User Interface Function

1. Main Setting 설정 가이드

- <1> Uart Interrupt 설정 [필수]

#include "WIFI_Config.h" 추가필요

```
void InterruptUartWifiTx ()           : Wi-Fi Module <-> MCU Uart TX 인터럽트 설정
```

```
    __interrupt static void r_uart1_interrupt_send(void)
    {
        InterruptUartWifiTx();
    }
```

```
void InterruptUartWifiRx (RXD1)       : Wi-Fi Module <-> MCU Uart RX 인터럽트 설정
```

```
    __interrupt static void r_uart1_interrupt_receive(void)
    {
        InterruptUartWifiRx(RXD1);
    }
```

```
void InterruptUartWifiError ()        : Uart Error Callback 설정
```

```
    static void r_uart1_callback_error(uint8_t err_type)
    {
        InterruptUartWifiError();
    }
```

- <2> 초기화 및 주기함수 설정 [필수] [Parameter : 1 / Return : X]

#include "WIFI_Config.h" 추가필요

```
WifiControlProcess(WIFI_TIME_INI)          : 초기화 (EEPROM 초기화 이전)
WifiControlProcess(WIFI_TIME_SET)          : 초기셋팅 (EEPROM 초기화 이후 While 이
전)
WifiControlProcess(WIFI_TIME_WHILE)        : Main While 문에서 호출
WifiControlProcess(WIFI_TIME_100MS)        : Main 100ms 타이머 에서 호출 (100ms 주
기)
```

- <3> Wifi Key 호출 [Parameter : 1 / Return : X]

#include "WIFI_Config.h" 추가필요

```
WifiKey(WIFI_KEY_AP)                       : Wifi Pairing 모드 진입 Key
WifiKey(WIFI_KEY_BLE)                      : BLE Pairing 모드 진입 Key
WifiKey(WIFI_KEY_ONOFF)                    : WIFI Power On/Off 설정 (Toggle)
WifiKey(WIFI_KEY_BLE_CERT)                 : BLE 점유인증 확인
WifiKey(WIFI_KEY_BLE_CERT_CANCEL)          : BLE 점유인증 취소
```

- <4> Wifi Send Data (Server DATA 전송) [Parameter : 1 / Return : X]

#include "WIFI_Config.h" 추가필요

```
[필수]
WifiSendData(WIFI_DATA_EXAMINE)            : 정밀 고장진단 완료 시 (이벤트)

[선택사항 - Protocol 설정 시 이벤트 방식 설정]
WifiSendData(WIFI_DATA_FUNCTION)           : 기능데이터 변경 시 (상태변경 시)
WifiSendData(WIFI_DATA_ERROR)              : 에러데이터 변경 시 (발생/해지)
WifiSendDataControl(WIFI_DATA_FUNCTION_CONTROL, U16 API ID) : 제품 KEY조작 시
```

- <5> 제품상태정보 설정 [Parameter : 2 / Return : X]

#include "WIFI_Config.h" 추가필요

```
(1) SetWifiSystemStatus(E_WIFI_USER_VALUE_T Status, U8 val)
```

```
WIFI_MODEL_TYPE                : 모델 타입 설정
    0 CLEAR                    : Model 0 (Model_0)
    1 SET                      : Model 1 (Model_1)

WIFI_FACTORY_STATUS            : Wi-Fi 검사모드 진입 설정
    0 CLEAR                    : Normal
    1 SET                      : Factory/PCB Test mode

WIFI_RX_STATUS                 : 일반제어 불가상태 설정
    0 CLEAR                    : Possible (Normal) (App Control O)
    1 SET                      : Impossible (Operating...) (App Control X)

WIFI_SMART_POSSIBLE_STATUS     : 스마트진단 불가상태 설정
    0 CLEAR                    : Possible (Normal)
    1 SET                      : Impossible (Operating...)

WIFI_SMART_SENSING_STATUS      : 스마트진단 중 상태 설정
    0 CLEAR                    : Normal
    1 SET                      : Operating

WIFI_FOTA_POSSIBLE_STATUS      : FOTA 불가 상태 설정
    0 CLEAR                    : Normal
    1 SET                      : Impossible (Operating...)

WIFI_FACTORY_DIRECTFOTA_STATUS : 다이렉트 포타 진입 설정
    0 CLEAR                    : Normal
    1 SET                      : Direct FOTA mode
```

