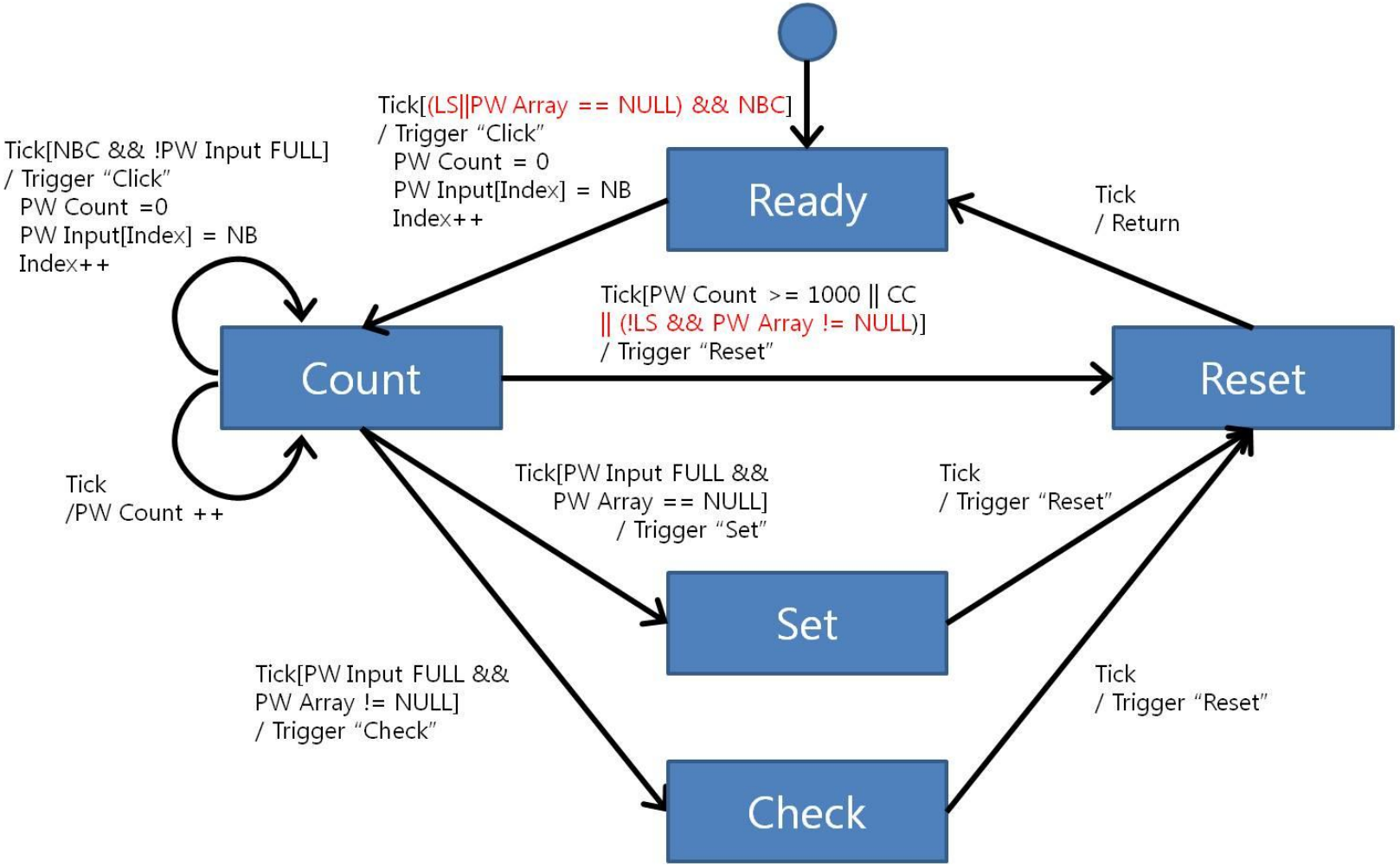
Modification

SRA 수정 사항 (2) STD – Sound Controller



Modification

SRA 수정 사항 (2) STD –PW Controller



Modification

SRA 수정 사항 (3) Process

Reference No.	2.2.1
Name	Sound Controller
Input	PW Correct Data, PW Clicked, Index, Tick
Output	Trigger
Process Description	It can be input Boolean type data. If PW Correct Data is input as TRUE, sends Trigger to Sound 1, as FALSE, sends Trigger to Sound 2. When PW Clicked is input as True, it initializes counting time and starts time counting. If counting time is 10 seconds or more it sends Trigger to Sound3. If the input data 'Index' is 0(When the PW Input is reset), it stops counting and initialize count.

