SOFTWARE ENGINEERING

Team Presentation #4 (Impl.)

201114188 김종연
201114191 정재욱
201114192 정재철
201114195 홍호탁

Contents

목적

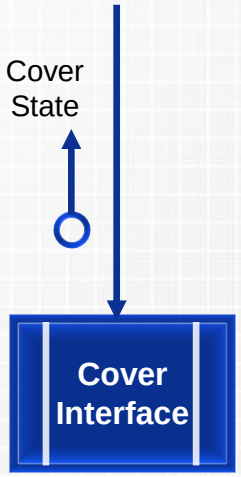
- 작성한 **SD**문서를 토대로 **EDLS** 프로그램 구현

소스 (Input)

```
int UserInput(char *Input){
    *Input = NULL;
    if (_kbhit()){
        *Input = _getch();
    }

    return TRUE;
}
```

- 소프트웨어상으로 구현을 위해 키보드 입력 값을 받아오는 부분 구현

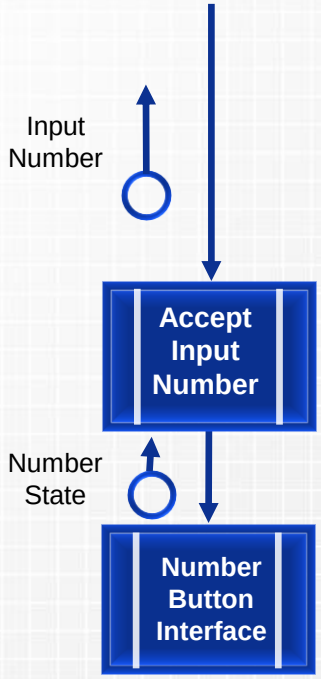
소스 (Cover Interface)




```
int CoverSensor(char Input, int *CoverState){
    static int CurrentState = FALSE;
    *CoverState = NONE;

    if (('c' == Input || Input == 'C')){
        if (CurrentState == TRUE){
            CurrentState = *CoverState = FALSE;
        }
        else{
            CurrentState = *CoverState = TRUE;
        }
    }

    return TRUE;
}
```

- 키보드 **C**버튼을 이용해 커버를 열고 닫는 기능을 구현
- **C**버튼이 입력되면 **Coverstate** 상태를 변경하여 리턴

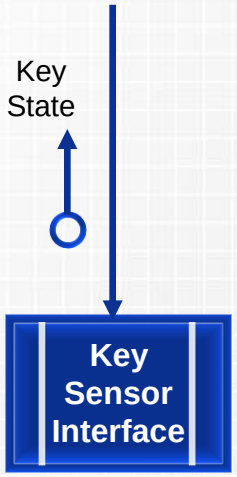
소스 (Number button Interface)




```
int NumButton(char Input, int *Number){
    *Number = NONE;
    if (('0' <= Input && Input <= '9')){
        *Number = Input - '0';
    }
    return TRUE;
}
```

❖ 키보드 숫자버튼이 입력되는지 확인하는 부분 구현

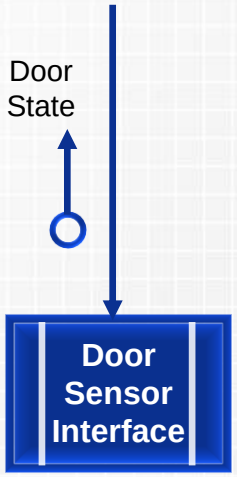
소스 (Key Sensor Interface)




```
int KeySensor(char Input, int *KeySensor){
    *KeySensor = FALSE;
    if (('k' == Input || Input == 'K')){
        *KeySensor = TRUE;
    }
    return TRUE;
}
```

- 키보드 **K**버튼을 이용해 키가 센서에 닿았는지를 감지하는 기능을 구현
- **K**버튼이 입력되면 **KeySensor** 상태를 변경하여 리턴

