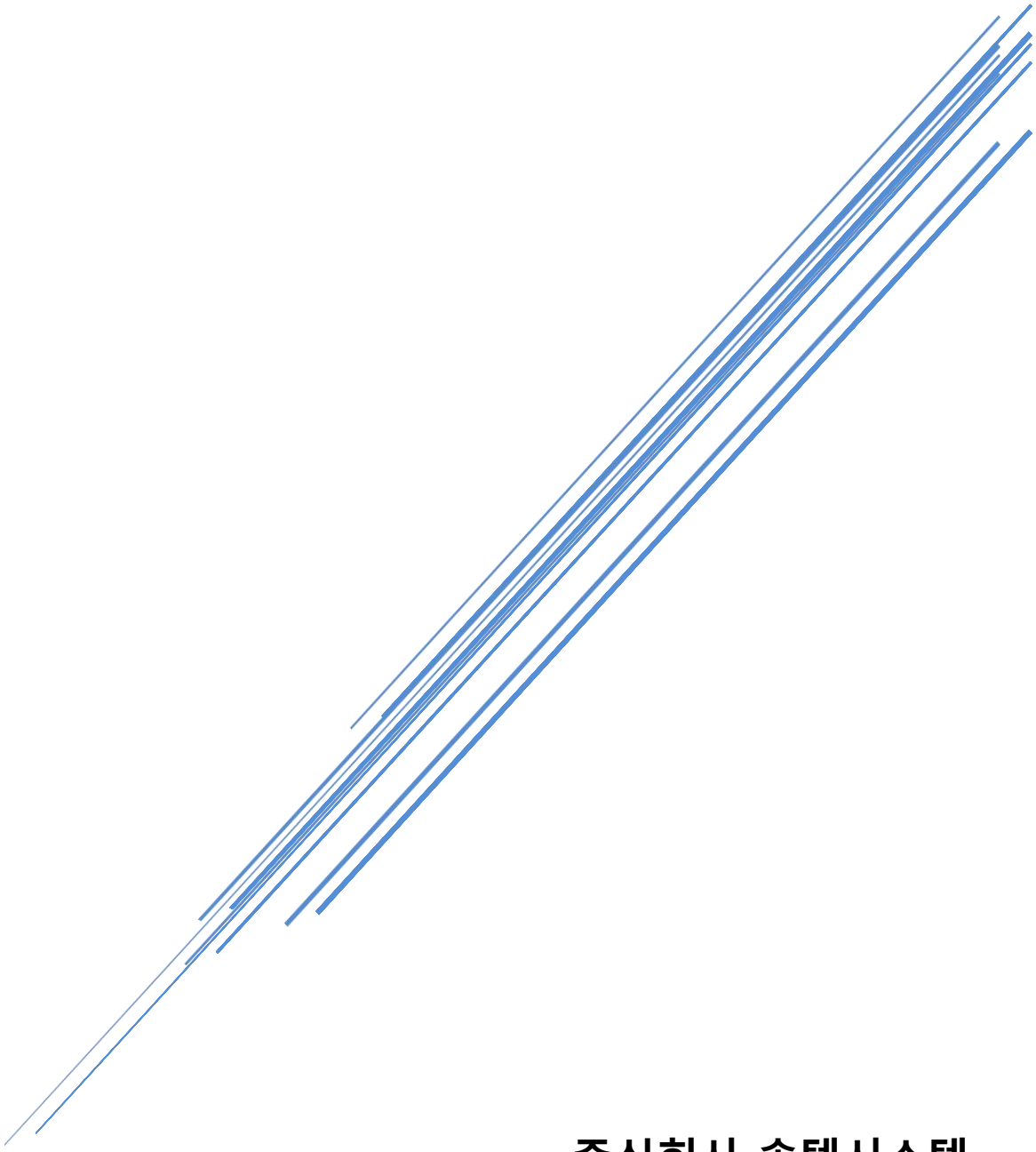
스마트 게이트웨이

관리자/사용자 매뉴얼



주식회사 솔텍시스템

개정이력

날짜	Rev.	개정내역
2023-05-20	Rev 1.00	초안
2023-06-23	Rev 1.10	LS Electric PLC사용매뉴얼 추가
2023-08-18	Rev 1.20	표준 시리얼통신을 위한 설명 추가 Frame 오브젝트 추가
2023-09-18	Rev 1.30	Frame을 Arr 오브젝트로 변경 Arr 오브젝트에 getAsc() 함수 추가
2023-09-21	Rev 1.40	MSSQL 서버 드라이버 추가
2023-09-27	Rev 1.50	Db.set(tag, value)를 Db.set(tag, value, timestamp)로 확장 Melsec Q시리즈 드라이버 추가
2023-10-06	Rev 1.60	Siemens driver 추가(ISO-TSAP Class 0 방식))
2023-11-29	Rev 1.70	Db.set() 함수기능 확장(품질정보, 데이터타입정보 추가) MBtcprtu 드라이버(Slave No 설정 가능하도록 함수 확장)
2024-05-27	Rev 1.80	FX 시리즈 PLC 드라이버 추가 LS Electric 시리얼 드라이버 추가 JDBC 드라이버 추가
2024-06-21	Rev 1.90	각종변환관리를 위한 오브젝트와 시스템관련 오브젝트 추가
2024-07-01	Rev 1.91	Modbus TCP-RTU mapTo함수 추가
2024-07-08	Rev 1.92	장치쓰기 기능 설명 추가
2024-09-12	Rev 2.00	ABB PLC 드라이버 추가
2024-11-28	Rev 2.10	Panasonic 조명제어 PLC 드라이버 추가
2025-01-21	Rev 2.20	Restful API client 드라이버 추가
2025-05-12	Rev 2.30	LCS Restful API 드라이버 추가 오타 수정

목차

1. 개요 4
1.1. 문서개요 4
1.2. 매뉴얼 구성 4
2. 스마트 게이트웨이 개요 5
2.1. 개요 5
2.2. 내부 모듈 구성 6
3. 스마트 게이트웨이 구성 (또는 화면) 8
3.1. 스마트 게이트웨이 접속 8
3.2. 로그인 화면 8
3.3. 일반 사용자 9
3.4. 관리자 12
3.4.1. 시스템 정보 12
3.4.2. 태그관리 화면 15
3.4.3. 스크립트 관리 화면 15
4. 스마트 게이트웨이 기능 19
4.1. 장치로부터 읽기 19
4.2. 장치로 쓰기 19
4.3. 자바스크립트 엔진 20
5. 내장 자바스크립트 오브젝트 22
5.1. 통신드라이버 관리를 위한 함수 23
5.2. 데이터관리를 위한 함수 23
5.3. 데이터변환을 위한 오브젝트 25
5.4. 데이터 포맷관리를 위한 함수 26
5.5. 배열관리를 위한 오브젝트 27
5.6. 로그관리를 위한 오브젝트 29
5.7. 시스템관리를 위한 오브젝트 29
6. 드라이버별 자바스크립트 오브젝트 30
6.1. 범용 드라이버 30
6.1.1. Jdbc 30
6.1.2. Rest 31
6.2. 표준 통신드라이버 32
6.2.1. Serial 33
6.3. MODBUS 통신드라이버 35
6.3.1. mbmaster 37
6.3.2. mbslave 39
6.3.3. mbclient 40
6.3.4. mbserver 42

6.3.5. mbrtutcp 42
6.3.6. mbtcprtu 45
6.4. OPC UA 통신드라이버 47
6.4.1. uaclient 47
6.5. PLC 통신드라이버 48
6.5.1. LS ELECTRIC PLC: lsclient 48
6.5.2. LS ELECTRIC PLC: lsmaster 50
6.5.3. 미쓰비시 PLC: mqclient 53
6.5.4. 미쓰비시 PLC: fxmaster 56
6.5.5. 지멘스 PLC: s7client 59
6.5.6. ABB PLC: eipclient 62
6.5.7. Panasonic 조명제어 PLC: NCUmaster 64
6.5.8. Panasonic OPIU 조명제어 PLC: LCSRest 76
7. 스마트 게이트웨이 구성사례 79
7.1. 범용 79
7.2. SERIAL 80
7.3. MODBUS 83
7.4. 표준 프로토콜 86

1. 개요

1.1. 문서개요

본 문서는 자바스크립트 기반의 스마트 게이트웨이 단말을 운용하기 위한 사용자 매뉴얼 문서로 스마트 게이트웨이의 개요, 구성, 스마트 게이트웨이에서 확장된 자바스크립트 함수 설명과 프로토콜별 자바스크립트 작성 방법에 대하여 기술하였습니다.

1.2. 매뉴얼 구성

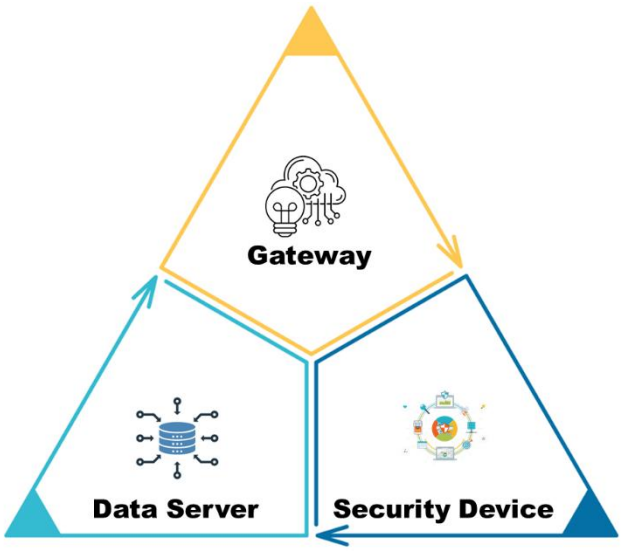
본 문서는 자바스크립트 기반 스마트 게이트웨이 단말의 사용자 매뉴얼로 다음과 같은 항목이 기술되어 있습니다.

- ❑ 스마트 게이트웨이 개요
- ❑ 스마트 게이트웨이 구성 (화면)
- ❑ 스마트 게이트웨이 구성사례
- ❑ 내장 자바스크립트 함수 설명
- ❑ 드라이버별 함수 설명

2. 스마트 게이트웨이 개요

2.1. 개요

스마트 게이트웨이는 상이한 프로토콜을 가진 장치 간의 데이터 교환과, 수집된 데이터 가공을 위한 게이트웨이 제품으로 프로토콜 간의 변환을 위한 게이트웨이 기능, 수집된 데이터를 MES, ERP 시스템과 같은 상위 서버로의 연계를 위한 데이터 서버, 이더넷 시리얼 변환 기능을 이용한 네트워크 보안장비의 기능을 제공합니다.



[그림 1] 스마트 게이트웨이 역할

스마트 게이트웨이가 제공하는 기능은 다음과 같습니다.

프로토콜 변환 Gateway

서로 상이한 프로토콜을 갖는 장치 간의 데이터 변환을 통하여 두 장치 간의 데이터를 교환할 수 있는 기능을 수행합니다.

데이터 서버

현장 내의 설비로부터 수집된 공정 데이터를 일시 저장하고, MES, ERP 시스템 등과 같은 상위 시스템으로 전송하기 위한 기능을 수행하며, 이를 위하여 수집된 데이터를 가공할 수 있는 ECMAScript E5/E5.1 규격과 호환되는 자바스크립트 엔진을 내장하고 있습니다.

보안장비

이더넷으로 구성된 네트워크 구간을 시리얼로 변환하여 두 이더넷 네트워크를 물리적으로 망을 분리하여 보안사고의 전파를 예방할 수 있습니다.

2.2. 내부 모듈 구성

스마트 게이트웨이는 프로토콜변환, 데이터서버, 보안장비 기능을 수행하기 위하여 독립적으로 구성된 소프트웨어 모듈로 이루어져 있습니다.

[그림 2] 스마트 게이트웨이 모듈 구성

스마트 게이트웨이를 구성하는 소프트웨어 모듈은 위의 그림과 같으며 각각 다음과 같은 기능을 제공합니다.

Data Repository

이름 기반의 데이터를 관리하기 위한 데이터 저장소이며 통신 드라이버를 통하여 수집된 데이터를 일시 저장하고, 저장된 데이터를 통신드라이버를 통하여 연결된 설비로 전송 가능하게 합니다.

APIs for Data Repository and Managing drivers.

데이터저장소로부터 데이터를 읽거나 쓰기 위한 API 와 통신드라이버를 로드하기 위한 API 함수 모듈입니다.

Loadable Drivers

메모리 절약을 위하여 필요시 통신드라이버를 로드 및 언로드 할 수 있는 소프트웨어 모듈입니다. 통신드라이버는 프로토콜별로 다양하게 제공합니다.