Modification

SRA 수정 사항 (4) Data Dictionary

Input/output Event	Description	Format / Type
PW Array & PW Input Data	These data are included in PW State. PW Input Data has PW Input and Index. PW Input is made up four Number Button Data in incoming order and Index with a range of 0-4 is the size of PW Input. PW Array Data also saves integer array data used to checks PW Input Data. If PW Array saves NULL then Password hasn't set yet.	Structure
PW Correct Data	Using information generated by PW Check Process, it saves Boolean data whether the password is correct or not. It saves NIL as init state.	NIL /True / False, Interrupt
PW Clicked	The data generated by PW Clicked. It sends Boolean data. (It sends true when PW Click.)	True / False, Interrupt

Modification

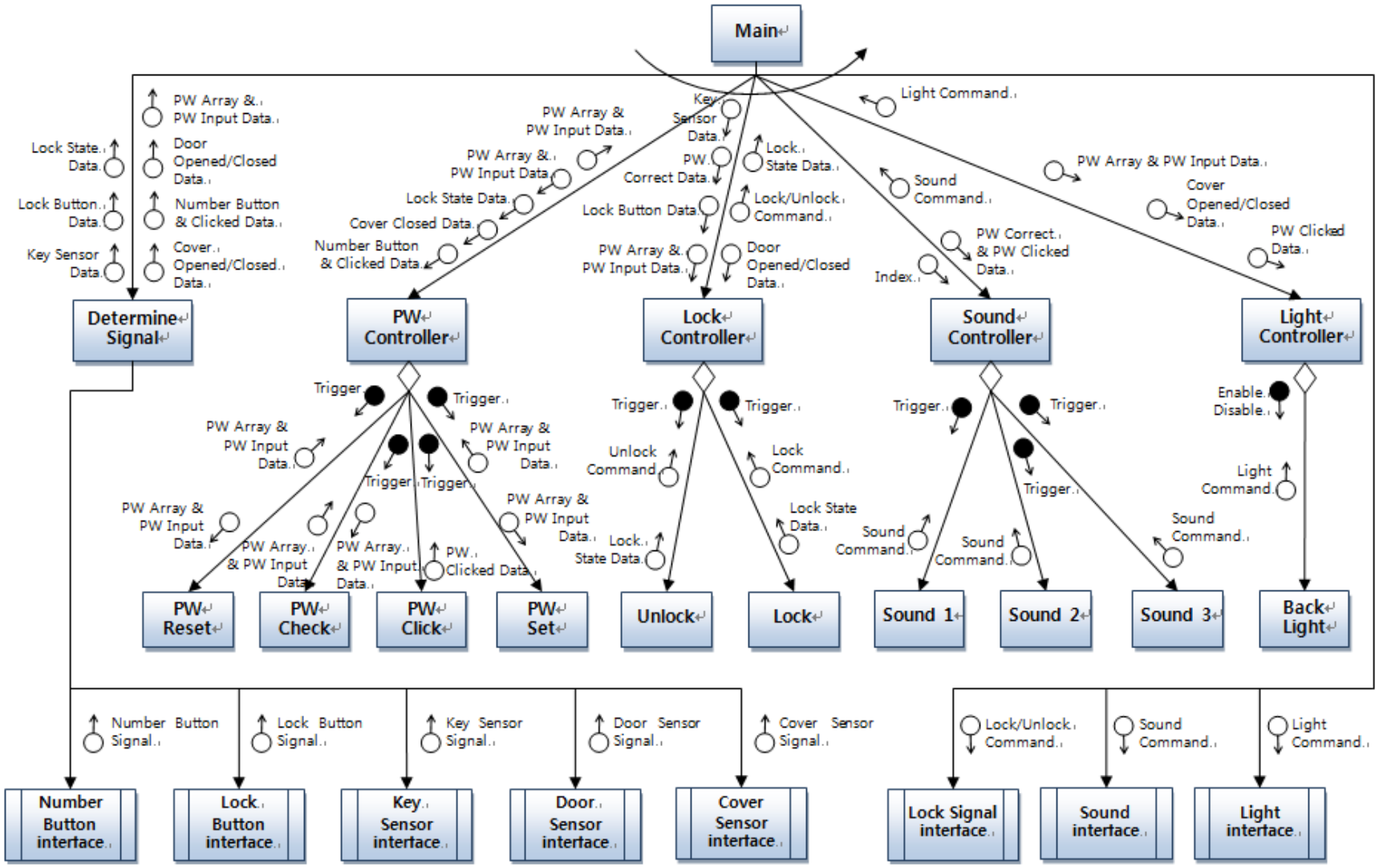
SRA 수정 사항 (5) UTP

- Determine signal (process 1.6)에 대한 test case 추가
  - 모든 경우의 수를 포함하기 어려워 일부만 선별
  - 우선순위 반영에 대한 부분을 test할 수 있도록 작성

- 수정된 dfd와 std에 따라 test case 수정

About Code

About Code


Main
Determine Signal
PW Controller
Lock Controller
Sound Controller
Light Controller
PW Reset
PW Check
PW Click
PW Set
Unlock
Lock
Sound 1
Sound 2
Sound 3
Back Light
Number Button interface
Lock Button interface
Key Sensor interface
Door Sensor interface
Cover Sensor interface
Lock Signal interface
Sound interface
Light interface
PW Array & PW Input Data