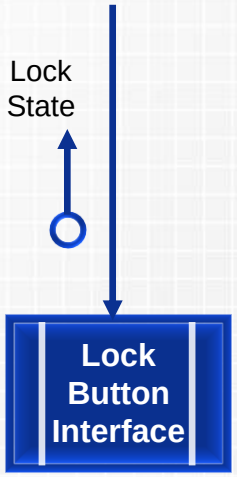
소스 (Door Sensor Interface)




```
int DoorSensor(char Input, int *DoorState){
    static int CurrentState = TRUE;

    if (('d' == Input || Input == 'D')){
        if (CurrentState == TRUE){
            CurrentState = *DoorState = FALSE;
        }
        else{
            CurrentState = *DoorState = TRUE;
        }
    }

    return TRUE;
}
```

- 키보드 **D**버튼을 이용해 문을 열고 닫는 기능을 구현
- **D**버튼이 입력되면 **Doorstate** 상태(열림,닫힘)를 변경하여 리턴

소스 (Lock Button Interface)




```
int LockButton(char Input, int *LockState){
    *LockState = FALSE;
    if (('l' == Input || Input == 'L')){
        *LockState = TRUE;
    }
    return TRUE;
}
```

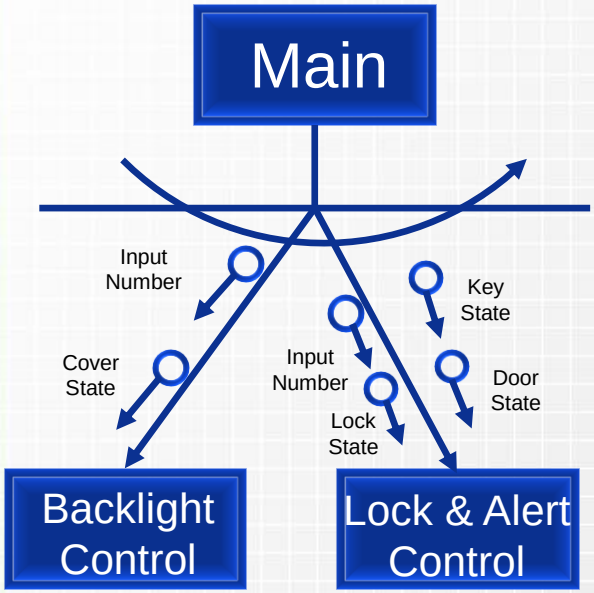
- 키보드 **L**버튼을 이용해 잠금장치를 풀고 잠그는 기능을 구현
- **L**버튼이 입력되면 **Lockstate** 상태**(**잠김**,**열림**)**를 변경하여 리턴

소스 (digital clock)

```c
#include "header.h"

int Tick(){
    static clock_t temp = 0;

    if ((clock() - temp) >= (1000 / TickPerSec)){
        temp = clock();
        return TRUE;
    }

    return FALSE;
}
```