---

2. WiFi 상태 정보 DATA (Get Function) 설정 가이드

- <1> Wi-Fi Module 상태 DATA [Parameter : 1 / Return : 1]

#include "WIFI_Config.h" 추가필요

```
(1) GetWifiStatusValue(WIFI_STATUS_DISP) : WIFI 상태 DISPLAY DATA (Enum 참고:
E_WIFI_DISPLAY_T)
```

Return DATA : Wi-Fi OFF [소등]

```
WIFI_DISP_OFF
```

Return DATA : Soft AP Pairing MODE [페어링점멸]

```
WIFI_DISP_AP_START    // Soft AP MODE START     : WiFi 페어링 모드 진입
WIFI_DISP_AP_STEP0    // Pairing Mode ing       : Soft AP 페어링 모드 활성화 완료
WIFI_DISP_AP_STEP1    // APP Pairing START      : APP으로부터 페어링 DATA수신 (25%)
WIFI_DISP_AP_STEP2    // AP Connect             : AP연결완료 (50%)
WIFI_DISP_AP_STEP3    // AUTHENTICATE           : 인증서버 확인 중 (75&)
```

Return DATA : BLE Pairing MODE [페어링점멸]

```
WIFI_DISP_BLE_START   // BLE MODE START         : BLE 페어링 모드 진입
WIFI_DISP_BLE_STEP0   // Pairing Mode ing       : Soft AP 페어링 모드 활성화 완료
WIFI_DISP_BLE_STEP1   // APP Pairing START      : APP으로부터 페어링 DATA수신 (25%)
WIFI_DISP_BLE_STEP2   // AP Connect             : AP연결완료 (50%)
WIFI_DISP_BLE_STEP3   // AUTHENTICATE           : 인증서버 확인 중 (75%)
```

Return DATA : Server 연결완료 [점등]

```
WIFI_DISP_CONNECT     // Server Connected       : 서버 연결완료 (100%)
```

Return DATA : Wi-Fi ON & 서버 연결끊김 [점멸]

```
WIFI_DISP_AP1         : 공유기 암호 불일치
WIFI_DISP_AP2         : 공유기 미검색
WIFI_DISP_AP3         : 공유기 접속 거부
WIFI_DISP_AP4         : 공유기 응답 없음
WIFI_DISP_AP5         : 서버 응답없음
WIFI_DISP_TRY         : 연결 시도 중
```

(2) GetWifiStatusValue(WIFI_STATUS_TEST) : TestMode 에서 연결확인

Return DATA

```
0: Disconnected
1: Connected (검사성공/완료)
```

(3) GetWifiStatusValue(WIFI_STATUS_ERROR) : Wifi Error (이전 연결 Error 이력)

Return DATA

```
0: Normal
1~5: WiFi Err No.
```

(4) GetWifiStatusValue(WIFI_STATUS_AP_ERROR) : Wifi Error (페어링중 Error)

Return DATA

```
0:Normal
1~5: 페어링 중 Err No.
```

(5) GetWifiStatusValue(WIFI_STATUS_BLE_CERT) : 점유인증 상태

Return DATA

```
0: 점유인증불가상태
1: 점유인증가능 상태
2: 점유인증완료상태
```

(6) GetWifiStatusValue(WIFI_STATUS_SERVER_CON) : 서버 연결 중 상태

Return DATA

```
0: 연결끊김 중
1: 연결중
```

(7) GetWifiStatusValue(WIFI_STATUS_PAIRING) : 페어링 모드 상태

Return DATA

```
0: 페어링 모드 비활성화 상태
1: BLE 활성화 중 상태
2: Wi-Fi 연결모드 활성화 중 상태
```

(8) GetWifiStatusValue(WIFI_STATUS_LOCATION) : APP 위치정보 설정상태 (Iocare2.0)

Return DATA

```
    0: APP에서 위치정보 설정 X
    1: APP에서 위치정보 설정 완료
```

(9) GetWifiStatusValue(WIFI_STATUS_SAP_CODE) : SAP 부가정보 DATA 수신여부 (A2021 SAP DATA)

Return DATA

```
    0: SAP DATA 가져오기 실패
    1: SAP DATA 수신성공 (S1000)
    2: SAP DATA 없음 (E9000)
```