Lock State Data
Door Opened/Closed Data
Number Button & Clicked Data
Cover Opened/Closed Data
Key Sensor Data
Lock Button Data
Cover Closed Data
PW Correct Data
Lock/Unlock Command
Sound Command
Light Command
Index
PW Correct & PW Clicked Data
PW Clicked Data
Trigger
Unlock Command
Lock Command
Enable Disable
Number Button Signal
Lock Button Signal
Key Sensor Signal
Door Sensor Signal
Cover Sensor Signal

About Code

Data Storage → 구조체로 구현

```
//number button data. 숫자 버튼하고 숫자 버튼 클릭 여부 변수
typedef struct _NUM_BUTTON_DATA{
    int num_button;
    BOOLEAN num_ck;
}NUM_BUTTON_DATA;

//lock state하고 pw state 변수
typedef struct _STATE_DATA{
    BOOLEAN lockstate;  //잠금 상태
    struct _PW_DATA *pw_data;
}STATE_DATA;

//door와 cover의 opened&closed data
typedef struct _DOOR_AND_COVER_DATA{
    BOOLEAN door_opened_data;
    BOOLEAN door_closed_data;
    BOOLEAN cover_opened_data;
    BOOLEAN cover_closed_data;
}DAC_DATA;
```

```
//door cover data, key sensor data 저장
typedef struct _SENSOR_DATA{
    BOOLEAN keysensor;
    DAC_DATA dac_data;
}SENSOR_DATA;
```

About Code

전체 데이터 묶음 구조체

```
//determine에서 반환해줄 구조체
//크게 button, state, sensor data 저장
typedef struct _BUTTON_AND_SENSOR_AND_STATE_DATA{
    struct _NUM_BUTTON_DATA *numbtn_data;
    BOOLEAN lockbutton_data;
    struct _STATE_DATA *state_data;
    struct _SENSOR_DATA *sensor_data;
}BSS_DATA;
```