Java Script Engine

ECMAScript E5/E5.1 규격을 지원하는 자바스크립트 엔진으로, 자바스크립트를 이용하여 통신드라이버를 로드 하거나, 데이터저장소로부터 데이터를 읽거나 쓸 수 있는 기능을 지원합니다.

Java Script Manager

통신드라이버 별로 연결하거나 수집된 데이터를 가공하기 위한 자바스크립트 파일을 관리하기 위한 소프트웨어 모듈입니다.

Driver Manager

통신드라이버가 메모리에 로드 되어 통신드라이버 고유의 자바스크립트 함수를 생성한 이후 자바 스크립트 엔진에 등록을 하기 위한 소프트웨어 모듈입니다.

3. 스마트 게이트웨이 구성 (또는 화면)

3.1. 스마트 게이트웨이 접속

스마트 게이트웨이는 메모리용량으로 웹서버 내장이 어려운 Lite 버전을 제외하고 자체에 웹서버를 내장하고 있어 웹페이지를 이용하여 상태를 감시하거나 설정 내용을 변경할 수 있습니다. 웹서버에 접속하기 위하여 아래와 같이 입력합니다.

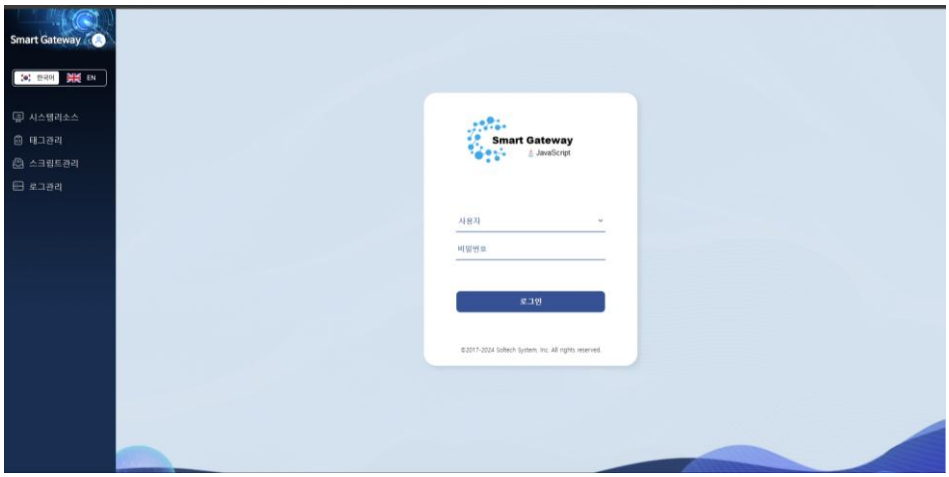
```
http://ip-address:8080/smartgateway/
```

위에서 *Ip-address* 는 스마트 게이트웨이에 설정된 IP 주소로 공장초기 설정 값은 192.168.1.231 로 되어 있어 스마트 게이트웨이에 접속하기 위하여 아래와 같이 입력합니다.

```
http://192.168.1.231:8080/smartgateway/
```

3.2. 로그인 화면

스마트 게이트웨이를 모니터링하거나 설정하기 위해서는 로그인을 하여야 하며 초기 사용자 암호는 "1234"입니다.



[그림 3] 로그인 화면

스마트 게이트웨이에서 제공하는 화면은 세 종류의 권한에 따라 운영이 가능합니다. 일반 사용자 권한은 모니터링만 가능하며, 관리자 권한으로 로그인할 경우 설정내용을 변경할 수 있는 기능을 제공합니다. 로그인 관리자의 경우에는 다른 메뉴는 접근이 불가하고 로그인 정보변경만 가능합니다.

초기 설정되어 있는 로그인 화면은 권한별로 다음과 같습니다. 초기암호는 로그인 관리자로 로그인 하여 변경 가능하며, 변경된 이후에는 변경된 로그인 암호로만 접근이 가능합니다.

사용자	관리자	로그인 관리자
1234	SmartGateway!1	SmartGateway!1

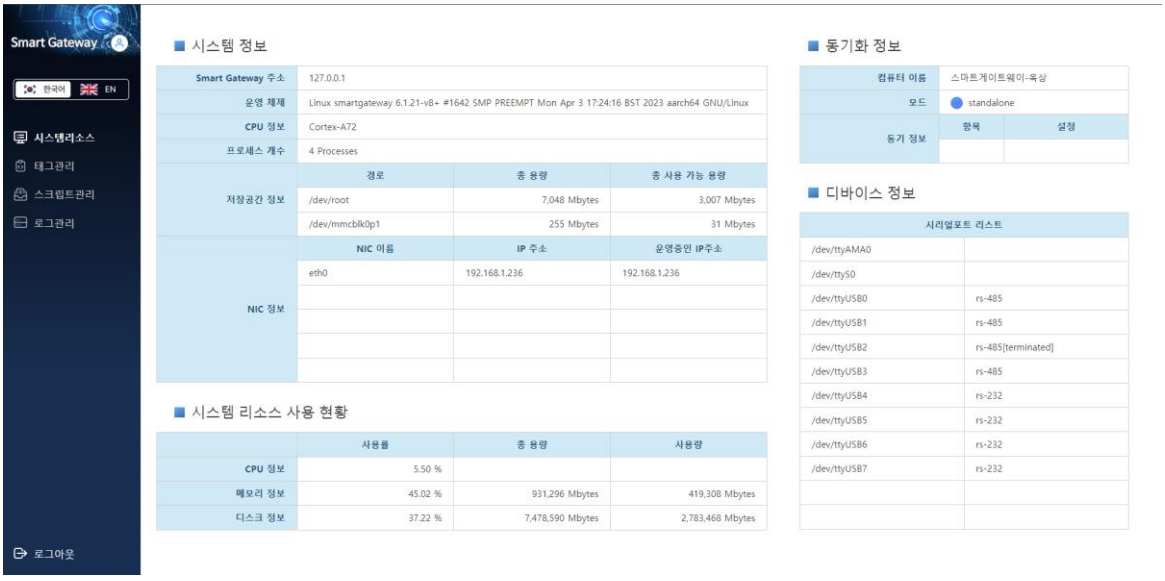
[표 1] 로그인 암호

3.3. 일반 사용자

로그인 화면에서 일반사용자로 로그인 하면 스마트 게이트웨이의 설정은 불가능하고 단순 모니터링 가능합니다.

시스템 정보

스마트 게이트웨이의 시스템 정보, 리소스 현황 등을 확인하기 위한 화면입니다. 운영자는 시스템 정보화면에서 시스템 정보, 리소스 사용현황, 동기화정보, 디바이스 정보를 확인할 수 있습니다.



[그림 4] 사용자-시스템 상태감시

시스템 정보

스마트 게이트웨이 연결주소, 운영체제, CPU 정보, 프로세스개수, 저장공간정보, NIC(Network Interface Card)정보를 화면에 표시합니다.

동기화 정보

컴퓨터 이름과 이중화 구성 시 필요한 동기화 설정 정보를 표시합니다.

시스템 리소스 사용현황

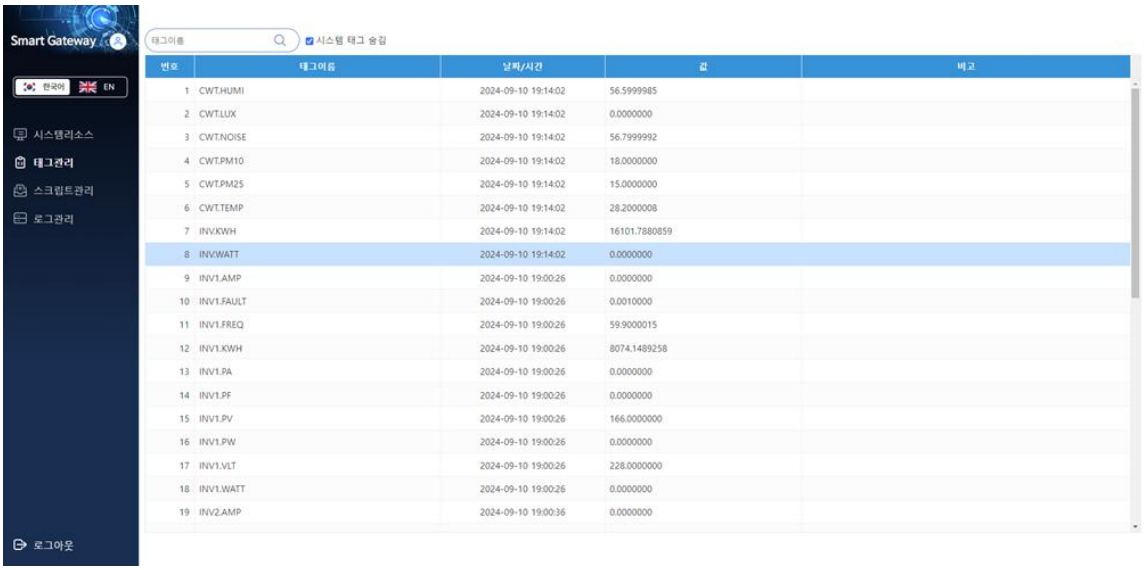
스마트 게이트웨이에서 사용중인 CPU, 메모리, 디스크 사용량 정보를 표시합니다.

디바이스 정보

스마트 게이트웨이에서 이용 가능한 시리얼 포트 리스트 정보를 화면에 표시합니다. 스마트 게이트웨이-STD 의 경우 시리얼포트의 타입(RS-232/422/485) 정보 및 RS-422/485 인 경우 내부 종단저항 연결여부를 포함된 정보를 표시합니다.

태그관리

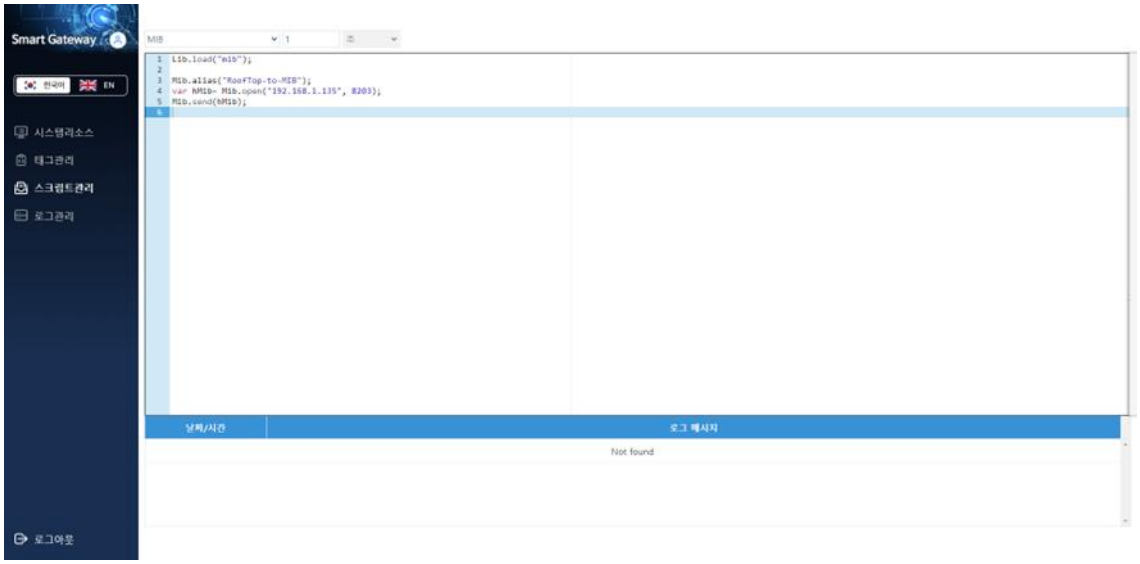
스마트 게이트웨이에서 수집하고 있는 태그의 현재 값, 수집시각 등의 정보를 확인할 수 있습니다.