(10) GetWifiStatusValue(WIFI_STATUS_FOTA) : MCU FOTA 상태

Return DATA

```
    0: FOTA 상태 아님
    1: FOTA 중 상태 (FOTA 시작)
```

- <2> Wifi Infomation data

#include "WIFI_Config.h" 추가필요

```
(1) GetWifiRequestValue(WIFI_RQST_WEATHER)          : 날씨데이터
```

Return DATA

```
    1: 맑음 / 2: 구름조금 / 3: 구름많음 / 4: 흐림 / 5: 흐리고 비 / 6: 흐리고 눈 / 7: 비온뒤 갬 / 8: 소나기 / 9: 비 또는 눈 / 11: 천둥번개 / 12: 안개
```

```
(2) GetWifiRequestValue(WIFI_RQST_AQI) : AQI (1 ~ 4)
```

Return DATA

```
    1: 좋음 / 2: 보통 / 3: 나쁨 / 4: 매우나쁨
```

```
(3) GetWifiRequestValue(WIFI_RQST_CURRENT_TEMP)     : 현재온도
```

Return DATA

```
    -50 ~ 50
```

```
(4) GetWifiRequestValue(WIFI_RQST_HIGH_TEMP)        : 최고온도
```

Return DATA

```
    -50 ~ 50
```

```
(5) GetWifiRequestValue(WIFI_RQST_LOW_TEMP)         : 최저온도
```

Return DATA

```
    -50 ~ 50
```

```
(6) GetWifiRequestValue(WIFI_RQST_FILTER_D_DAY1)    : 자가관리 / 방문관리 구분
```

Return DATA

```
0xFFFF : 데이터가 없는 경우(자가) / 9999 : 방문관리
```

```
(7) GetWifiRequestValue(WIFI_RQST_FILTER_CHANGE_CYCLE) : 필터교환 주기(%) (Ex:
15Month(기본) X 100%)
```

Return DATA

```
100: 100%(기본), 1~999 %
```

```
(8) GetWifiRequestValue(WIFI_RQST_PART_PERIOD)      : 상시진단 주기 적용
```

Return DATA

```
1: 1시간(기본), 1~720
```

- <3> SAP 부가정보 DATA 요청 [Parameter : X / Return : X]

#include "WIFI_Config.h" 추가필요

```
GetWifiRequestSAP(void);                              : SAP 부가정보 DATA 요청 (A2021
Json DATA 요청)
```

3. [WIFI_Config.h] / [WIFI_Config.c] Settings : 환경 설정 가이드

- <1> [WIFI_Config.h]

(1) User TEST MODE 설정

```
- TEST_MODE_WIFI_OTA    :  OTA 감도 테스트 진행 시 활성화 (WiFi OFF/RESET X )
    0: Normal / 1: OTA Test Mode
- TEST_MODE_BLE_FCC     :  BLE FCC 인증용 테스트 진행 시 활성화 (BLE OFF X)
    0: Normal / 1: BLE FCC Test Mode
```

(2) Version / Model Name / POP Code 설정

```
- USER_WIFI_VERSION     :   MCU Version (Ex: 0.0.0.1)
- MODEL_NAME_0          :   Main Model Name
- MODEL_NAME_1          :   Sub Model Name (CHP/CP 등 공용화 설계 시 활용)
- PRODUCT_CODE_0        :   Main Model POP Code
- PRODUCT_CODE_1        :   Sub Model POP Code (CHP/CP 등 공용화 설계 시 활용)
- WIFI_REQUEST_ID       :   MCU -> SERVER DATA 전송 시 Reqeust ID prefix
```

(3) H/W Port 설정

```
- TXD1, RXD1 사용       :   Uart Baudrate (115200bps)
- P_WIFI_ON             :   Wi-Fi Module Power Pin No.
- P_WIFI_RSTN           :   Wi-Fi Module Reset Pin No.
```

(4) FOTA 설정

```
- USER_FOTA_ENABLE      :   MCU FOTA 사용 유/무 설정
    0: DISABLE   /   1: ENABLE
- USER_FOTA_SIZE        :   MCU ROM SIZE 설정
```

(5) 부가정보 DATA 설정

```
- USER_USE_A2010_INFO           : A2010 / A2011 사용 유/무 설정
    0: DISABLE   /   1: ENABLE
- USER_USE_A2010_INFO_WEATHER   : 날씨정보 사용 유/무 설정
    0: DISABLE   /   1: ENABLE
- USER_USE_A2021_SAPINFO        : SAP정보(JSON) 사용 유/무 설정
    0: DISABLE   /   1: ENABLE
```