About Code

각 controller 들에 대한 구조체

```
//////////////PW control////////////////

#define PW_SIZE 4 //PW input size


//pw input & pw array
typedef struct _PW_DATA{
    int input[4];
    int *arr;
    int index;
}PWD;

typedef enum _PW_STATE{
    READY,  //0
    COUNT,  //1
    SET,    //2
    CHECK,  //3
    RESET   //4
}PW_STATE;

//////////////////Lock control////////////////

//Unlock과 Lock프로세스에서 상태와 커맨드 반환을 위한 구조체
//Controller에서 들어가는거랑 구분하기 위해서 '_'는 빼고 씀
typedef struct _LockCommand{
    BOOLEAN lock_command;
    BOOLEAN lockstate;
}LOCK_CMD;


typedef enum _LOCK_CONTROL_STATE{
    Unlock,
    Lock,
    Unlock_Counts//Unlock count
}LC_STATE;
```

```
////////////////Light Control//////////

typedef enum _LIGHT_CONTROL_STATE{
    Light_OFF,
    Light_ON
}LTC_STATE;

enum ABLE{
    Disable = 0,
    Enable = 1
};

///////////////Sound control//////////

typedef enum _SOUND_CONTROL_STATE{
    Sound_OFF,  //0
    Sound_1,    //1
    Sound_2,    //2
    Sound_3,    //3
    Sound_Count_Mod //4(변수명 sound_count와 이름이 겹쳐 Sound_Count_Mod로 이름 변경)
}SC_STATE;
```

About Code

함수의 proto type

```
/////////////determine///////
BSS_DATA * determine(BOOLEAN LB_signal, BOOLEAN DS_signal, BOOLEAN KS_signal, BOOLEAN CS_signal,int NB_signal,  BSS_DATA *bss_data);
int mainmodule( );
/////////////PW control////////////////
BOOLEAN pw_check(PWD * pwd);
int pw_set(PWD * pwd);
BOOLEAN pw_click(BOOLEAN *nbc);
void pw_reset(int *index);
int pw_controller(PW_STATE *pw_state, STATE_DATA *lock_and_pw_data, NUM_BUTTON_DATA *numb_data, BOOLEAN CC, BOOLEAN *pw_clk, BOOLEAN *pw_crt, int *

////////////////Lock control////////////////
LOCK_CMD *UnlockP(LOCK_CMD *Loc); //unlock Process
LOCK_CMD *LockP(LOCK_CMD *Loc);
LOCK_CMD * lock_controller(LC_STATE *State, int *Door_Count, STATE_DATA *state,SENSOR_DATA *sens_data, BOOLEAN LB, BOOLEAN pw_crt, LOCK_CMD *cmd);

////////////////Light Control/////////////
BOOLEAN light_controller(LTC_STATE *state, int *Light_Count,BOOLEAN CO,BOOLEAN CC, BOOLEAN pw_clk, PWD * pwd,BOOLEAN light_cmd);
BOOLEAN BackLight(int able, BOOLEAN light_cmd) ;
////////////////Sound control/////////////
int Sound1( );
int Sound2( );
int Sound3( );
int sound_controller(SC_STATE *state,int *Sound_Count,BOOLEAN pw_clk,BOOLEAN pw_crt, int index);
```

About Code

Determine Signal (Process 1.6)