[그림 5] 사용자-태그관리

스크립트관리

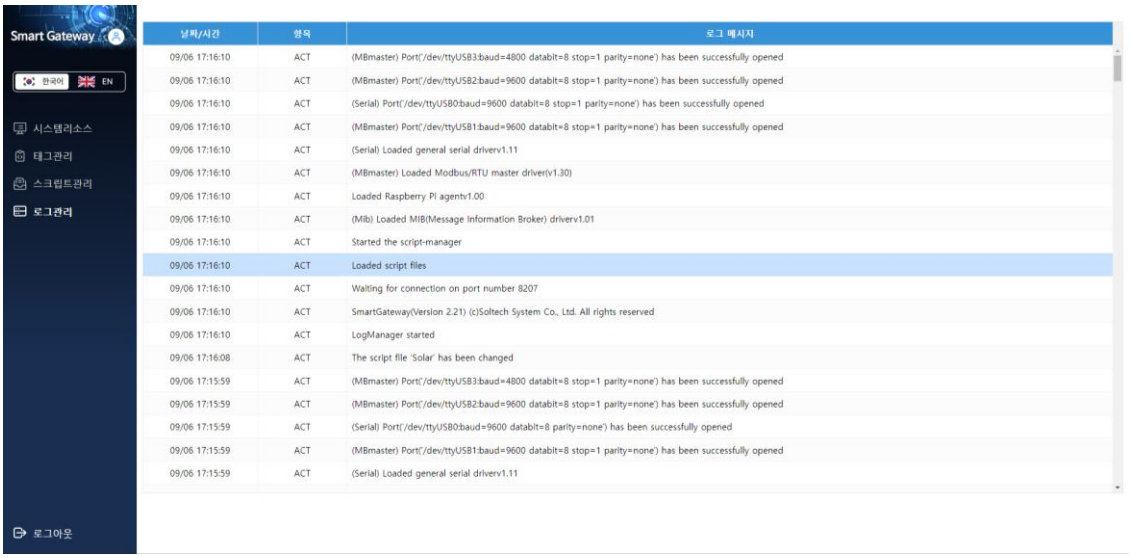
스마트 게이트웨이에서 동작중인 스크립트를 조회하거나 작성된 스크립트의 내용을 확인하거나, 스크립트의 동작주기를 확인할 수 있습니다.



[그림 6] 사용자-스크립트관리

로그관리

로그관리 화면은 스마트 게이트웨이가 동작하면서 생성되는 로그를 확인하기 위한 화면으로 "새로 고침" 버튼을 누르거나 화면이 갱신될 때마다 추가로 생성된 로그를 확인할 수 있습니다.

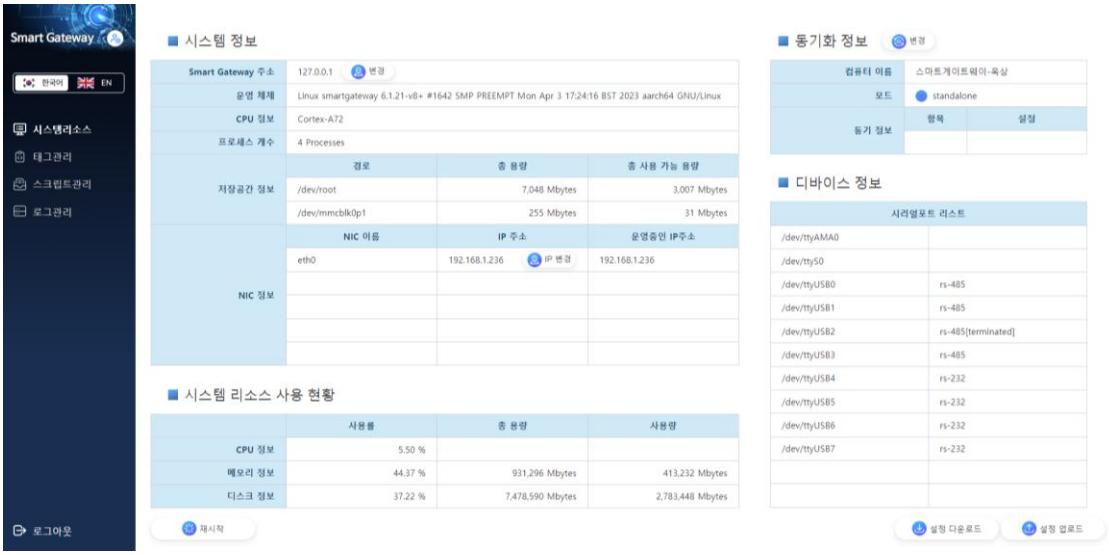


[그림 7] 사용자-스크립트관리

3.4. 관리자

3.4.1. 시스템 정보

시스템 정보화면에서는 스마트 게이트웨이의 하드웨어 및 리소스 상태정보를 출력하여 운영자가 스마트 게이트웨이의 상태를 모니터링 하거나 IP 주소를 변경할 수 있도록 합니다.



[그림 8] 관리자-시스템 상태감시화면

시스템 정보

스마트 게이트웨이 연결주소, 운영체제, CPU 정보, 프로세스개수, 저장공간정보, NIC(Network Interface Card)정보를 화면에 표시합니다.

동기화 정보

컴퓨터 이름과 이중화 구성 시 필요한 동기화 설정 정보를 표시합니다.

시스템 리소스 사용현황

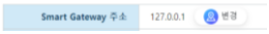
스마트 게이트웨이에서 사용중인 CPU, 메모리, 디스크 사용량 정보를 표시합니다.

디바이스 정보

스마트 게이트웨이에서 이용 가능한 시리얼 포트 리스트 정보를 화면에 표시합니다. 스마트 게이트웨이-STD 의 경우 시리얼포트의 타입(RS-232/422/485)정보 및 RS-422/485 인 경우 내부 종단저항 연결여부를 포함된 정보를 표시합니다

게이트웨이 연결주소변경

☞ 스마트 게이트웨이 Lite 버전의 경우 리소스 부족으로 웹서버를 내장할 수 없는 경우 외부 PC에 웹서버를 설치하여 스마트 게이트웨이와의 연결한 다음 모니터링 및 구성변경이 가능하게 하기 위한 기능입니다. 스마트 게이트웨이 자체에 웹서버가 내장된 경우에는 사용하지 않는 기능입니다.

여러 대의 게이트웨이를 하나의 웹페이지를 이용하고자 할 경우에는 스마트 게이트웨이의 연결 주소를 변경하여 사용할 수 있습니다. 스마트 게이트웨이의 리소스가 부족하여 스마트 게이트웨이에서 웹서버를 제공하지 않을 경우 운영 PC 설치된 웹서버를 이용하여 스마트 게이트웨이의 감시 및 설정이 가능하게 됩니다. 시스템 상태감시화면에서 

내의 버튼을 클릭하여 연결하고자 하는 주소로 변경한 다음 PC 의 웹서버를 재 시작하여 새로운 스마트 게이트웨이와 연결을 하여 감시 및 변경작업을 진행합니다.

[그림 9] 관리자-스마트게이트웨이 연결주소 변경

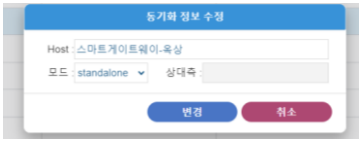
동기화정보

☞ 동기화 정보를 변경한 이후에는 스마트 게이트웨이 재시작버튼을 눌러야만 변경된 동기화 정보에 따라 스마트 게이트웨이가 동작됩니다.

스마트 게이트웨이의 이름을 변경하고자 하거나 동기화 정보를 변경하고자 할 경우 사용하는 화면으로

에서 변경 버튼을 눌러 동기화 정보를 변경합니다.

이름 변경은 변경 후 즉시 반영되며 동기화 정보를 변경한 경우에는 재시작 버튼을 눌러 스마트 게이트웨이의 소프트웨어 모듈이 재 시작하도록 하여 변경된 동기화 정보를 인식할 수 있도록 합니다.



[그림 10] 관리자-동기화 정보 변경

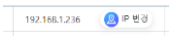
스마트 게이트웨이의 모드는 Standalone, Main, Shadow 가 있으며 Main 및 Shadow 모드 일 경우 두 대의 스마트 게이트웨이는 태그 동기화 작업이 진행됩니다.

모드	설명
Standalone	스마트 게이트웨이 하나로만 운영하고자 할 경우 사용
Main	이중화 구성할 경우 서비스를 제공하거나 장치로부터 데이터를 취득하는 모드
Shadow	이중화 구성에서 Main이 중지될 경우, Main을 대신하여 서비스를 제공하거나 PLC로부터 데이터를 취득하기 위한 모드

[표 2] 동기화 모드

스마트 게이트웨이 IP 주소 변경

☞ 스마트 게이트웨이 IP주소를 변경한 이후에는 스마트 게이트웨이 리부트 버튼을 눌러야만 스마트 게이트웨이 하드웨어가 재시작 되어야만 변경된 IP주소로 스마트 게이트웨이가 동작됩니다.

스마트 게이트웨이의 IP 주소는 운영자에 의하여 변경이 가능하며, IP 주소를 변경하기 위하여 

에서 변경버튼을 눌러 변경하고자 하는 IP 주소로 변경한 이후 스마트 게이트웨이를 리부팅하면 변경된 IP 주소로 스마트 게이트웨이가 동작합니다.

[그림 11] 관리자-동기화 정보 변경

재시작

스마트 게이트웨이의 소프트웨어 모듈을 재시작하기 위한 기능으로 재시작 버튼을 누르면 소프트웨어 모듈이 재 시작되어 수집된 모든 태그 정보는 초기화되고 모든 통신 드라이버 모듈로 재 기동 됩니다.

리부트

IP 변경작업과 같이 스마트 게이트웨이의 하드웨어 모듈을 재 시작할 필요가 있을 경우 리부트 메뉴가 표시되며, 리부트 버튼을 클릭하면 스마트 게이트웨이 하드웨어가 재 시작됩니다.

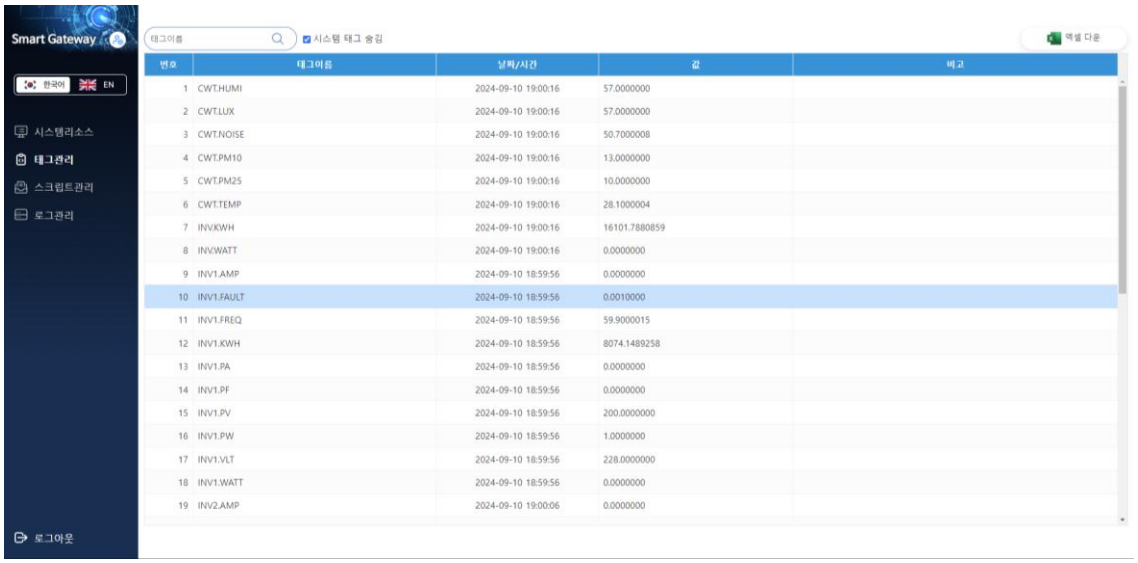
설정 다운로드/설정 업로드

스마트 게이트웨이의 IP 정보, 동기화정보, 스크립트 정보를 다운로드하기 위한 기능으로 스마트 게이트웨이를 재 설정하고자 할 경우 이전에 다운로드 받은 정보를 이용하여 손쉽게 스마트 게이트웨이의 설정을 완료할 수 있습니다.

3.4.2. 태그관리 화면

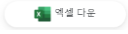
태그관리 화면은 스마트 게이트웨이에서 수집하고 있는 태그 값을 실시간으로 출력하기 위한 화면으로 운영자에게 스마트 게이트웨이의 통신드라이버를 통하여 수집된 자료 값, 수집시간 정보를 제공합니다.

운영자는 이 화면을 통하여 드라이버가 정상적으로 동작하고 있는지를 확인할 수 있으며, 또한 자바스크립트에 의하여 가공된 값이 정상적으로 동작하고 있는지를 실시간 확인 가능합니다.