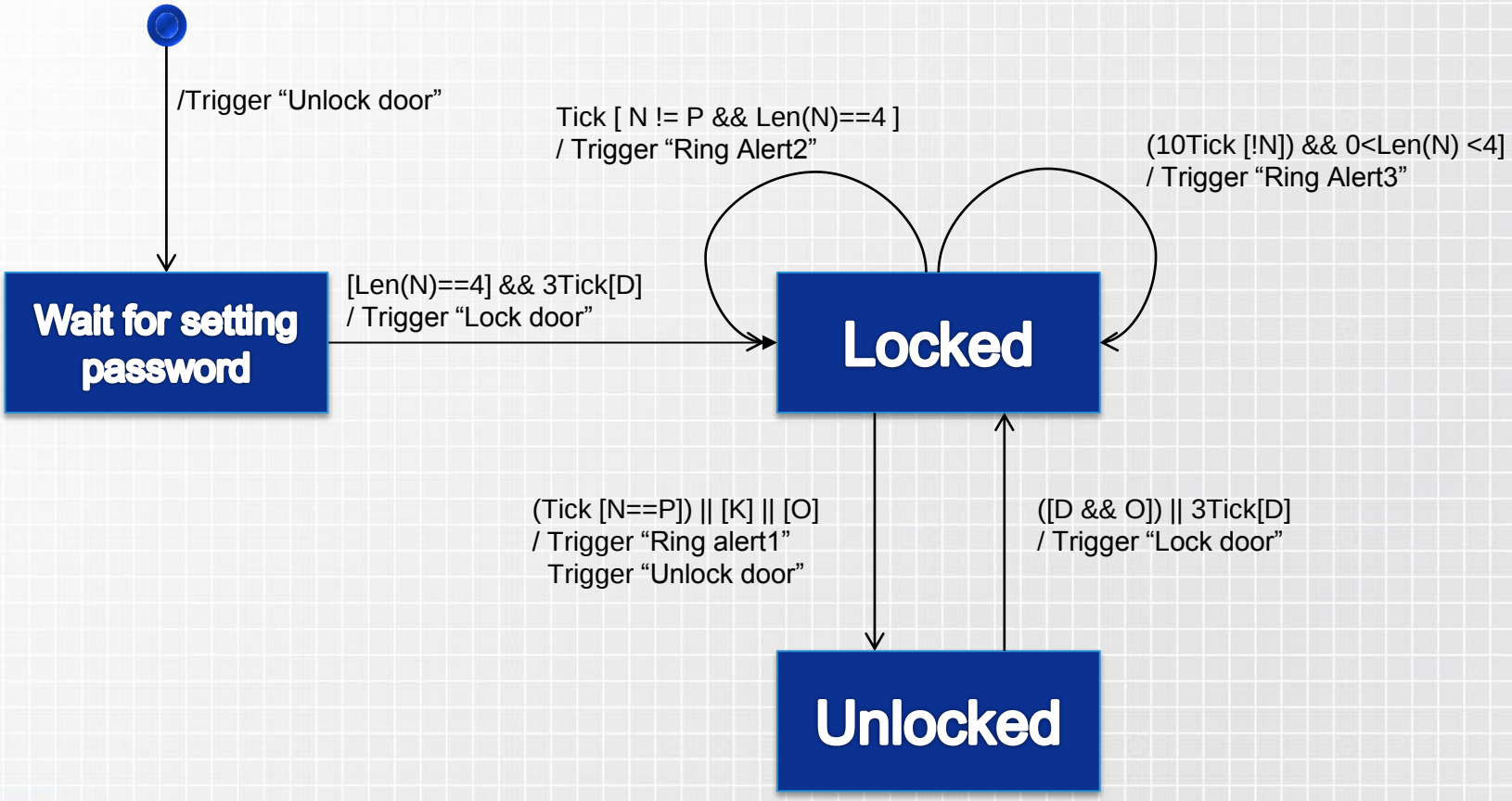
- 주기적으로 한번씩 **Tick**신호를 보내는 소스 구현

소스 (main)




```
int main(){
    char Input = '\0';
    int CoverState = NONE;
    int Number = NONE;
    int KeyState = NONE;
    int DoorState = NONE;
    int LockState = NONE;

    int tick;
    int Enable;
    int trigger = 0;

    CursorView(HIDE);   // ƒø⁰≠⁰°±,±,

    ////////////////////
    print_manual(5, 2);
    print_border(1, 1, 40, 12, 30);
    ////////////////////

    while (1){
        UserInput(&Input);

        NumButton(Input, &Number);          // 0~9
        CoverSensor(Input, &CoverState);    // C
        KeySensor(Input, &KeyState);        // K
        LockButton(Input, &LockState);      // L
        DoorSensor(Input, &DoorState);      // D

        tick = Tick();

        Enable = BacklightControl(Number, CoverState, tick);
        Event_Backlight(Enable);

        trigger = LockControl(Number, KeyState, LockState, DoorState, tick);
        Event_Lock(trigger);

        ////////////////////  √,∑¬ ≈◊Ω∫ΔÆ  ////////////////////
            PrintScreen(Number, CoverState, KeyState, LockState, DoorState, Enable, trigger, tick);
        ////////////////////////////////////////////////////////


    }

    return 0;
}
```