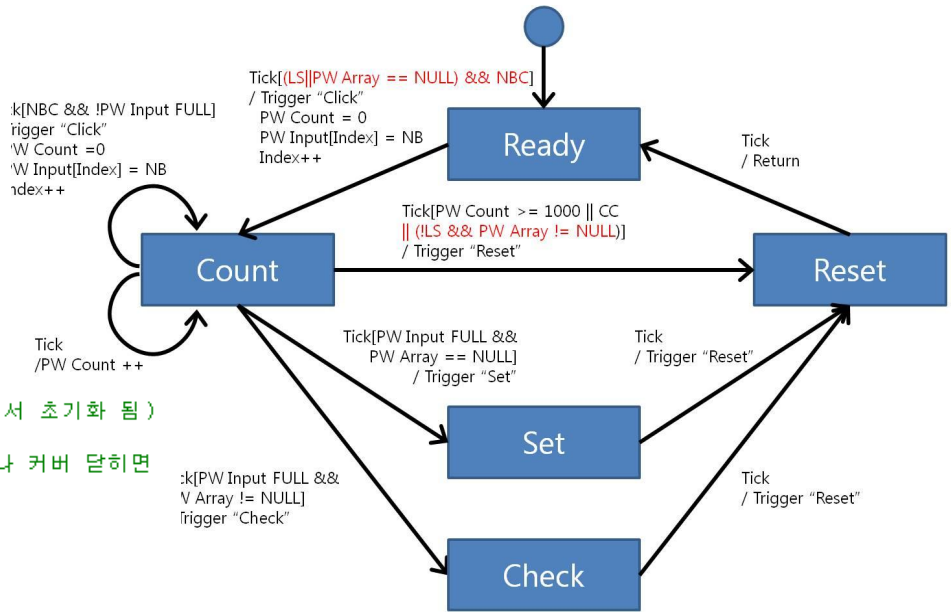
```
BSS_DATA * determine(BOOLEAN LB_signal, BOOLEAN DS_signal, BOOLEAN KS_signal, BOOLEAN CS_signal,int NB_signal,  BSS_DATA *bss_data) {
    /* ... */
    if (bss_data == NULL) {
        //단, NULL일때 딱 한번만 생성
        bss_data = (BSS_DATA*) malloc(sizeof(BSS_DATA));
        memset(bss_data, 0x0, sizeof(BSS_DATA));
        //그 내부 구조체 포인터들에도 메모리 할당을 해준 다음에(이것도 한번만)
        bss_data->numbtn_data = (NUM_BUTTON_DATA*) malloc(  sizeof(NUM_BUTTON_DATA));
        memset(bss_data->numbtn_data, 0x0, sizeof(NUM_BUTTON_DATA));
        bss_data->state_data = (STATE_DATA*) malloc(sizeof(STATE_DATA));
        memset(bss_data->state_data, 0x0, sizeof(STATE_DATA));
        bss_data->state_data->pw_data = (PWD*) malloc(sizeof(PWD));
        memset(bss_data->state_data->pw_data, 0x0, sizeof(PWD));
        bss_data->sensor_data = (SENSOR_DATA*) malloc(sizeof(SENSOR_DATA));
        memset(bss_data->sensor_data, 0x0, sizeof(SENSOR_DATA));
    }
    /* ... */

    if (LB_signal == TRUE) {
        //Lock button = TRUE
        bss_data->lockbutton_data = TRUE;
    } else if (!(DS_signal & 0x2)) {    //DS signal이 들어왔는가
        if (DS_signal == TRUE) {
            //door closed data = TRUE
            bss_data->sensor_data->dac_data.door_closed_data = TRUE;
        } else if (DS_signal == FALSE) {    //
            //door opened data = TRUE
            bss_data->sensor_data->dac_data.door_opened_data = TRUE;
        }
    } else if (KS_signal == TRUE) {
        //key sensor data = TRUE
        bss_data->sensor_data->keysensor = TRUE;
    } else if (!(CS_signal & 0x2)) {    //CS signal이 들어왔는지
        if (CS_signal == TRUE) {    //
            //          cover closed data = TRUE
            bss_data->sensor_data->dac_data.cover_closed_data = TRUE;
        } else if (CS_signal == FALSE) {    //
            //cover opened data = true
            bss_data->sensor_data->dac_data.cover_opened_data = TRUE;
        }
```

About Code

PW Controller (Process 2.1.1)

```
case READY :
    if((!sll*pwd->arr==NULL) && nbd->num_ck)
    {
        *pw_clk = pw_click(&nbd->num_ck);       //pw_clk값 반환받아오기(Trigger)
        *pw_count = 0;                          //count값 초기화
        pwd->input[pwd->index] = nbd->num_button;   //nb값 배열에 넣기
        (pwd->index)++;                         //index값 증가
        nbd->num_button = NIL;                      //nb값 초기화
        *pw_state = COUNT;                      //pw 상태 변화
    }
    break;
case COUNT :
    (*pw_count)++;                          //Tick 조건 : 무조건 증가
    if(nbd->num_ck && pwd->index != PW_SIZE)            //Count 조건
    {
        *pw_clk = pw_click(&nbd->num_ck);       //pw_clk 값 반환받아오기(Trigger)
        *pw_count = 0;                          //pw_count값 초기화
        pwd->input[pwd->index] = nbd->num_button;   //input에 nb 삽입
        (pwd->index)++;                         //index 증가
        nbd->num_button = NIL;                      //nb값 초기화(nbc는 pw_click에서 초기화 됨)
    }
    else if(*pw_count >= 1000 || CC || (!ls && pwd->arr != NULL))   //10초 지나거나 커버 닫히면
    {
        pw_reset(&pwd->index);
        *pw_state = RESET;
    }
    else if(pwd->index == PW_SIZE && pwd->arr == NULL)
    {
        pw_set(pwd);
        *pw_state = SET;
    }
    else if(pwd->index == PW_SIZE && pwd->arr != NULL)
    {
        *pw_crt = pw_check(pwd);
        *pw_state = CHECK;
    }
    break;
```