[그림 12] 태그관리 화면

엑셀다운

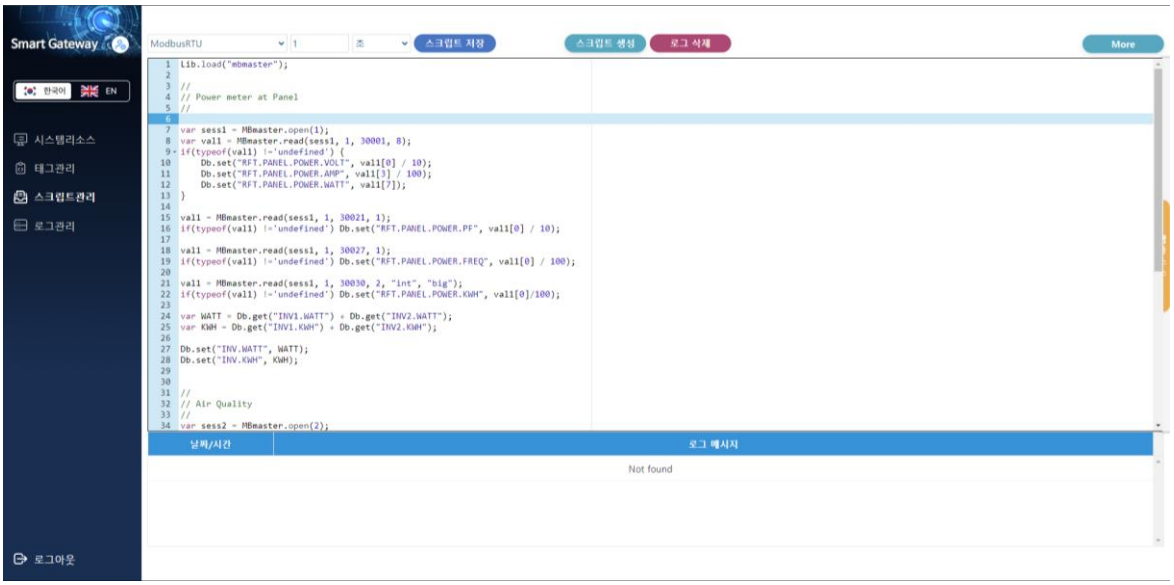
스마트 게이트웨이에서 수집된 태그이름, 시각, 현재 값 등의 정보를 엑셀파일로 다운로드하기 위한 기능으로 

버튼을 클릭하면 현재까지 수집되어 있는 태그정보를 엑셀로 다운받을 수 있습니다.

3.4.3. 스크립트 관리 화면

스크립트 관리화면은 스마트 게이트웨이가 지원하는 자바스크립트를 이용하여 통신 설정을 하거나 수집된 데이터의 변환을 할 수 있는 기능을 제공합니다. 운영자는 이 화면을 이용하여 작성된 스크립트를 확인할 수 있으며 관리자 권한으로 로그인 하게 되면 스크립트 수정이 가능합니다.

스크립트를 변경한 이후에는 스크립트 저장버튼을 눌러야만 변경된 내용이 스마트 게이트웨이 내부에 저장되며 스크립트 저장버튼을 누르지 않고 다른 화면으로 이동할 경우 수정한 내용이 반영되지 않을 수 있습니다.



[그림 13] 스크립트 관리 화면

스크립트 실행간격

스크립트를 실행하기 위한 방식은 주기적으로 실행하는 방법과 지정된 시각에 실행하는 방식과 같이 두 가지 종류가 있습니다. 실행 간격에서 '초'로 선택하면 주기적으로 실행하는 것을 의미하며 주기적으로 실행하고자 하는 시간을 초 단위로 입력하면 됩니다.

'초' 이외의 선택 값은 지정된 시각에 시작하기 위한 선택 값으로, 예를 들어 '시간'을 선택하면 매일 입력된 값의 시각에 실행하게 됨을 의미합니다. 만일 '10 시' 값이 입력되었다면 매일 10 시에 스크립트는 1 회 실행하게 됩니다.

설정내용에 따른 실행 간격은 다음과 같습니다.

방법	실행간격	설명	기준
1회	한번	스마트게이트웨이가 시작되면 1회 시작됩니다.	
주기	10 초	10초 간격으로 스크립트가 실행됩니다.	
	60 초	60초(1분) 간격으로 스크립트가 한번 실행됩니다.	
지정시간	15 분	매 시 15분에 스크립트가 한번 실행됩니다.	매시간
	1 시	매일 1:00시에 스크립트가 한번 실행됩니다.	매일
	5 일	매월 5일에 스크립트가 한번 실행됩니다.	매월

[표 3] 스크립트 실행간격 예제

스크립트 생성

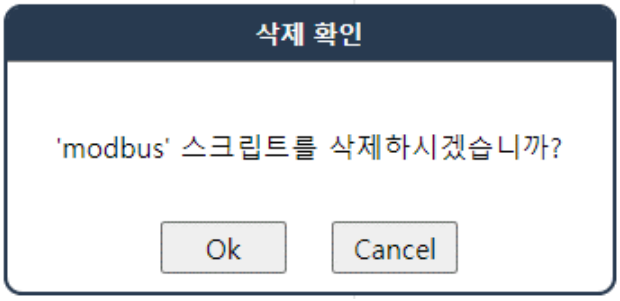
스크립트를 새로 작성하고자 하는 경우에는 스크립트 생성버튼을 눌러 생성하고자 하는 스크립트 이름과 실행주기를 입력한 이후 스크립트를 작성한 다음 스크립트 저장버튼을 눌러 저장하도록 합니다.

[그림 14] 관리자 모드 - 스크립트 등록 화면

스크립트 실행 간격은 초, 분, 시간, 일 간격으로 지정이 가능하며 지정된 시간 간격으로 스크립트는 동작하게 됩니다. 예를 들어 10 분으로 지정할 경우 매시간 10 분에 스크립트는 실행됩니다.

스크립트 삭제

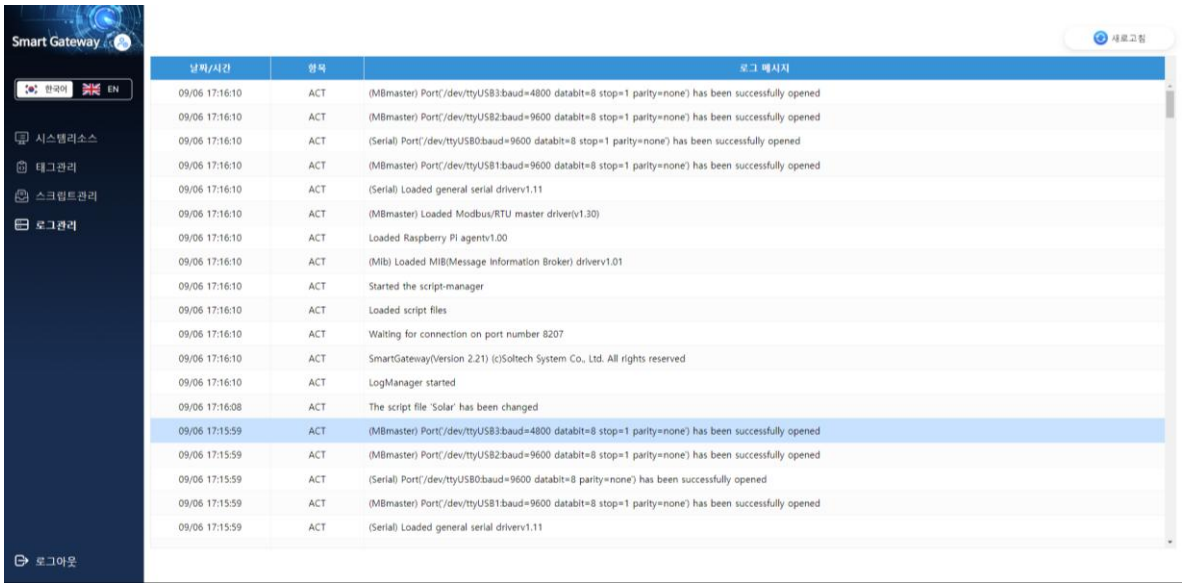
불필요한 스크립트는 삭제가 가능하며 삭제버튼을 클릭하면 선택된 스크립트는 스마트 게이트웨이에서 삭제됩니다. 삭제할 경우 다시 복구할 수 없으므로 삭제하기 전에 충분히 확인하고 삭제하여야 합니다.



[그림 15] 관리자 모드 - 스크립트 삭제 화면

로그관리

로그관리 화면은 스마트 게이트웨이가 동작하면서 생성되는 로그를 확인하기 위한 화면으로 "새로 고침" 버튼을 누르거나 화면이 갱신될 때마다 추가로 생성된 로그를 확인할 수 있습니다.



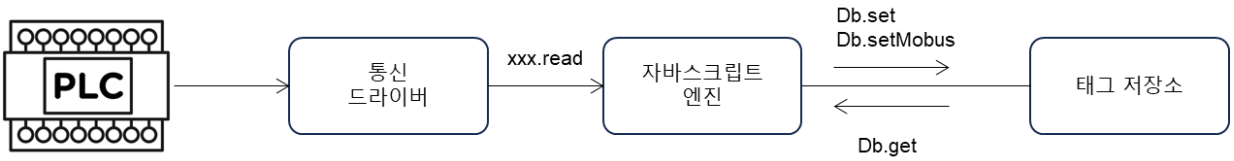
[그림 16] 로그관리 화면

4. 스마트 게이트웨이 기능

4.1. 장치로부터 읽기

스마트 게이트웨이는 연결된 장치로부터의 데이터 읽기 기능을 제공합니다. 연결된 장치로의 읽기 기능은 자바 스크립트를 이용한 읽기가 가능하며, 장치로부터 읽은 데이터는 태그로 구성하여 스마트 게이트웨이에서 관리를 하거나 mbrtutcp 혹은 mbtcprtu 드라이버를 적용하여 사용할 경우에는 태그로 구성할 필요가 없을 경우에는 장치로부터 읽어온 데이터를 타 시스템으로 전송이 가능합니다.

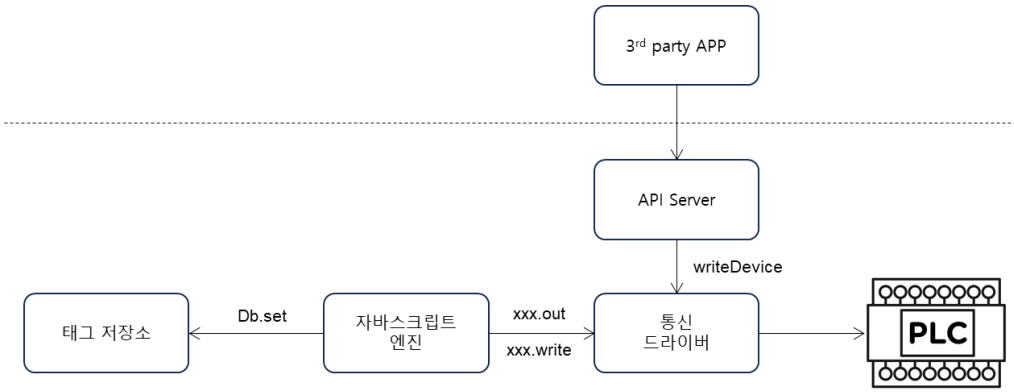
단, 장치로부터 읽어온 자료를 태그로 구성되지 않은 경우에는 태그 간의 연산이나 모니터링 기능이 제공되지 않습니다.



[그림 17] 장치로부터 읽기

4.2. 장치로 쓰기

스마트 게이트웨이는 연결된 장치로의 쓰기 기능을 제공합니다. 연결된 장치로의 쓰기 기능은 자바 스크립트를 이용한 직접 쓰기 기능과 외부로부터의 쓰기명령을 수신하여 장비로의 쓰기를 수행할 수 있는 간접 쓰기 기능을 제공합니다.



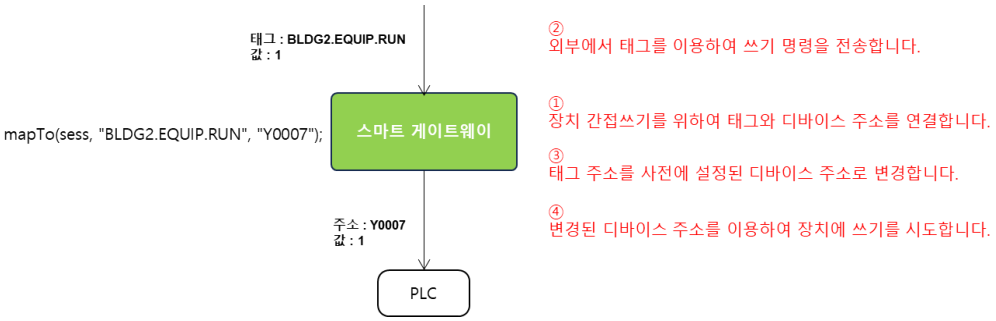
[그림 18] 장치로 쓰기

직접쓰기

스마트 게이트웨이에서 제공하는 직접쓰기 기능은 자바스크립트를 이용하여 제공되는 기능으로 사용자는 드라이버에서 제공하는 쓰기기능과 자바스크립트를 이용하여 연결된 장비의 특정위치에 원하는 값에 대한 쓰기를 처리할 수 있습니다.