Lock control process



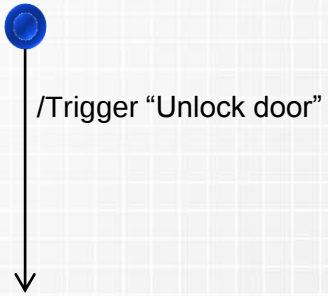
Lock control 전체소스 (1)

```
int LockControl(int Number, int KeyState, int LockState, int DoorState, int tick){
    static int TickCount = 0;
    static int password[4];
    static int inputNum[4];
    static int state = SYS_START;
    static int InNumCount = 0;
    int trigger = 0;
    int i;
    switch (state){
        case SYS_START:
            trigger |= UNLOCK_DOOR;
            state = WAIT_SET_PWD;
            break;
        case WAIT_SET_PWD:
            if (InNumCount < 4 && (0 <= Number && Number <= 9))
                password[InNumCount++] = Number;
            else if (InNumCount == 4){
                if (tick){
                    TickCount++;
                }
                if ((TickCount >= 3*TickPerSec) && DoorState){
                    InNumCount = 0;
                    TickCount = 0;
                    trigger |= LOCK_DOOR;
                    state = LOCKED;
                }
                if (TickCount >= 3*TickPerSec){
                    TickCount = 0;
                }
            }
            break;
```

Lock control 전체소스 (2)

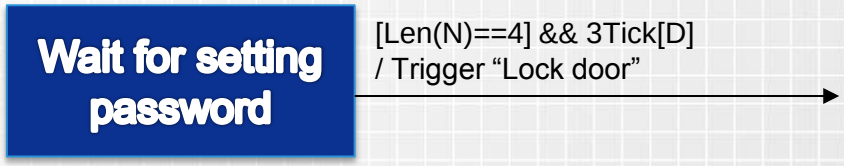
```
        case LOCKED:
            if (tick){
                TickCount++;
            }
            if (0 <= Number && Number <= 9){
                TickCount = 0;
                inputNum[InNumCount++] = Number;
                if (InNumCount == 4){
                    if (NumCmp(password, inputNum)){
                        trigger |= RING_ALERT_1;
                        trigger |= UNLOCK_DOOR;
                        state = UNLOCKED;
                    }
                    else{
                        trigger |= RING_ALERT_2;
                    }
                    for (i = 0; i < 4; i++){
                        inputNum[i] = NONE;
                    }
                    InNumCount = 0;
                }
            }
            else if (KeyState || LockState){
                TickCount = 0;
                trigger |= RING_ALERT_1;
                trigger |= UNLOCK_DOOR;
                state = UNLOCKED;
            }
            else if (TickCount >= 10*TickPerSec && 0 < InNumCount && InNumCount < 4){
                TickCount = 0;
                InNumCount = 0;
                trigger |= RING_ALERT_3;
            }
            break;
        case UNLOCKED:
            if (DoorState && LockState){
                TickCount = 0;
                trigger |= LOCK_DOOR;
                state = LOCKED;
            }
            if (DoorState){
                if (tick){
                    TickCount++;
                }
                if (TickCount >= 3 * TickPerSec){
                    TickCount = 0;
                    trigger |= LOCK_DOOR;
                    state = LOCKED;
                }
            }
            else{
                TickCount = 0;
            }
            if (TickCount > 3 * TickPerSec){
                TickCount = 0;
            }
            break;
    }
    return trigger;
}
```

상세 소스(system start)



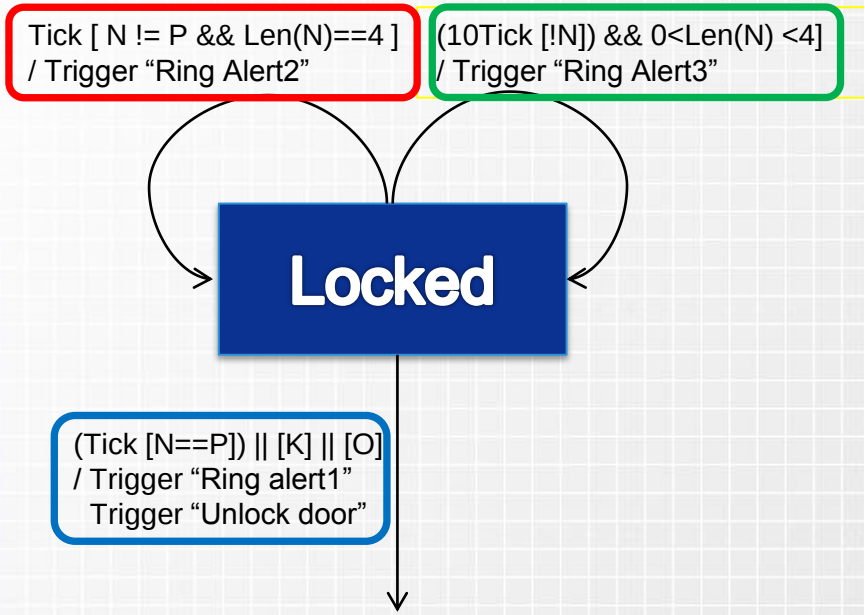

```
case SYS_START:
     trigger |= UNLOCK_DOOR;
     state = WAIT_SET_PWD;
     break;
```