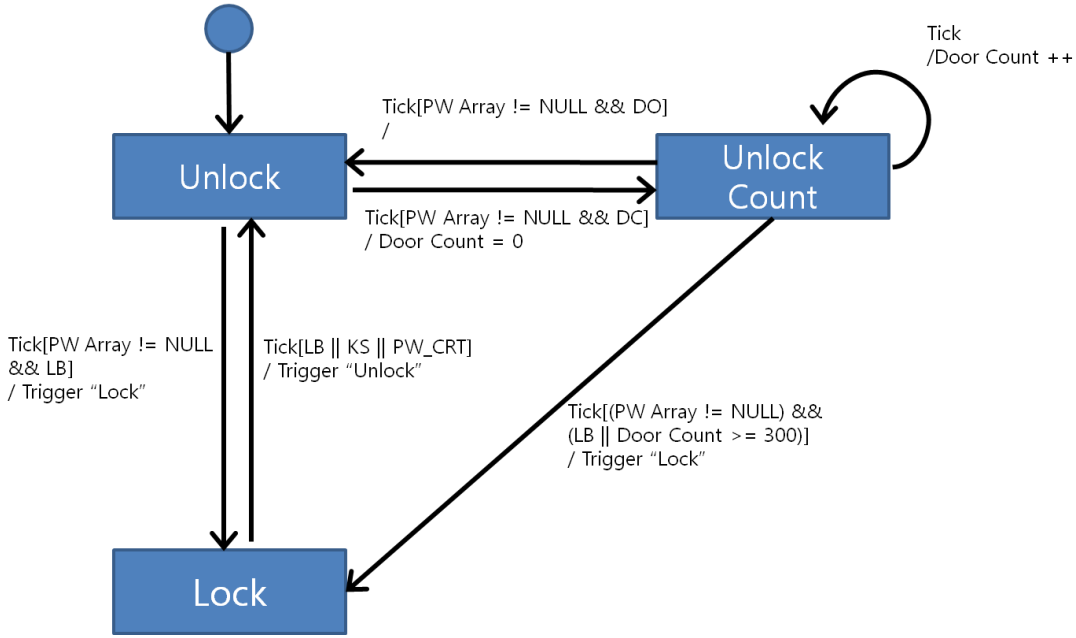


About Code

Lock Controller (Process 2.1.1)