간접쓰기

간접쓰기 기능은 외부로부터의 쓰기명령을 수신하여 연결된 장비의 특정위치에 수신된 값에 대한 장치쓰기를 처리하기 위한 기능으로 태그를 디바이스 주소로 변환하기 위한 mapTo 기능을 이용하여 처리됩니다.



[그림 19] 기능 – 간접쓰기

4.3. 자바스크립트 엔진

스마트 게이트웨이는 자바스크립트를 내장하고 있으며, 내장된 자바스크립트를 이용하여 장치로부터 데이터를 읽어오거나, 읽어온 데이터를 가공하거나, 가공된 데이터를 장치로 쓸 수 있는 기능을 제공합니다.

자바 스크립트 엔진에서 사용가능한 함수 오브젝트는 라이브러리 관리 함수를 이용하여 메모리에 로드 하여 사용가능한 함수 오브젝트와 메모리에 로드 할 필요 없이 기본적으로 사용가능한 오브젝트 함수로 나뉘어집니다.

함수 오브젝트 분류	비고
기본 함수	라이브러리 관리 함수를 이용하여 메모리에 로드 할 필요 없이 사용 가능한 함수 오브젝트 ❑ Arr: 배열 관리 함수 ❑ Db: 태그관리 함수 ❑ Lib: 라이브러리 관리 함수 ❑ Log: 로그관리 함수 ❑ System: 시스템 관리 함수
로더블 함수	라이브러리 관리 함수를 이용하여 메모리에 로드 한 이후 사용 가능한 함수 오브젝트 ❑ Conv: 엔지니어링 변환 함수 ❑ Jdbc: 데이터베이스 관리 함수 ❑ 모든 통신 드라이버 함수

[표 4] 스크립트 실행간격 예제

5. 내장 자바스크립트 오브젝트

내장 자바스크립트 함수는 통신드라이버 관리, 데이터 저장소 관리, 변환관리, 로그 관리를 위하여 드라이버 로딩과 같은 작업 없이 사용할 수 있는 함수입니다.

통신라이브러리 관리 오브젝트

통신라이브러리 관리함수는 통신드라이버를 메모리로 로딩하기 위하여 'Lib'로 명명된 오브젝트를 사용할 수 있으며, 해당통신 드라이버가 로딩된 이후에는 해당 통신 드라이버의 오브젝트를 사용할 수 있습니다. 예를 들어

Lib.load("mbclient") 함수의 경우 modbus/tcp client 프로토콜을 이용하기 위한 함수로 드라이버가 성공적으로 로드되면 MBclient 로 명명된 오브젝트와 그 부속함수를 사용할 수 있습니다.

Lib.load(), Lib.unload()

통신드라이버 관리를 위한 오브젝트로 load(), unload() 함수를 제공합니다.

데이터관리 오브젝트

데이터 저장소내의 데이터를 조회하거나 데이터를 저장하기 위하여 'Db'로 명명된 오브젝트를 사용할 수 있으며 데이터를 가져오거나, 저장할 수 있는 함수를 제공합니다.

Db.get(), Db.set(), Db.setModbus()

데이터 저장소 관리를 위한 오브젝트로 get(), set(), setModbus() 함수를 제공합니다.

로그관리 오브젝트

스크립트 작성시 디버깅 등을 위하여 디버그 정보를 화면에 표시하기 위하여 'Log'로 명명된 오브젝트를 사용할 수 있습니다.

Log.add()

로그 관리를 위한 오브젝트로 add() 함수를 제공합니다.

시스템관리 오브젝트

스크립트 작성시 시스템 관련정보에 접근하기 위하여 'System'로 명명된 오브젝트를 사용할 수 있습니다.

System.sleep()

지정된 msec 만큼 기능을 일시 정지합니다.

System.isActive()

이중화 구성에서 동작중인 스마트게이트인지 확인하기 위한 목적으로 사용합니다.

Main/Shadow 구성에서 Main 스마트 게이트웨이가 정상적으로 동작 중이면 Main 에서는 True 를, Shadow 에서는 False 값이 반환됩니다.

5.1. 통신드라이버 관리를 위한 함수

스마트 게이트웨이는 메모리를 효율적으로 관리하기 위하여 모든 통신드라이버를 메모리에 올려 사용하는 방식이 아닌 필요시 메모리에 로드 하여 사용할 수 있도록 하기 위하여 Lib 오브젝트와 load(), unload() 함수를 제공하고 있습니다.

Lib.load(*driver-name*)

driver-name 으로 지정된 통신드라이버를 메모리에 로드 하여 연결된 장치로부터 데이터를 읽거나 쓰기 위한 함수입니다.

Lib.unload(*driver-name*)

driver-name 으로 지정된 통신드라이버를 메모리에서 언 로드하기 위한 함수입니다.

5.2. 데이터관리를 위한 함수

스마트 게이트웨이 내부의 데이터 저장소 관리를 위한 함수로 함수에 접근하기 위한 오브젝트명은 Db 이며 데이터 쓰기, 읽기와 Modbus 통신을 위한 데이터 구성을 위한 쓰기 함수를 제공하고 있습니다.

Db.get(*item-name*)

데이터 저장소로부터 ***item-name*** 으로 지정된 데이터 값을 읽어오기 위한 함수입니다.

- ***item-name**:* 저장하고자 하는 항목명

Db.set(*item-name*, *value*, *timestamp*, *quality*, *type*)

데이터 저장소로부터 ***item-name*** 으로 지정된 위치에 ***value*** 로 지정된 데이터 값을 저장하기 위한 함수입니다.

- ***item-name*** : 저장하고자 하는 항목명
- ***value*** : 저장하고자 하는 값
- ***timestamp*** : timestamp 값은 "YYYY-MM-DD HH:mm:SS" 형식으로 지정되어야 하며, 지정되지 않은 경우에는 현재 날짜로 저장합니다.
- ***quality*** : 품질정보로 "bad", "good", "err" 값을 지정할 수 있으며, 지정되지 않으면 good 으로 지정됩니다.
- ***type*** : 저장하고자 하는 값의 타입을 지정하기 위한 항목으로 "bool", "word", "dword", "int", "real", "string" 지정이 가능하며 지정하지 않을 경우 "real"로 처리합니다.

예를 들어 날짜와 quality 정보를 생략하고 데이터형식만 지정하는 예의 경우

- Db.set("Test", 2.65, "", "", "int")

Test 라는 항목명으로 정수형태의 값이 저장되어 소수점을 버린 2 값이 저장됩니다.

Change of Value 기능을 제공하기 위하여 이전에 저장된 값이 반환되어 반환 값과 비교하면 새로운 값이 저장되었는지 확인 가능합니다.

Db.setModbus(*address*, *type*, *value*)

모드버스 통신을 위하여 ***address*** 로 지정된 모드버스 주소에 ***type*** 형식으로 변화하여 ***value*** 값을 저장하기 위한 함수입니다.

- ***address*** : 저장하고자 하는 지정 주소
- ***type*** : 저장하고자 하는 값의 타입.

- ***value**:* 저장하고자 하는 값.

5.3. 데이터변환을 위한 오브젝트

데이터저장소에 저장된 데이터를 가공하기 위한 기능으로 엔지니어링 단위 변환, 직선의 방정식을 이용한 보간 법, 각종 환경관련 인덱스 계산기능을 제공합니다.

엔지니어링 단위 변환

Conv.tempF(*섭씨온도*)

섭씨온도를 화씨온도로 변환하기 위한 함수

Conv.tempC(*화씨온도*)

화씨온도를 섭씨온도로 변환하기 위한 함수

Conv.tempK(*섭씨온도*)

섭씨온도를 절대온도로 변환하기 위한 함수

직선의 방정식을 이용한 보간 함수

Conv.norm(*x, x1, y1, x2, y2*)

두 지점 간의 직선식을 이용하여 특정위치의 값을 유추하기 위한 함수로 아래의 수학식을 이용하여 계산된 결과 값을 반환합니다.

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

각종 인덱스를 계산하기 위한 함수

Conv.thi(*섭씨온도, 상태습도*)

섭씨온도와 상대습도를 이용하여 불쾌지수를 계산하기 위한 함수

5.4. 데이터 포맷관리를 위한 함수

자바스크립트에서 제공되지 않는 자릿수 채우기 함수와 날짜 포맷함수를 제공합니다.

***Number*.pad(*length*)**

숫자 형식의 자료를 지정된 자리수의 문자열로 변환하기 위하여 사용되는 함수로 변환하고자 하는 숫자의 자리수가 ***length*** 보다 작은 경우에는 앞부분에 0 값으로 채워집니다.

```
123.pad(5)
456.pad(6)
```

[예시 1] pad()

위 예시의 값은 각각 "00123", "000456"으로 변환합니다.

***Date*.format(*format-string*)**

날짜 형식의 데이터를 주어진 포맷에 따라 변환하는 함수입니다.

```
var now = new Date(2025,1,25, 15, 10, 05);
now.format("yyyy-MM-dd HH:mm:ss"
```

[예시 2] Arr.get()

위 예시의 값은 현재 시각 값이 을 "2025-01-20 15:10:05"로 변환합니다. format 스트링에서 사용 가능한 문자는 다음과 같습니다.

포맷 스트링	설명
yyyy	4자리의 년도를 표시합니다.
yy	년도를 두 자리 형식으로 표시합니다.
MM	1~12월을 표시하며 한자릿수의 월인 경우 앞자리를 0으로 채웁니다.
dd	1~31일을 두 자리 수에 맞추어 표시합니다.
HH	0~23시 정보를 두 자리에 맞추어 표시합니다.
hh	1~12시 정보를 두 자리에 맞추어 표시합니다.
a/p	am or pm을 표시합니다.
mm	0~59분 정보를 두 자리에 맞추어 표시합니다.
ss	0~59초 정보를 두 자리에 맞추어 표시합니다.

[표 5] 날짜 포맷 함수

5.5. 배열관리를 위한 오브젝트

통신 드라이버를 통하여 수집된 바이트 배열로부터 데이터를 추출하기 위한 함수로 함수에 접근하기 위한 오브젝트명은 Arr 이며 값 추출을 위한 find(), get(), getAsc() 함수를 제공하고 있습니다.

Arr.find(*array* | *string*, *value*)

주어진 ***array*** 로부터 ***value*** 값을 찾아 위치를 반환합니다. 또는 첫번째 파라메터는 문자열로 입력될 수 있습니다.

```
// 주어진 배열에서 "ST"의 위치에 해당하는 2 값을 반환
Arr.find([0x30, 0x52, 0x53, 0x54, 0x30, 0x2E, 0x35], "ST")
// 주어진 문자열에서 "ST"의 위치에 해당하는 2 값을 반환
Arr.find("0RST0.005", "ST")
```

[예시 3] Arr.find()

Arr.get(frame-array, index, type)

주어진 ***frame-array*** 로부터 ***index*** 위치에서 ***type*** 형식의 값을 추출하기 위한 함수입니다.

```
// 통신프레임에서 1번째 위치의 값을 byte로 변환하여 반환하게 되므로 5 값을 반환
Arr.get([2,5,6,3,7,8], 1, "byte")
```

[예시 4] Arr.get()

Arr.getAsc(array | string, pos, cnt)

주어진 ***array*** 로부터 ***pos*** 위치부터 ***cnt*** 개수만큼의 문자열을 반환합니다.

```
// 주어진 배열의 4번째부터 3자리를 반환하면 "0.5" 값을 반환
Arr.getAsc([0x30, 0x52, 0x53, 0x54, 0x30, 0x2E, 0x35], 4, 3)
// 주어진 배열의 4번째부터 3자리를 반환하면 "0.3" 값을 반환
Arr.getAsc("0RST0.3", 4, 3)
```

[예시 5] Arr.getAsc()

get() 함수

Arr.get(*frame-array*, *index*, *type*)

주어진 통신프레임 배열에서 ***index*** 위치의 값을 ***type*** 으로 지정된 데이터 포맷으로 반환합니다.

- ***frame-array*** : 통신프레임으로 수신된 byte 배열구조.
- ***index*** : 통신 프레임에서 데이터를 취하고자 하는 인덱스(0부터 시작).
- ***type*** : 변환하고자 하는 데이터 포맷.