(6) User 프로토콜 DATA 사이즈 설정

```
- USER_DATA_MAX_SIZE                  : A1010 상태데이터 프로토콜의 최대 갯수 설정
- USER_DATA_MULTI_CONTROL_MAX_SIZE    : A1011 멀티제어 프로토콜의 최대 갯수
- USER_INFO_MAX_SZIE                  : A2010 부가정보 프로토콜의 최대 갯수
- USER_DIAGNOSIS_MAX_SIZE             : A1091 스마트진단(제품자체진단) 프로토콜 최대
갯수
```

(7) A2021 SAP 부가정보 데이터의 구조체 설정

```
- USER_JSON_MAX_ITEMS           :   [Key : Value] 쌍값의 최대 갯수  -> { Key1 :
Val1, Key2 : Val2, ... Key5 : Val5 }
- USER_JSON_MAX_ARRAY_VALUE     :   Array 의 최대 갯수    ->  [ 1 ], [ 2 ]
- USER_JSON_MAX_ARRAY_OBJECT    :   Array 의 Object 최대 갯수   -> [ {1}, {2} ]
```

- <2> [WIFI_Config.c]

(1) EEPROM Write / Read 설정

```
- EEPROM_ADDR_WIFI_POWER    : Wi-Fi ON/OFF 상태 기억
- EEPROM_ADDR_FIRST_PARING  : Wi-Fi 최초 페어링 상태 기억
- EEPROM_ADDR_FOTA_MODULE   : Wi-Fi Module FOTA 상태 기억
- EEPROM_ADDR_FOTA_MCU      : MCU FOTA 상태 기억
```

Write 함수 작성

```
Wi-Fi 관련 EEPROM 사용을 위한 EEPROM Write 함수 작성
```

```
    void UserWriteEep ( U16 mu16Addr, U8 mu8Val )
    {
        /* Start user code.*/

        /* End user code. */
    }
```

Read 함수 작성

```
Wi-Fi 관련 EEPROM 사용을 위한 EEPROM Read 함수 작성
```

```
U8 UserReadEep ( U16 mu16Addr )
{
    /* Start user code.*/

    /* End user code. */
}
```

(2) Wi-Fi Buzz / Sound 설정

```
- WIFI_BUZZER_SELECT            : 페어링 진행 음 (선택음 "띵")
- WIFI_BUZZER_AP_CONNECT        : AP연결음 (진입음 "띠리링" / Wi-Fi 연결모드가 활성화 되었습니다.)
- WIFI_BUZZER_BLE_CONNECT       : AP연결 (진입음 "띠리링" / BLE연결모드가 활성화 되었습니다.)
- WIFI_BUZZER_SETUP             : WIFI ON (설정음 "띵동↗" / 제품의 Wi-Fi가 켜졌습니다.)
- WIFI_BUZZER_SERVER_CONNECT    : 서버연결 (연결음 "띠리링↗" / 제품과 앱이 성공적으로 연결되었습니다.)
- WIFI_BUZZER_CANCEL            : WIFI OFF (해제음 "동띵" / 제품의 Wi-Fi가 꺼졌습니다.)
- WIFI_BUZZER_AP_START          : AP시작음 (설정음 "띵동 ↗")
- WIFI_BUZZER_ERROR             : 불가음 (에러음 "띵깡띵깡")
- WIFI_BUZZER_WiFi_OFF          : Wi-Fi 페어링 해제음(동띵) / Wi-Fi 연결 모드가 비활성화 되었습니다.
- WIFI_BUZZER_BLE_OFF           : 블루투스 페어링 해제음(동띵) / 블루투스 연결 모드가 비활성화 되었습니다.
- WIFI_BUZZER_BLE_CERT          : 점유인증완룡 음(선택음 "띵") / 제품인증을 진행 중입니다. 앱에서 가이드하는 버튼을 눌러주세요.
- WIFI_BUZZER_SERVER_FIRST      : 페어링 완료 후 서버연결 (연결음 "띠리링↗" / 제품과 앱이 성공적으로 연결되었습니다.)
- WIFI_BUZZER_SMART_CHECK_OK    : 설정음 ("띵동 ↗") / 스마트 진단이 완료 되어 제품을 사용하실 수 있습니다.
- WIFI_BUZZER_SMART_CHECK_ERROR : 불가음 (에러음 "띵깡띵깡") / 스마트 진단이 완료 되었습니다.
```