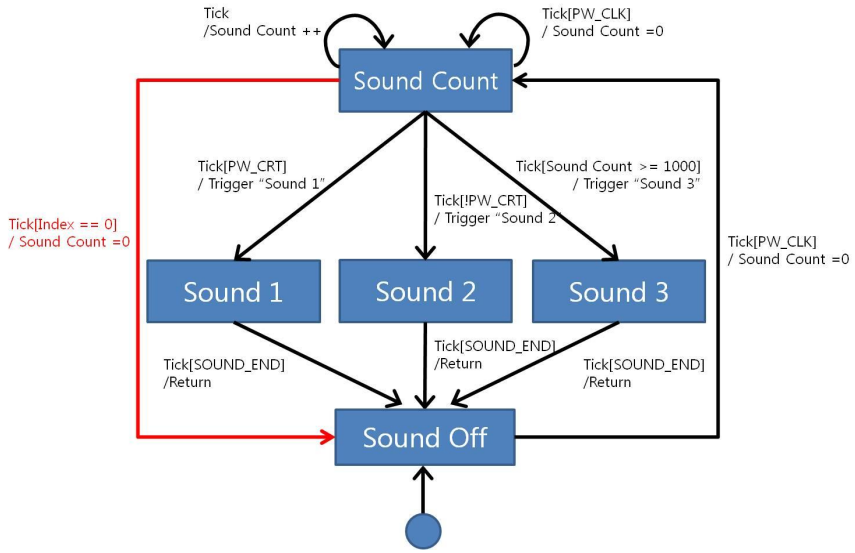
```
LOCK_CMD * lock_controller(LC_STATE *State, int *Door_Count, STATE_DATA *state,SENSOR_DATA *sens_data, BOOLEAN LB, BOOLEAN pw_crt, LOCK_CMD *cmd) {

    PWD *pwd = state->pw_data;
    switch (*State) {
        //언락상태
    case Unlock:
        //문이 닫혔을 경우
        if (pwd->arr != NULL && (sens_data->dac_data.door_closed_data == 1)) {
            *Door_Count = 0;
            *State = Unlock_Counts;
        }
        //lock button이 눌린 경우
        else if (pwd->arr != NULL && (LB == TRUE)) {
            *State = Lock;
            LockP(cmd);
        }
        break;
        //잠금상태
    case Lock:
        //락버튼이 눌리거나 열쇠를 입력했거나 비밀번호가 맞았거나
        if ((LB || sens_data->keysensor || (pw_crt==TRUE)) == TRUE) {
            *State = Unlock;
            UnlockP(cmd);
        }
        break;
        //문이 닫히고 안잠긴 상태
    case Unlock_Counts:
        //비밀번호 설정된 상태에서 문 열림
        if (pwd->arr != NULL && (sens_data->dac_data.door_opened_data == 1)) {
            *State = Unlock;
            //UnlockP(cmd);
            //비밀번호 설정된 상태에서 락버튼 눌리거나 3초이상 시간이 지남
        } else if (pwd->arr != NULL && ((LB == TRUE) || (*Door_Count >= 300))) {
            *State = Lock;
            LockP(cmd);
        } else{
            (*Door_Count)++;
        }
        break;
    }
    state->lockstate = cmd->lockstate;
    return cmd;
}
```



About Code

Sound Controller (Process 2.2.1)




```
if(pw_crt==TRUE){
    *state = Soun
    //          *Sound_Command=Sound1();    //Trigger 'Sound1'
    return Sound1();
}
else if(pw_crt==FALSE){
    *state = Sound_2;   //상태변화
    //          *Sound_Command=Sound2();    //Trigger 'Sound2'
    return Sound2();
}
else if(*Sound_Count >= 1000){
    *state = Sound_3;   //상태변화
    //          *Sound_Command=Sound3();    //Trigger 'Sound3'
    return Sound3();
}
else if(pw_clk==TRUE)
    *Sound_Count = 0;   //Sound Count 초기화
else if(index == 0)     //131121 3:38 추가 - reset 조건을 index로 검사 ! std와 dfd 변경 요망. (요망한것)
    *state = Sound_OFF;
break;
```

```
switch(*state){
case Sound_OFF :
    if(pw_clk == TRUE){
        *Sound_Count = 0;   //sound count 초기화
        *state=Sound_Count_Mod; //상태변화
    }
    break;
    /* ... */
case Sound_1 :
    *state = Sound_OFF;
    break;
case Sound_2 :
    *state = Sound_OFF;
    break;
case Sound_3 :
    *state = Sound_OFF;
    break;
case Sound_Count_Mod :
    (*Sound_Count)++;   //tick조건 count 증가
```