사용 가능한 형식의 데이터 포맷은 다음과 같습니다.

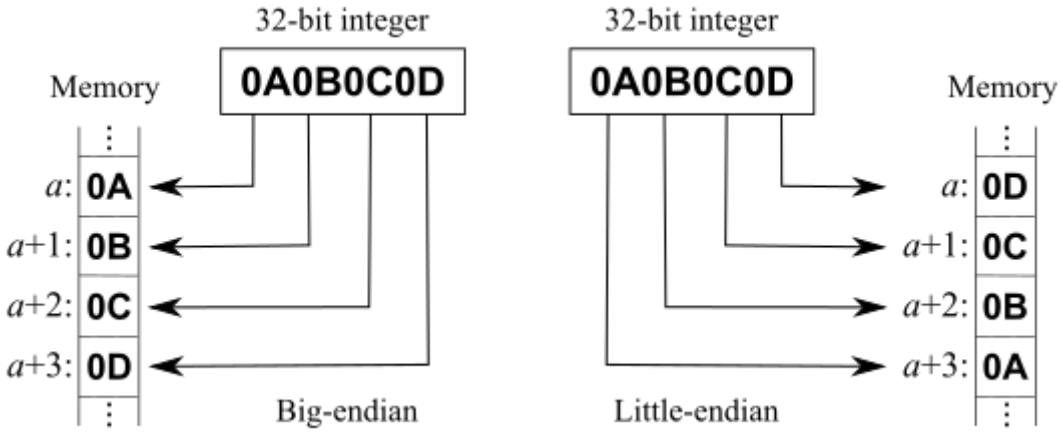
Type	설명
byte	1 byte 데이터이며 0~255까지의 값으로 변환됩니다.
word	2 bytes 데이터이며 little endian 형식의 정수형으로 변환됩니다. 만일 0x01 0x02 값이 저장되어 있으면 513 값이 반환됩니다.
word1	2 bytes 데이터이며 big endian 형식의 정수형으로 변환됩니다. 만일 0x01 0x02 값이 저장되어 있으면 258이 반환됩니다.
int	4 bytes로 구성된 데이터이며 little endian 형식의 4 bytes 정수 값이 반환됩니다.
int1	4 bytes로 구성된 데이터이며 big endian 형식의 4 bytes 정수 값이 반환됩니다.
long	8 bytes로 구성된 데이터이며 little endian 형식의 8 bytes 정수 값이 반환됩니다.
long1	8 bytes로 구성된 데이터이며 big endian 형식의 8 bytes 정수 값이 반환됩니다.
float	4 bytes로 구성된 데이터이며 little endian 형식의 4 bytes 실수 값이 반환됩니다.
float1	4 bytes로 구성된 데이터이며 big endian 형식의 4 bytes 실수 값이 반환됩니다.
real	8 bytes로 구성된 데이터이며 little endian 형식의 8 bytes 실수 값이 반환됩니다.
real1	8 bytes로 구성된 데이터이며 big endian 형식의 8 bytes 실수 값이 반환됩니다.

[표 6] Type 종류 및 설명

엔디언(Endian)

엔디언(Endianness)은 컴퓨터의 메모리와 같은 1 차원의 공간에 여러 개의 연속된 대상을 배열하는 방법을 뜻하며, 바이트를 배열하는 방법을 특히 바이트 순서(Byte order)라 합니다.

엔디언은 보통 큰 단위가 앞에 나오는 빅 엔디언(Big-endian)과 작은 단위가 앞에 나오는 리틀 엔디언(Little-endian)으로 나눌 수 있으며, 두 경우에 속하지 않거나 둘을 모두 지원하는 것을 미들 엔디언(Middle-endian)이라 부르기도 합니다.



[그림 20] Big-endian vs. Little-Endian

5.6. 로그관리를 위한 오브젝트

설비와 연결된 통신드라이버가 정상적으로 수행하는지 디버깅하기 위한 Log 이름의 오브젝트를 제공하며 add 함수를 제공하고 있습니다.

Log.add(*message*)

message 의 내용을 웹관리자 화면에 표시하기 위한 함수로 통신드라이버로부터 수집된 데이터를 디버깅하기 위한 함수입니다.

5.7. 시스템관리를 위한 오브젝트

시스템의 상태정보를 취득하기 위한 오브젝트로 System 이름의 오브젝트를 제공합니다.

System.isActive()

이중화로 구성된 시스템에서 현재 상태가 active 인 경우에는 True 값을, inactive 인 경우에는 false 값을 반환합니다.

System.sleep(*msec*)

일정시간 동안 스크립트 동작을 정지시키기 위한 함수로 지정된 밀리 세컨드 초만큼 스크립트를 중단한 후 재 기동합니다.

6. 드라이버별 자바스크립트 오브젝트

통신용 드라이버는 필요시 로드 하여 사용하거나 불필요할 경우 언 로드 할 수 있는 오브젝트로 프로토콜별/역할별로 드라이버를 지원합니다.

6.1. 범용 드라이버

범용 드라이버는 MES, ERP, 데이터베이스 관리 시스템 등과 같은 상위 시스템과의 인터페이스를 위하여 제공되는 JDBC, Restful API 등과 같은 드라이버입니다.

Jdbc

Java Ver. 8 기반의 JDBC 를 지원하기 위한 드라이버로 JDBC 드라이버가 제공되는 드라이버의 연결 및 데이터 조회, SQL 문장 실행이 가능합니다.

Rest

Restful API Server 로부터 데이터를 수집하기 위한 드라이버로 Restful 표준 통신함수인 GET, POST 를 지원합니다. 또한 Restful API 는 보안이 설정되지 않은 HTTP Server 는 물론 보안이 강화된 HTTPS 서버로부터의 데이터 취득 및 전송이 가능합니다.

6.1.1. Jdbc

JDBC 기반의 데이터베이스 연결을 위한 드라이버로 JDBC 드라이버가 제공되는 데이터베이스와의 연결을 가능하도록 하기 위한 드라이버입니다.

Jdbc.open(*connection-string*, *user-name*, *password*)

- ***connection-string*** **:** Jdbc 드라이버를 이용하여 데이터베이스에 연결하기 위한 문자열로 연결하고자 하는 데이터베이스에서 지정한 문자열 정보입니다. 연결이 성공하면 0 보다 큰 값의 세션 값이 반환되며 이 세션 값을 이용하여 데이터베이스로부터 자료를 조회하거나 SQL 문장을 실행할 수 있습니다. 마이크로 소프트사의 SQL 서버에서 제공하는 문자열의 예는 다음과 같습니다.

```
jdbc:sqlserver://192.168.1.115;encrypt=false
```

[예시 6] Jdbc.open()

- ***user-name*** : 데이터베이스 연결에 사용되는 사용자 이름을 지정합니다.
- ***password*** : 데이터베이스 연결에 사용되는 사용자이름을 위한 사용자 암호를 지정합니다.

Jdbc.query(*sess*, *select-statement*)

데이터베이스로부터 자료를 조회하기 위한 함수로 조회 결과는 오브젝트 배열형식으로 변환되므로 자바스크립트에서 오브젝트 형식으로 접근이 가능합니다.

- ***sess*** : open 함수에 반환된 세션 번호.
- ***select-statement*** : 데이터베이스로부터 데이터를 조회하기 위한 SQL 문장.

Jdbc.exec(*sess*, *insert(update)-statement*)

데이터베이스에 자료를 저장하거나, 갱신하고자 할 경우 사용되는 함수입니다. 반환 결과는 성공과 실패 정보밖에 없으므로 select 와 같은 SQL 문장은 사용이 불가합니다.

- ***sess*** : open 함수에 반환된 세션 번호.
- ***insert(Update)-statement*** : 데이터베이스로부터 데이터를 처리하기 위한 SQL 문장.

SQL 서버와 연결하여 데이터를 추가하고 조회하기 위한 예제는 다음과 같습니다.

```
Lib.load("jdbc");
var url = "jdbc:sqlserver://192.168.1.115;encrypt=false";
var sess = Jdbc.open(url, "pims", "pims");
if(typeof(sess) != 'undefined') {
    var val = Math.random();
    Jdbc.exec(sess, "INSERT INTO [BR_PIMS].[dbo].[T_RANDOM] (F_DATETIME,  F_VAL)
VALUES('2024-05-24 12:00:00', " + val + ")");
    var ret = Jdbc.query(sess, "SELECT *  FROM [BR_PIMS].[dbo].[T_RANDOM]");
   Log.add(ret);
}
```

[예시 7] Jdbc Driver

6.1.2. Rest

Restful API 를 지원하는 서버로부터 데이터를 취득하거나 계산된 결과를 전송하기 위한 드라이버로 라이브러리 관리자로부터 드라이버가 로드되면 Rest 오브젝트가 생성되며 이를 이용하여 get, post 함수를 수행할 수 있습니다.

Rest.get(*url*)

- ***url*** : Restful API 가 지원되는 서버로 전송되는 URL 이며 성공적으로 수행하면 결과를 스트링 또는 JSON 오브젝트로 반환합니다. 만일 반환된 결과가 Json 형식이면

Json 오브젝트로 변환되며, 그 이외의 경우에는 스트링 형식으로 반환됩니다. 아래의 예시는 공공데이터 포털로부터 관측소 기상정보를 가져오기 위한 URL 예시입니다.

```
http://apis.data.go.kr/1360000/AsosHourlyInfoService/getWthrDataList?serviceKey=공공데이터포털서비스키&numOfRows=10&pageNo=1&dataCd=ASOS&dateCd=HR&stnIds=108&endDt=20200310&endHh=01&startHh=01&startDt=20190120
```

[예시 8] Jdbc.open()

Rest.pos(*url*, *data*)

데이터베이스로부터 자료를 조회하기 위한 함수로 조회 결과는 오브젝트 배열형식으로 변환되므로 자바스크립트에서 오브젝트 형식으로 접근이 가능합니다.

- ***url*** : Restful API 가 지원되는 서버로 전송되는 URL 이며 성공적으로 수행하면 결과를 스트링 또는 JSON 오브젝트로 반환합니다. 만일 반환된 결과가 Json 형식이면 Json 오브젝트로 변환되며, 그 이외의 경우에는 스트링 형식으로 반환됩니다.
- ***data*** : Post 하고자 하는 데이터를 지정하는 항목으로 스트링 값으로 전송되어야 하므로 Json 오브젝트를 전송하고자 할 경우에는 JSON.Stringify()함수를 이용하여 스트링으로 변환하여 전달합니다.

6.2. 표준 통신드라이버

표준 통신드라이버는 지정된 프로토콜 없이 시리얼 포트 혹은 이더넷 포트를 통하여 통신프레임을 편집할 수 있는 기능을 제공하는 드라이버로 연결하고자 하는 디바이스에 따라 serial, ethernet 드라이버로 분류할 수 있습니다.

Serial

시리얼 포트를 이용하여 통신 프레임을 구성하여 전송하거나 수신된 프레임을 파싱 한 후 값을 추출하기 위한 드라이버.

TCP Client

TCP 클라이언트를 이용하여 통신 프레임을 구성하여 전송하거나 수신된 프레임을 파싱 한 후 값을 추출하기 위한 드라이버.

TCP Server

TCP 서버를 이용하여 통신 프레임을 구성하여 전송하거나 수신된 프레임을 파싱 한 후 값을 추출하기 위한 드라이버.

6.2.1. Serial

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 Serial 로 지정된 오브젝트명을 사용할 수 있으며 해당 오브젝트명을 이용하여 모드버스를 지원하는 설비와 연결, 데이터 교환을 가능하게 합니다.

Serial.open(*port-name*, *communication-param*)

port-name 으로 지정된 시리얼 포트를 오픈한 다음 ***communication-param*** 으로 지정된 설정 내용을 적용합니다. ***communication-param*** 은 통신속도, 패리티 등과 같은 정보를 문자열로 구성되어 있습니다. 함수를 호출하면 session id 가 반환되면 반환된 session id 를 이용하여 데이터를 읽거나 쓰기를 할 수 있습니다.

- ***port-name*** : 연결하고자 하는 통신 포트.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다.

사용 예는 다음과 같습니다.

```
baud=2400 databit=8 parity=none
baud=9600 databit=8 parity=odd
baud=19200 databit=8 parity=even
```

[예시 9] Serial.open(): Communication-param

Serial.read(*sess*, *type*, ..)

시리얼 포트로부터 값을 읽어옵니다.