상세 소스(Wait for setting password)



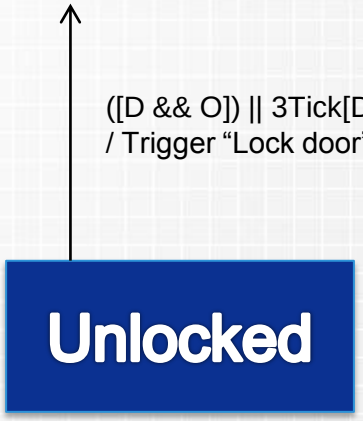

```
case WAIT_SET_PWD:
    if (InNumCount < 4 && (0 <= Number && Number <= 9))
        password[InNumCount++] = Number;
    else if (InNumCount == 4){
        if (tick){
            TickCount++;
        }
        if ((TickCount >= 3*TickPerSec) && DoorState){
            InNumCount = 0;
            TickCount = 0;
            trigger |= LOCK_DOOR;
            state = LOCKED;
        }
        if (TickCount >= 3*TickPerSec){
            TickCount = 0;
        }
    }
    break;
```

상세 소스(Locked state)



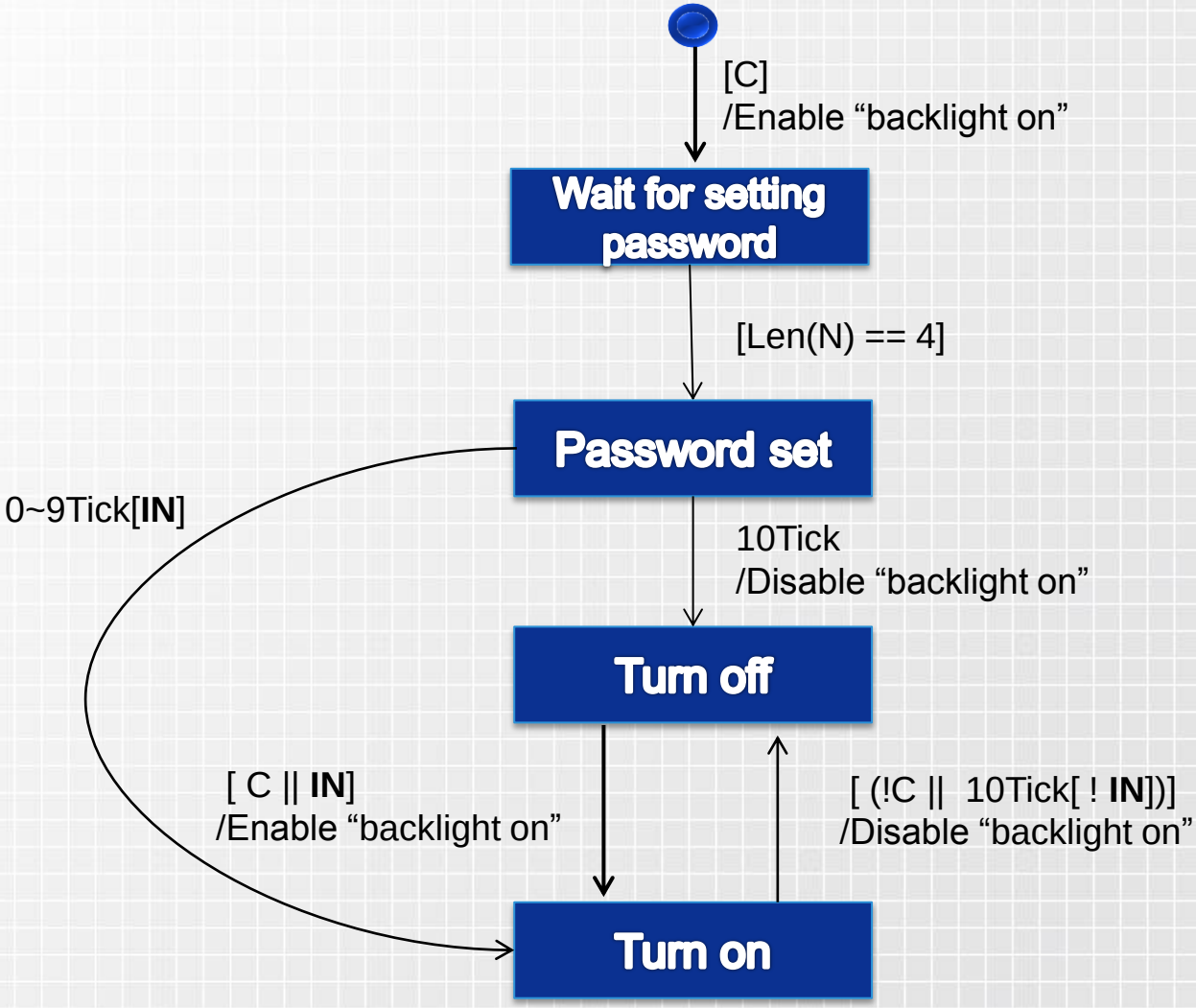

```
case LOCKED:
    if (tick){
        TickCount++;
    }
    if (0 <= Number && Number <= 9){
        TickCount = 0;
        inputNum[InNumCount++] = Number;
        if (InNumCount == 4){
            if (NumCmp(password, inputNum)){
                trigger |= RING_ALERT_1;
                trigger |= UNLOCK_DOOR;
                state = UNLOCKED;
            }
            else{
                trigger |= RING_ALERT_2;
            }
            for (i = 0; i < 4; i++){
                inputNum[i] = NONE;
            }
            InNumCount = 0;
        }
    }
    else if (KeyState || LockState){
        TickCount = 0;
        trigger |= RING_ALERT_1;
        trigger |= UNLOCK_DOOR;
        state = UNLOCKED;
    }
    else if (TickCount >= 10*TickPerSec && 0 < InNumCount && InNumCount < 4)
        {
        TickCount = 0;
        InNumCount = 0;
        trigger |= RING_ALERT_3;
    }
    break;
```

상세 소스(unlocked state)




```
case UNLOCKED:
    if (DoorState){
        if (tick) TickCount++;
        if (LockState || (TickCount >= 3 * TickPerSec)){
            TickCount = 0;
            trigger |= LOCK_DOOR;
            state = LOCKED;