Buzz / Sound 출력 함수 작성

```
Wi-Fi관련 부저/음성 사운드 출력을 위한 코드 작성
```

```
void UserBuzzSound ( U16 mu16Buzz )
{
    switch( mu16Buzz )
```

```
        {
        case WIFI_BUZZER_SELECT :
        /* Start user code.*/

        /* End user code. */
        break;
        ...
        }
    }
```

(3) 서버 시간 설정

```
Ex)
- mYear     : 2024 (2024년)
- mMonth    : 12 (12월)
- mDate     : 25 (25일)
- mHour     : 23 (오후 11시)
- mMin      : 59 (59분)
- mSec      : 59 (59초)
- mDay      : 3 (수요일) (1:월 ~ 7:일)
```

서버 시간 설정 함수 작성

```
서버로부터 시계데이터 수신 시 마다 아래 함수 호출 (이벤트)
```

```
    void SetUserSystemTime ( U16 mYear, U16 mMonth, U16 mDate, U16 mHour, U16
mMin, U16 mSec, U16 mDay )
    {
        /* Start user code.*/

        /* End user code. */
    }
```

4. Protocol List Settings : 프로토콜 설정 가이드

폴더: WifiServerProtocolList/

- <1> WIFI_A101x_FunctionList.c (상태 데이터)

(1) 프로토콜 LIST 작성

```
- API ID            : 프로토콜의 API ID
```

```
- DATA TYPE         : A1014 사용여부 / 스마트진단 프로토콜 사용여부 선택 (이외는 미사용)
    -> TYPE_UNUSED                  : DATA TYPE 미사용 (Normal)
    -> TYPE_A1014                   : A1014 사용선택 (제품에서 상태 설정 시 DATA전송)
    -> TYPE_SMART_DIAGNOSIS_CONTROL : 스마트진단 시작/정지 프로토콜 ID, 진단실행 중 예외처리를 위한 DATA TYPE 설정
```

```
- Initial VALUE     : Event DATA 처리를 위한 초기값 (수정 필요 없음. [0]고정)
```

```
- Event Form        : 해당 프로토콜 ID의 Event 처리를 위한 Evnet 유형 설정 (1초마다 Event Check 진행)
    -> EVENT_UNUSED         : Event 처리 없음
    -> EVENT_STATUS_CHANGE  : 해당 ID의 DATA 변경 시 Event DATA 전송
    -> EVENT_ZERO_TO_ONE    : 해당 ID의 DATA가 "0"에서 "1"로 변경 시만 Event DATA 전송
    -> EVENT_ZERO_TO_OTHER  : 해당 ID의 DATA가 "0"에서 "?"(0이 아닌값)으로 변경 시 Event DATA 전송
[Event처리가 상기 유형(Event Form)에 해당되지 않는 이벤트 처리의 경우 [EVENT_UNUSED]로 설정 후 Main 프로그램에서 "WifiSendData(WIFI_DATA_FUNCTION)" 호출 하여 Event 처리]
```

```
// Ex)
WifiTxFuncEventList_T WifiFuncEventList[] =
{   /*  API ID        DATA_TYPE                       (Initial val)
EventForm     */
    {   0x0001,       TYPE_A1014,                           0,
EVENT_STATUS_CHANGE },
    {   0x0002,       TYPE_A1014,                           0,
EVENT_STATUS_CHANGE },
    {   0x0003,       TYPE_A1014,                           0,
EVENT_STATUS_CHANGE },
    {   0x0004,       TYPE_UNUSED,                          0,
EVENT_STATUS_CHANGE },
```

```
    {   0x0005,       TYPE_UNUSED,                          0,
EVENT_STATUS_CHANGE },
    {   0x0006,       TYPE_UNUSED,                          0,
EVENT_STATUS_CHANGE },
    {   0x0007,       TYPE_UNUSED,                          0,
EVENT_STATUS_CHANGE },
    {   0x0008,       TYPE_SMART_DIAGNOSIS_CONTROL,         0,
EVENT_STATUS_CHANGE },
    {   0x0009,       TYPE_UNUSED,                          0,
EVENT_STATUS_CHANGE },
    {   0x000A,       TYPE_UNUSED,                          0,
EVENT_STATUS_CHANGE },
};
```