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***type*** : 프레임을 해석하고자 하는 파서를 지정하는 항목으로 지정된 파서에 따라 정의되는 항목이 다릅니다. 파서에 따라 정의되는 항목은 아래 "파서 및 항목 정의" 그림을 참조합니다.

```
// 수신된 프레임을 해석하지 않고 전체를 배열로 반환
var data = Serial.read(sess, "type=none")

// 수신된 프레임을 stxLen 파서로 이용하여 해석하고 배열로 반환, stx는 0x7E이며 수신 프레임의 3번째를 2 bytes의 길이로 변환하여 7을 더한 프레임을 반환
var data = Serial.read(sess, "type=stxLen stx=0x7E len-3:7:word1")

// 수신된 프레임을 stxEtx 파서를 이용하여 해석하고 배열로 반환, 프레임은 0x02로 시작하여 0x03으로 종료되며 0x02와 0x03 값이 포함된 프레임을 배열로 반환
var data = Serial.read(sess, "type=stxEtx stx=0x02 etx=0x03)

// 수신된 프레임을 len 파서를 이용하여 해석하고 배열로 반환, 0번째서 2 bytes의 길이만큼 프레임을 배열로 변환하여 반환
var data = Serial.read(sess. "type=len len=0:0:word1")
```

[예시 10] Serial Driver

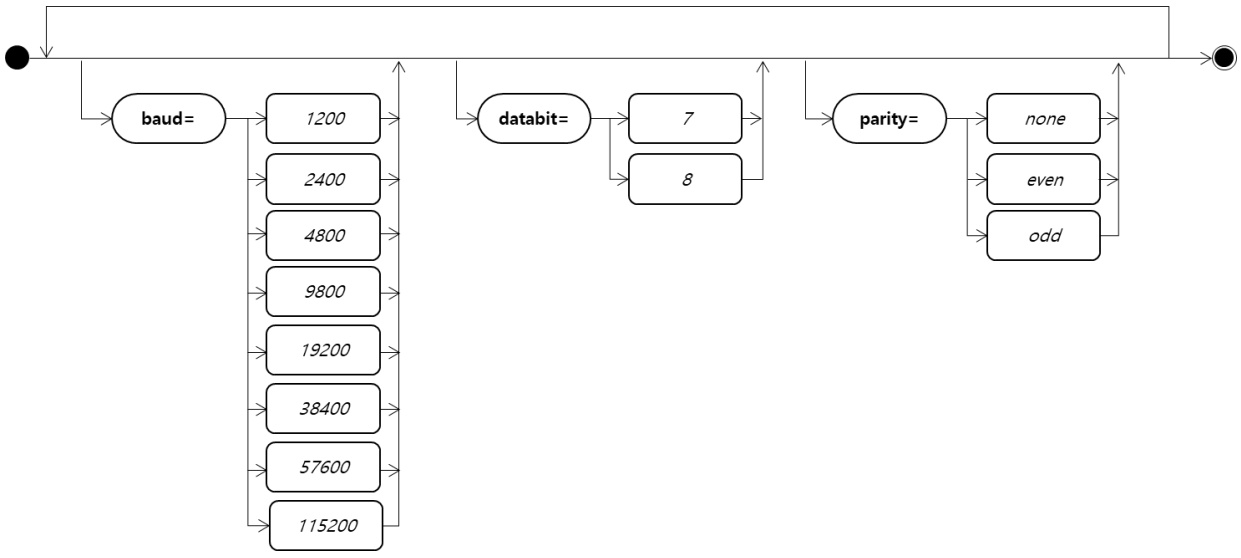
Serial.write(*sess*, *frame*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***frame*** : 전송하고자 하는 프레임을 지정. 프레임은 배열을 이용하여 함수로 전달합니다.

Serial.wait(*msec*)

- ***msec*** : 지정된 시간만큼 기다립니다. 최대 10,000msec(10 초)까지 대기할 수 있습니다. 데이터 요청 후 모든 프레임을 수신할 때까지 대기하고자 할 경우 사용합니다.

시리얼 포트를 오픈하기 위한 통신 파라메터 구성은 다음과 같습니다.



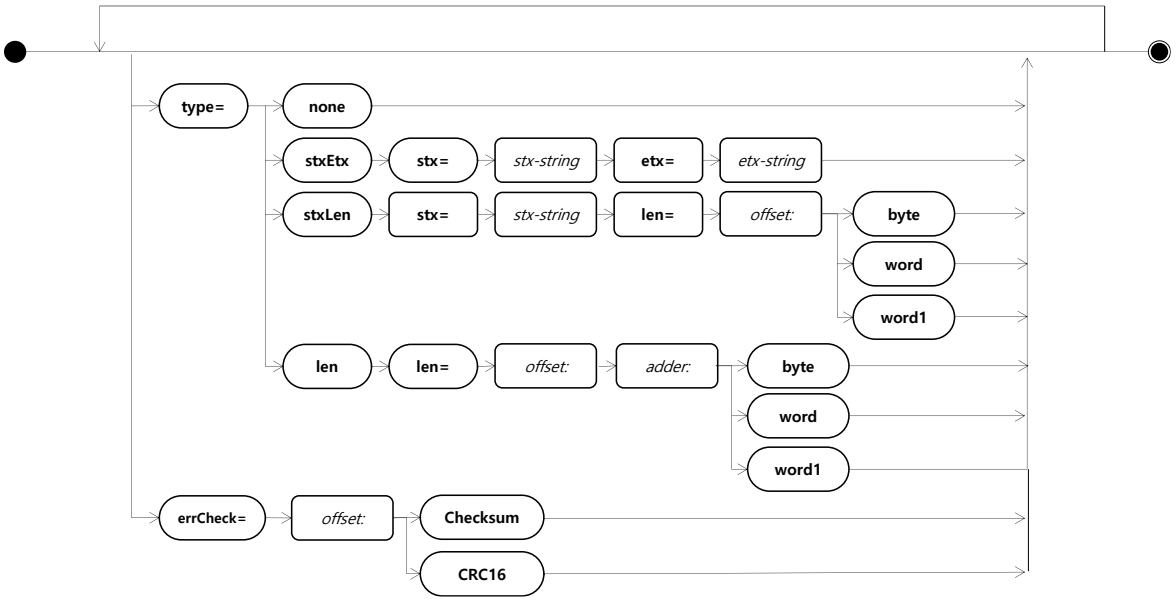
[그림 21] 시리얼 통신 파라메터

프레임을 해석하는 파서의 종류는 다음표와 같으며 지정된 파서에서 필요한 항목은 그림을 참고하시면 됩니다.

파서 종류	설명
none	지정된 파서가 없습니다. 수신된 프레임을 배열로 변환하여 반환합니다.
stxEtx	STX와 ETX로 구성된 프레임을 해석하여 전체 프레임을 배열로 변환 후 반환합니다.
stxLen	STX와 LEN으로 구성된 프레임을 해석하여 해석된 프레임을 배열로 변환 후 반환합니다.
len	프레임 길이만으로 수신 프레임을 해석하여 해석된 프레임을 배열로 변환 후 반환합니다.

[표 7] 파서 종류 및 설명

파서에 따라 지정하여야 하는 항목은 다음그림과 같습니다.



[그림 22] 파서 및 항목정의

6.3. Modbus 통신드라이버

모드버스의 경우 통신하고자 하는 채널에 따라 Modbus/RTU, Modbus/TCP 로 분류가 가능하고 지원하는 역할에 따라 Modbus/slave, Modbus/master, Modbus/client, Modbus/server 분류할 수 있습니다.

mbmaster

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBmaster 로 지정된 오브젝트 명을 사용할 수 있으며, 해당 오브젝트 명을 이용하여 모드버스를 지원하는 설비와 연결, 데이터 교환이 가능하게 합니다.

mbslave

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBslave 로 지정된 오브젝트 명을 사용할 수 있으며, 해당 오브젝트 명을 이용하여 Modbus/RTU master 설비와의 데이터 교환을 가능하게 합니다.

mbclient

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBclient 로 지정된 오브젝트 명을 사용할 수 있으며, 해당 오브젝트 명을 이용하여 Modbus/TCP server 설비와의 데이터 교환을 가능하게 합니다.

mbserver

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBserver 로 지정된 오브젝트 명을 사용할 수 있으며, 해당 오브젝트 명을 이용하여 Modbus/TCP client 설비와의 데이터 교환을 가능하게 합니다.

mbrtutcp

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBrtutcp 로 지정된 오브젝트 명을 사용할 수 있으며, 해당 드라이버가 로드 되고 설비와 연결되면 Modbus/RTU master 를 이용하여 Modbus/TCP server 와의 데이터 교환을 가능하게 합니다.

mbtcprtu

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBtcprtu 로 지정된 오브젝트 명을 사용할 수 있으며, 해당 드라이버가 로드되고 설비와 연결되면 Modbus/TCP client 를 이용하여 Modbus/RTC slave 와의 데이터 교환을 가능하게 합니다.

6.3.1. mbmaster

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBmaster 로 지정된 오브젝트명을 사용할 수 있으며 해당 오브젝트명을 이용하여 모드버스를 지원하는 설비와 연결, 데이터 교환을 가능하게 합니다.

MBmaster.open(*port-name*, *communication-param*)

port-name 으로 지정된 시리얼 포트를 오픈한 다음 ***communication-param*** 으로 지정된 설정 내용을 적용합니다. ***communication-param*** 은 통신속도, 패리티 등과 같은 정보를 문자열로 구성되어 있습니다. 함수를 호출하면 session ID 가 반환되며, 반환된 session ID 를 이용하여 데이터를 읽거나 쓰기를 할 수 있습니다.

- ***port-name*** : 연결하고자 하는 통신포트.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다. 사용 예는 다음과 같습니다.

```
baud=2400 databit=8 parity=none
baud=9600 databit=8 parity=odd
baud=19200 databit=8 parity=even
```

[예시 11] MBmaster.open(): Communication-param

MBmaster.read(*sess*, *slave*, *addr*, *cnt*, *[type*, *endian]*)

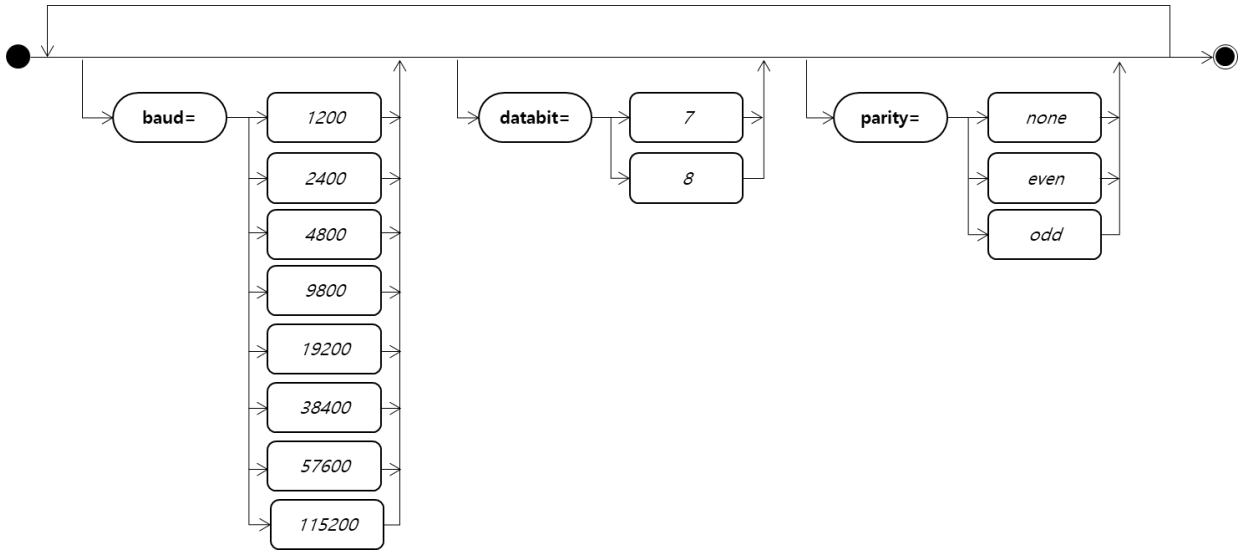
- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***slave*** : 연결하고자 하는 설비의 slave 번호.
- ***addr*** : 모드버스 통신을 위한 주소로 1 부터 49999 까지의 주소 영역을 사용합니다.
- ***cnt*** : 지정된 주소로부터 읽어오기 위한 워드(2 bytes) 개수.
- ***type*** : 지정된 주소로부터 읽어온 값을 ***type*** 로 지정된 데이터 형식으로 변환하여 반환합니다. 지정하지 않으면 2 bytes 의 데이터를 반환하며 "int"로 지정하면 4 bytes 크기의 정수 값을 반환하며, "float"로 지정하면 4 bytes 값의 실수 값을 반환합니다.