        }
    }
    else TickCount = 0;
    break;
```

Back light control process



Back light control 전체 소스

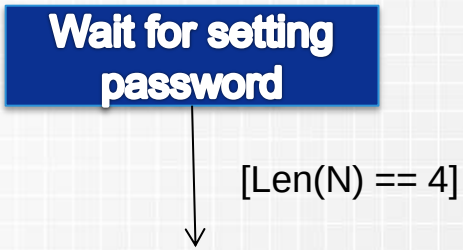
```
int BacklightControl(int Number, int CoverState, int tick){
    static int state = SYS_START;
    static int InNumCount = 0;
    static int TickCount = 0;

    switch (state){

        case SYS_START:
            if (CoverState == TRUE){
                state = WAIT_SET_PWD;
                return TRUE;
            }
            break;

        case WAIT_SET_PWD:
            if (0 <= Number && Number <= 9){
                InNumCount++;
            }
            if (InNumCount == 4){
                InNumCount = 0;
                state = PWD_SET;
            }
            break;

        case PWD_SET:
            if (tick){
                TickCount++;
            }
            if (TickCount < 10*TickPerSec && (0 <= Number && Number <= 9)){
                TickCount = 0;
                state = TURN_ON;
            }
            else if (TickCount >= 10*TickPerSec || (CoverState == FALSE)){
                TickCount = 0;
                state = TURN_OFF;
                return FALSE;
            }
            break;

        case TURN_OFF:
            if (CoverState == TRUE || (0 <= Number && Number <= 9)){
                state = TURN_ON;
                return TRUE;
            }
            return FALSE;

        case TURN_ON:
            if (tick){
                TickCount++;
            }
            if (0 <= Number && Number <= 9){
                TickCount = 0;
            }
            if (CoverState == FALSE || TickCount >= 10*TickPerSec){
                TickCount = 0;
                state = TURN_OFF;
                return FALSE;
            }
    }
    return NONE;
}
```