(2) 프로토콜 DATA 작성

- 프로토콜 ID에 해당하는 DATA 작성

- [제품] -> [서버]

```
// Ex)
U16 GetUserSystemFunction ( U16 mu16Func )
{
    U16 mu16Data = 0U;
    switch ( mu16Func )
    {
        case 0x0002 :               /* ID */
            mu16Data = F_ColdOff;   /* ID에 해당하는 DATA 설정 작성 */
        break;
    }
}
```

- [서버] -> [제품]

```
// Ex)
void SetUserSystemFunction ( U16 mu16Func, U16 mData )
{
    switch ( mu16Func )
```

```
    {
        case 0x0002 :               /* ID */
            if (mData == 1)         /* ID에 해당하는 DATA 설정 작성 */
            {
                F_ColdOff = CLEAR;
                Play_Voice(VOICE_COLD_ON_SETTING);
            }
            else
            {
                F_ColdOff = SET;
                Play_Voice(VOICE_COLD_OFF_SETTING);
            }
        break;
    }
}
```

- <2> WIFI_A102x_SensorList.c (센서 데이터)

(1) 프로토콜 LIST 작성

```
- API ID            : 프로토콜의 API ID
```

```
- DATA TYPE         : A1014 사용여부 / 스마트진단 프로토콜 사용여부 선택 (이외는 미사용)
    -> TYPE_UNSIGNED                : 부호 없는 데이터 (0 ~ 65535)
    -> TYPE_SIGNED                  : 부호 있는 데이터 (-32768 ~ 32767)
    -> TYPE_STYPE_SENSOR_WATT       : 전력 데이터 0.00 ~ 655.35
```

```
// Ex)
static const WifiNormalList_T WifiSensorProtocolList[] =
{   /*  API ID        Data Type     */
    {   0x0001,    TYPE_UNSIGNED    },
    {   0x0002,    TYPE_SIGNED      },
    {   0x0003,    TYPE_SENSOR_WATT },
    {   0x0004,    TYPE_SIGNED      },
    {   0x0005,    TYPE_SIGNED      },
    {   0x0006,    TYPE_SIGNED      },
    {   0x0007,    TYPE_SIGNED      },
    {   0x0008,    TYPE_SIGNED      },
    {   0x0009,    TYPE_SIGNED      },
    {   0x000A,    TYPE_SIGNED      },
};
```

(2) 프로토콜 DATA 작성

```
- 프로토콜 ID에 해당하는 DATA 작성
```

```
- [제품] -> [서버]
```

```
// Ex)
I16 GetUserSystemSensor ( U16 mu16Sen )
{
    I16 mi16Data = 0;

    switch ( mu16Sen )
    {
        case 0x0001:
            mi16Data = 1;
        break;
    }
    return mi16Data;
}
```

- <3> WIFI_A103x_ErrorList.c (에러 데이터)

(1) 프로토콜 LIST 작성

```
- API ID            : 프로토콜의 API ID
```

```
- DATA TYPE         : (미사용)
    -> TYPE_UNUSED  : DATA TYPE 미사용 (Normal)
```

```
- Initial VALUE     : Event DATA 처리를 위한 초기값 (수정 필요 없음. [0]고정)
```

```
- Event Form        : 해당 프로토콜 ID의 Event 처리를 위한 Evnet 유형 설정 (1초마다
Event Check 진행)
    -> EVENT_STATUS_CHANGE  : 해당 ID의 DATA 변경 시 Event DATA 전송 (고정)
```

```
// Ex)
WifiTxFuncEventList_T WifiErrorProtocolList[] =
{   /*  API ID              DATA_TYPE       PreVal(Initial)         EventForm
*/
    {   0x0001,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0002,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0003,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0004,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0005,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0006,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0007,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0008,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x0009,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
    {   0x000A,          TYPE_UNUSED,           0,
EVENT_STATUS_CHANGE },
};
```

(2) 프로토콜 DATA 작성

```
- 프로토콜 ID에 해당하는 DATA 작성
```

```
- [제품] -> [서버]
- 해당 에러프로토콜의 Error 없음 [0] / Error 있음 [1]
```

```
// Ex)
I16 GetUserSystemError ( U16 mu16Err )
{
    U16 mu16Data = 0;

    switch ( mu16Err )
    {
        case 0x0001:
            mu16Data = 1;
        break;
    }
    return mu16Data;
}
```

- <4> WIFI_A108x_A109x_PartList.c (스마트진단 데이터)