About Code

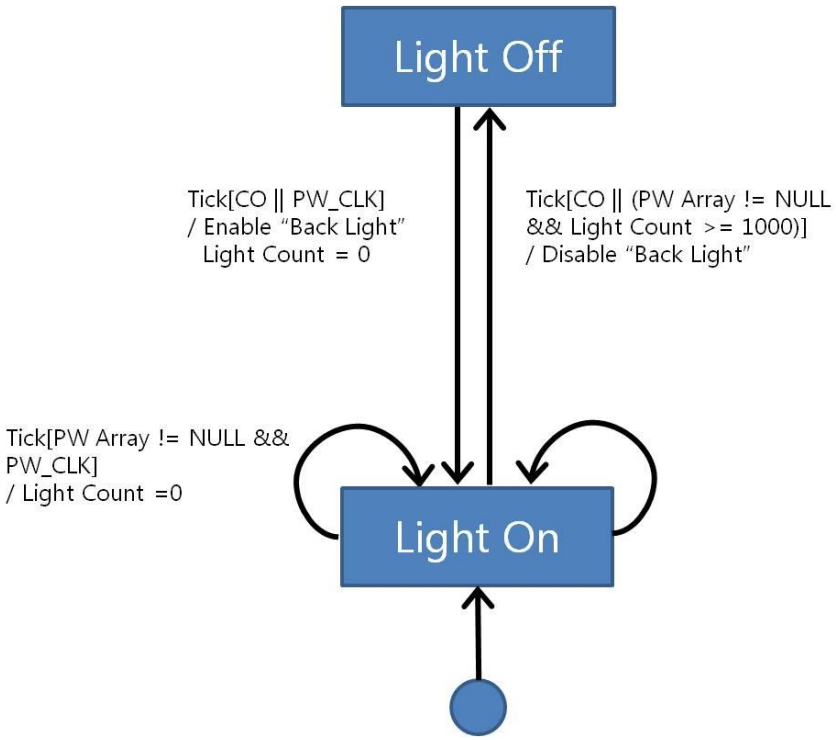
Light Controller (Process 2.2.1)

```
//input값은 구조체 형태에 따라 변경해야할 수 있음
BOOLEAN light_controller(LTC_STATE *state, int *Light_Count,BOOLEAN CO,BOOLEAN CC, BOOLEAN pw_clk, PWD * pwd,BOOLEAN light_cmd) {

    switch (*state) {
    case Light_OFF:
        if (CO || pw_clk==TRUE) {
            *state = Light_ON;
            (*Light_Count) = 0;
            light_cmd =  BackLight(Enable,light_cmd);
        }
        break;
    case Light_ON:
        (*Light_Count)++;

        if (CC || ((pwd->arr != NULL) && ((*Light_Count) >= 1000))){
            *state = Light_OFF;
            light_cmd = BackLight(Disable,light_cmd);
        }
        else if ((pwd->arr != NULL) && pw_clk==TRUE)
            *Light_Count = 0;

        break;
    }
    return light_cmd;
}
```



About Code

Main Module

```
/*
* determine 실행. 신호들 인자로 넘기고 출력으로 Button Data 구조체, Key Sensor Signal, State 구조체,
* Door Opened&Closed, Cover Opened&Closed 반환.
*/
bss_data = determine(LBS,DSS,KSS,CSS,NBS,bss_data);
/*
* 각 컨트롤 실행
*/
//1. PW Control
pw_controller(&pw_con_state,bss_data->state_data,bss_data->numbtn_data,bss_data->sensor_data->dac_data.cover_closed_data,
    &pw_clk,&pw_crt,&pw_count);

//2. Lock Control
lock_controller(&lock_con_state, &door_count, bss_data->state_data,bss_data->sensor_data, bss_data->lockbutton_data,
    pw_crt, &lock_cmd);

//3. Sound Control
sound_cmd = sound_controller(&sound_con_state,&sound_count,pw_clk,pw_crt,bss_data->state_data->pw_data->index);

//4. Light Control
light_cmd = light_controller(&light_con_state, &light_count, bss_data->sensor_data->dac_data.cover_opened_data,
    bss_data->sensor_data->dac_data.cover_closed_data,pw_clk, bss_data->state_data->pw_data,light_cmd);
num_pad_ui(NBS,CSS,light_cmd);


//그냥 가시적으로 보여주기 위한
showState(&lock_cmd,sound_cmd,bss_data->state_data->pw_data,CSS,DSS);
gotoxy(0,0);

//입력값들 및 pw_clk, pw_crt 초기화
NBS = NIL;
LBS = FALSE, KSS= FALSE;
DSS != 2, CSS != 2;
pw_crt = NIL;
pw_clk = FALSE;

//초기화 필요한 값들 초기화
init_bss(bss_data);
```

Demo

Feedback

Feedback

개발 과정

- SASD 개발 과정에 따라, waterfall 구조에 적합할 수 있도록, 최대한 앞서 진행한 결과에 충실하기 위해 노력
- 그러나 코드 작성 중 미쳐 생각지 못한 변수들 발견
- 이에 따라 앞선 단계들 수정
- 코드를 4명 모두가 나누어 작성한 뒤, 하나의 프로세스로 연결
- Std의 기능을 가능한 반영하여 작성하도록 약속 : 통일성
  → 작성된 문서에 대한 의존도가 매우 높음

Q&A

Thank You