상세 소스(system start)

[C]
/Enable “backlight on”

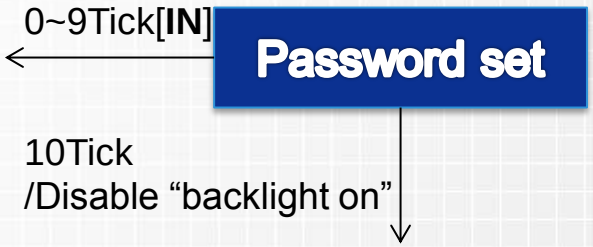
```
case SYS_START:
    if (CoverState == TRUE){
        state = WAIT_SET_PWD;
        return TRUE;
    }
    break;
```

상세 소스(Wait for setting password)



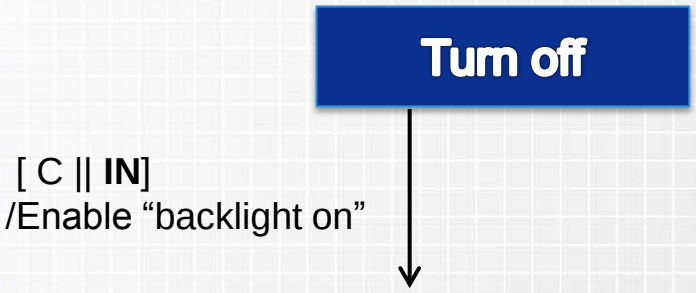

```
case WAIT_SET_PWD:
     if (0 <= Number && Number <= 9){
          InNumCount++;
     }
     if (InNumCount == 4){
          InNumCount = 0;
          state = PWD_SET;
     }
     break;
```

상세 소스(Password set)



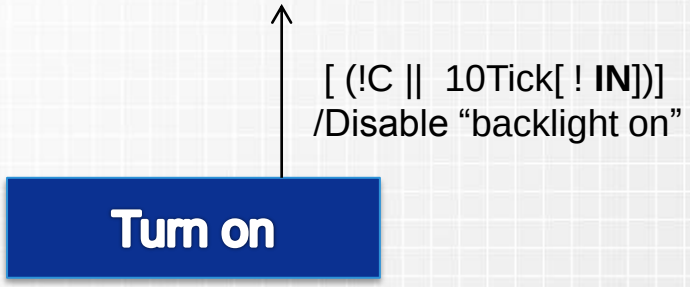

```
case PWD_SET:
    if (tick){
        TickCount++;
    }
    if (TickCount < 10*TickPerSec && (0 <= Number && Number <= 9)){
        TickCount = 0;
        state = TURN_ON;
    }
    else if (TickCount >= 10*TickPerSec || (CoverState == FALSE)){
        TickCount = 0;
        state = TURN_OFF;
        return FALSE;
    }
    break;
```

상세 소스(Turn off state)




```
case TURN_OFF:
    if (CoverState == TRUE || (0 <= Number && Number <= 9)){
        state = TURN_ON;
        return TRUE;
    }
    return FALSE;
```

상세 소스(Turn on state)




```
case TURN_ON:
    if (tick){
        TickCount++;
    }
    if (0 <= Number && Number <= 9){
        TickCount = 0;
    }
    if (CoverState == FALSE || TickCount >= 10*TickPerSec){
        TickCount = 0;
        state = TURN_OFF;
        return FALSE;
    }
}
```

시연

Q&A

Thank You !