(1) 프로토콜 LIST 작성

```
- API ID            : 프로토콜의 API ID
```

```
- DATA TYPE
    -> TYPE_UNSIGNED                : 부호 없는 데이터 (0 ~ 65535)
    -> TYPE_SIGNED                  : 부호 있는 데이터 (-32768 ~ 32767)
```

```
// Ex)
static const WifiNormalList_T WifiParaProtocolList[] =
{   /*  API ID        */
    {   0x0001,    TYPE_UNSIGNED   },
    {   0x0002,    TYPE_UNSIGNED   },
    {   0x0003,    TYPE_UNSIGNED   },
    {   0x0004,    TYPE_UNSIGNED   },
    {   0x0005,    TYPE_UNSIGNED   },
    {   0x0006,    TYPE_UNSIGNED   },
    {   0x0007,    TYPE_UNSIGNED   },
    {   0x0008,    TYPE_UNSIGNED   },
    {   0x0009,    TYPE_UNSIGNED   },
    {   0x000A,    TYPE_UNSIGNED   },
};
```

(2) 프로토콜 DATA 작성

```
- 프로토콜 ID에 해당하는 DATA 작성
```

```
- [제품] -> [서버]
```

```c
// Ex)
U16 GetUserSystemPara ( U16 mu16Para )
{
    U16 mu16Data = 0U;
    switch ( mu16Para )
    {
        case 0x0001 :       /* ID */
            mu16Data = 1;   /* Value */
        break;
    }
    return mi16Data;
}
```

- <5> WIFI_A20xx_InformationList.c (부가정보 데이터)

(1) 프로토콜 LIST 작성

```
- API ID            : 프로토콜의 API ID
```

```
- DATA TYPE
    -> TYPE_INFO_NORMAL                 : 기본 데이터
    -> TYPE_INFO_TIME                   : 시간 데이터
    -> TYPE_INFO_WEATHER                : 날씨 데이터
    -> TYPE_INFO_AQI                    : 공기질 데이터
    -> TYPE_INFO_TEMP                   : 온도 데이터
    -> TYPE_INFO_DIAGNOSIS_RQ           : 정밀진단 결과 요청 데이터
    -> TYPE_INFO_JSON                   : SAP 부가정보 JSON 데이터
```

```
// Ex) 사용하는 데이터만 활성화 설정
/* (1) Initial or 24H cycle                 : TYPE_INFO_TIME / TYPE_INFO_NORMAL /
TYPE_INFO_WEATHER / TYPE_INFO_TEMP / TYPE_INFO_JSON */
/* (2) WifiSendData(WIFI_DATA_INFO) Reqest  : TYPE_INFO_TIME / TYPE_INFO_NORMAL /
TYPE_INFO_WEATHER */
/* (3) 20Min cycle                          : TYPE_INFO_NORMAL / TYPE_INFO_WEATHER
/ TYPE_INFO_TEMP */
/* (4) Smart Daignosis Result receive (Server->Product) : TYPE_INFO_DIAGNOSIS_RQ
(Displaying results on LCD)*/
/* (5) GetWifiRequestSAP() Reqest           : TYPE_INFO_JSON */
/* Event List */
static const WifiNormalList_T WifiInfoProtocolList[] =
{   /*  API ID                                  Data Type     */
    {   WIFI_INFO_0000_YEAR,                    TYPE_INFO_TIME         },
    {   WIFI_INFO_0001_MONTH,                   TYPE_INFO_TIME         },
    {   WIFI_INFO_0002_DATE,                    TYPE_INFO_TIME         },
    {   WIFI_INFO_0003_HOUR,                    TYPE_INFO_TIME         },
    {   WIFI_INFO_0004_MINUTE,                  TYPE_INFO_TIME         },
    {   WIFI_INFO_0005_SECOND,                  TYPE_INFO_TIME         },
    {   WIFI_INFO_0006_DAY,                     TYPE_INFO_TIME         },
    {   WIFI_INFO_0007_PART_PERIOD,             TYPE_INFO_NORMAL       },

    {   WIFI_INFO_0010_WEATHER,                 TYPE_INFO_WEATHER      },
    {   WIFI_INFO_0011_AQI,                     TYPE_INFO_AQI          },
    // {   WIFI_INFO_0012_PM10,                    TYPE_INFO_AQI          },
    // {   WIFI_INFO_0013_PM25,                    TYPE_INFO_AQI          },
    {   WIFI_INFO_0014_CURRENT_TEMP,            TYPE_INFO_TEMP         },
    {   WIFI_INFO_0015_HIGH_TEMP,               TYPE_INFO_TEMP         },
    {   WIFI_INFO_0016_LOW_TEMP,                TYPE_INFO_TEMP         },

    // {   WIFI_INFO_0020_LAST_HEART_YEAR,         TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0021_LAST_HEART_MONTH,        TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0022_LAST_HEART_DAY,          TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0023_NEXT_HEART_YEAR,         TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0024_NEXT_HEART_MONTH,        TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0025_NEXT_HEART_DAY,          TYPE_INFO_NORMAL       },
    {   WIFI_INFO_0026_FILTER_CHANGE_DAY1,      TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0027_FILTER_CHANGE_DAY2,      TYPE_INFO_NORMAL       },
    // {   WIFI_INFO_0028_FILTER_CHANGE_CYCLE,     TYPE_INFO_NORMAL       },
    {   WIFI_INFO_0029_DIAGNOSIS_RESULT_REQUEST, TYPE_INFO_DIAGNOSIS_RQ },
    {   WIFI_INFO_0030_STERILIZE_DATE,          TYPE_INFO_JSON         },
    {   WIFI_INFO_0031_SUPPLAY,                 TYPE_INFO_JSON         },
    {   WIFI_INFO_0032_ENVIRONMENT,             TYPE_INFO_JSON         }
};
```

(2) JSON 프로토콜 데이터 작성

- 프로토콜 ID에 해당하는 DATA 작성

- [서버] -> [제품]

```
// Ex) JSON 데이터 파싱
void SetUserSystemJsonObject ( U16 mu16Info, I8 *pBuf )
{
    U8 mu8KeyIndex = 0;
    U8 mu8SAPCode = 0;
    switch ( mu16Info )
    {
        case WIFI_INFO_JSON_0030_UV_DATE:   // UV 최근살균시간 / UV살균예정시간 프로
토콜

            _MEMSET_( (void __FAR*) &JsonObjectDepth_1, 0, sizeof(JSONObject) );
            _MEMSET_( (void __FAR*) &JsonObjectDepth_2, 0, sizeof(JSONObject) );

            cJsonParseJson(pBuf, &JsonObjectDepth_1, NULL);
            // Object O, Array X - Json parsing Depth1 c:"S1000"

            mu8SAPCode = JsonSAPCodeSuccess(&JsonObjectDepth_1);
            if (mu8SAPCode == F1000)      // SAP 접속 실패 (CODE : F1000, SAP001)
            {
                SetWifiSapStatus(STATUS_SVR_SAP_0030_UV_DATE, F1000);
                return;
            }
            else if (mu8SAPCode == E9000) // E9000 (데이터 없음)
            {
                SetWifiSapStatus(STATUS_SVR_SAP_0030_UV_DATE, E9000);
            }
            else                          // S1000 (데이터 가져오기 성공)
            {
                SetWifiSapStatus(STATUS_SVR_SAP_0030_UV_DATE, S1000);
            }

            mu8KeyIndex = (U8)GetJsonKeyIndexSearch(JSON_KEY_UV_FAUCETINFO,
&JsonObjectDepth_1);   // Key "fc"
            if(mu8KeyIndex != (U8)0xFF){ // Index 찾기 완료 시
                cJsonParseJson(JsonObjectDepth_1.items[mu8KeyIndex].value,
&JsonObjectDepth_2, NULL);
                // Object O, Array X  - Depth2 Parsing : "fc": { Key:Value,
Key:Value }

                /* 파우셋 UV 최근 동작시간 */
                GetJsonDateValue(gJsonValue.UV_Faucet.u8JsonDateLD,
JSON_KEY_LAST_DATE, &JsonObjectDepth_2);
                // [gJsonValue.UV_Faucet.u8JsonDateLD]에 파싱 DATA 설정

                /* 파우셋 UV 동작 예정시간 */
                GetJsonDateValue(gJsonValue.UV_Faucet.u8JsonDateND,
```

```
JSON_KEY_NEXT_DATE, &JsonObjectDepth_2);
                //[gJsonValue.UV_Faucet.u8JsonDateND]에 파싱 DATA 설정
            }
        break;
        default:

            break;

    }
}
```

- <6> WIFI_A105x_ParaList.c (파라미터 데이터)

- <7> WIFI_A10Ax_AccumulateList.c (누적정보 데이터)