- ***endian*** : 바이트 배열을 지정하는 값으로 지정하지 않으면 모드버스 규격에서 지원하는 방식으로 변환하며, "big"으로 지정하면 big-endian 방식, 즉 인텔 CPU 가 채택한 방식으로 데이터를 변환하여 반환합니다.

MBmaster.write(*sess*, *slave*, *func*, *addr*, *values*, *[type*, *endian]*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***slave*** : 연결하고자 하는 설비의 slave 번호.
- ***func*** : 모드버스 통신을 위한 함수로 5, 6, 15, 16 중 하나의 함수코드를 사용합니다.
- ***addr*** : 모드버스 통신을 위한 주소로 1 부터 9999 까지의 주소 영역을 사용합니다.
- ***values*** : 배열로 전달되야 하며 ***func***에 지정된 함수 코드에 따라 하나의 값 혹은 여러 개의 값을 통신에 전송할 수 있습니다.
- ***type*** : ***type***으로 지정된 데이터 형식으로 변환하여 전송합니다. 지정하지 않으면 2 bytes 의 데이터를 반환하며 "int"로 지정하면 4 bytes 크기의 정수 값을 반환하며, "float"로 지정하면 4 bytes 값의 실수 값으로 변환하여 전송합니다.
- ***endian*** : byte 배열을 지정하는 값으로 지정하지 않으면 모드버스 규격에서 지원하는 방식으로 변환하여 전송하며, "big"으로 지정하면 big-endian 방식, 즉 인텔 CPU 가 채택한 방식으로 데이터를 변환하고 반환하여 전송합니다.

시리얼 포트를 오픈하기 위한 통신 파라메터 구성은 다음과 같습니다.



[그림 23] 시리얼 통신 파라메터

6.3.2. mbslave

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBslave 로 지정된 오브젝트명을 사용할 수 있으며 해당 오브젝트명을 이용하여 Modbus/RTU master 설비와의 데이터 교환을 가능하게 합니다.

MBslave.open(*port-name*, *communication-param*)

port-name 으로 지정된 시리얼 포트를 오픈한 다음 ***communication-param*** 으로 지정된 설정 내용을 적용합니다. ***communication-param*** 은 통신속도, 패리티 등과 같은 정보를 문자열로 구성되어 있습니다. 함수를 호출하면 session ID 가 반환되며 반환된 session ID 를 이용하여 데이터를 읽거나 쓰기를 할 수 있습니다.

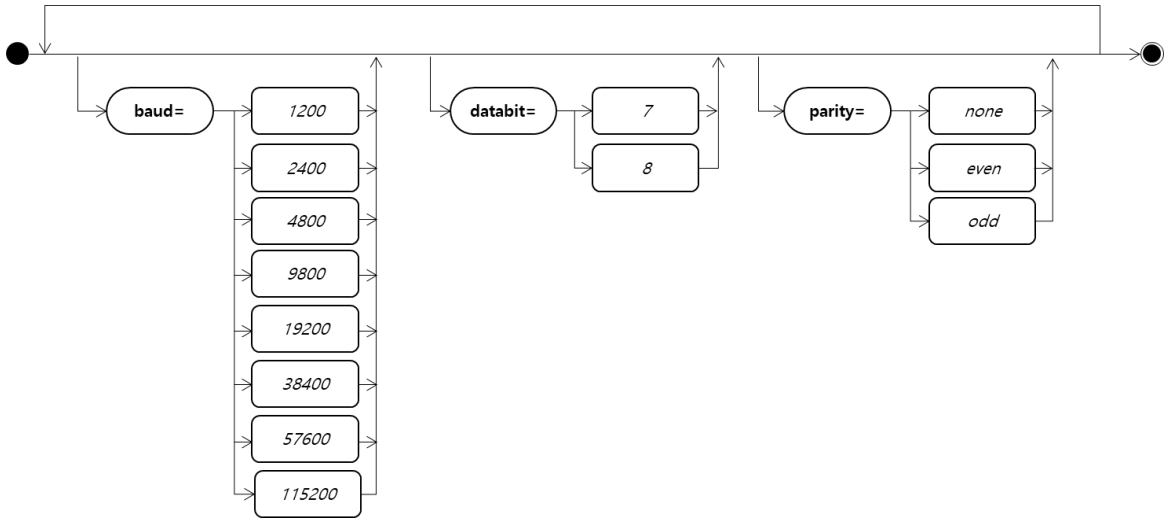
- ***port-name*** : 연결하고자 하는 통신포트
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다.

사용 예는 다음과 같습니다.

```
baud=2400 databit=8 parity=none
baud=9600 databit=8 parity=odd
baud=19200 databit=8 parity=even
```

[예시 12] MBslave.open(): communication-param

시리얼 포트를 오픈하기 위한 통신 파라메터 구성은 다음과 같습니다.



[그림 24] 시리얼 통신 파라메터

6.3.3. mbclient

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBclient 로 지정된 오브젝트명을 사용할 수 있으며 해당 오브젝트명을 이용하여 Modbus/TCP server 설비와의 데이터 교환을 가능하게 합니다.

MBclient.open(*ip-addr* , *port-no*)

ip-addr 와 ***port-no*** 를 이용하여 설비와 연동합니다.

- ***ip-addr*** : 연결하고자 하는 IP 주소.
- ***port-no* :** 소켓 통신에 사용되는 포트 번호.

MBclient.read(*sess*, *addr*, *cnt*, *[type*, *endian]*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***addr* :** 모드버스 통신을 위한 주소로 1 부터 49999 까지의 주소 영역을 사용합니다.
- ***cnt*** : 지정된 주소로부터 읽어오기 위한 워드(2 bytes) 개수
- ***type*** : 지정된 주소로부터 읽어 온 값을 ***type*** 로 지정된 데이터 형식으로 변환하여 반환합니다. 지정되지 않으면 2 bytes 의 데이터를 반환하며 "int"로 지정하면 4 bytes 크기의 정수 값을 반환하며, "float"로 지정하면 4 bytes 값의 실수 값을 반환합니다.

- ***endian*** : 바이트 배열을 지정하는 값으로 지정하지 않으면 모드버스 규격에서 지원하는 방식으로 변환하며, "big"으로 지정하면 big-endian 방식, 즉 인텔 CPU 가 채택한 방식으로 데이터를 변환하여 반환합니다.

MBclient.write(*sess*, *func*, *addr*, *values*, *[type*, *endian]*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***func* :** 모드버스 통신을 위한 함수로 5, 6, 15, 16 중 하나의 함수코드를 사용합니다.
- ***addr*** : 모드버스 통신을 위한 주소로 1 부터 9999 까지의 주소 영역을 사용합니다.
- ***values*** : 배열로 전달되어야 하며 ***func*** 에 지정된 함수코드에 따라 하나의 값, 혹은 여러 개의 값을 통신에 전송할 수 있습니다.
- ***type*** : ***type*** 으로 지정된 데이터 형식으로 변환하여 전송합니다. 지정하지 않으면 2 bytes 의 데이터를 반환하며 "int"로 지정하면 4 bytes 크기의 정수 값을 반환하며, "float"로 지정하면 4 bytes 값의 실수 값으로 변환하여 전송합니다.
- ***endian*** : 바이트 배열을 지정하는 값으로 지정하지 않으면 모드버스 규격에서 지원하는 방식으로 변환하며 정송하며, "big"으로 지정하면 big-endian 방식, 즉 인텔 CPU 가 채택한 방식으로 데이터를 변환하고 반환하여 전송합니다.

MBclient.mapTo(sess, from, to)

from 으로 입력된 태그를 ***to*** 에 지정된 modbus 주소로 변환하여 장치로 전송합니다.

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***from*** : 태그명
- ***to*** : 모드버스 주소

MBclient.mapTo(sess, "IO-001", 1")으로 지정된 경우 IO-001 태그 값이 "outValues"명령어로 전송되면 모드버스 주소 1 번으로 수신된 값을 전송합니다.

6.3.4. mbserver

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBserver 로 지정된 오브젝트명을 사용할 수 있으며, 해당 오브젝트명을 이용하여 Modbus/TCP client 설비와의 데이터 교환을 가능하게 합니다.

MBserver.open(*port-no*, *use-repo*)

port-no 를 이용하여 설비와 연동합니다.

- ***port-no*** : 소켓 통신에 사용되는 포트번호.
- ***use-repo*** : true or false 값이 지정되며 기본값은 false

use-repo 는 Modbus write 명령이 수신되면 repository 에 태그 형식으로 수신된 값을 저장하는 기능으로 이중화를 통하여 두 게이트웨이 간의 데이터를 공유하고자 할 경우 반드시 true 로 지정되어야 합니다.

6.3.5. mbrtutcp

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBrtutcp 로 지정된 오브젝트명을 사용할 수 있으며, 해당 드라이버가 로드되고 설비와 연결되면 modbus/RTU master 를 이용하여 modbus/TCP server 와의 데이터 교환을 가능하게 합니다.

[그림 25] Modbus/RTU-TCP 게이트웨이

mbtrutcp 드라이버는 시리얼 통신망으로 연계된 Modbus/RTU master 에서 이더넷으로 연결된 Modbus/TCP Server 의 데이터를 연계하기 위하여 사용합니다.

MBrtutcp.openRtu(*port-name*, *communication-param*)

port-name 으로 지정된 시리얼 포트를 오픈한 다음 ***communication-param*** 으로 지정된 설정 내용을 적용합니다. ***communication-param*** 은 통신속도, 패리티 등과 같은 정보를 문자열로 구성되어 있습니다.

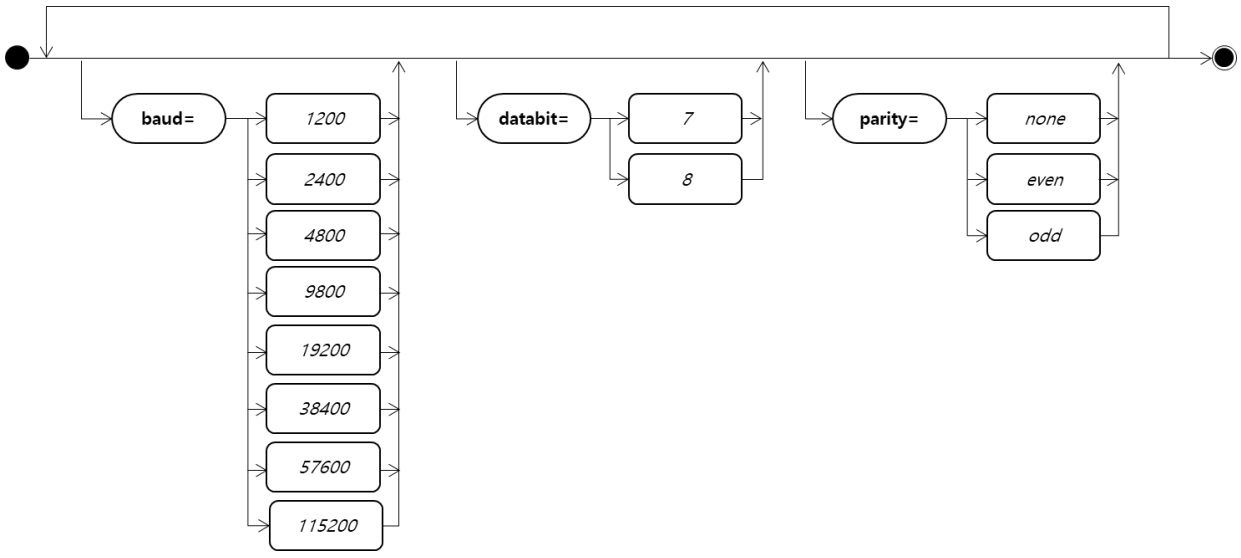
- ***port-name*** : 연결하고자 하는 통신포트.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다.

사용 예는 다음과 같습니다.

```
baud=2400 databit=8 parity=none
baud=9600 databit=8 parity=odd
baud=19200 databit=8 parity=even
```

[예시 13] MBrtutcp.openRtu(): communication-param

시리얼 포트(openRtu)를 오픈하기 위한 통신 파라메터 구성은 다음과 같습니다.



[그림 26] 시리얼 통신 파라메터

MBrtutcp.openRtu(*port-no, baud-rate*)

port-no 로 지정된 시리얼 포트를 오픈한 다음 ***baud-rate*** 로 지정된 값으로 통신속도를 설정합니다. 통신속도 외의 기본값(databit=8, stop=1, parity=none)으로 설정됩니다.

- ***port-no*** : 연결하고자 하는 통신 포트번호로 1 부터 시작합니다. 1 값이 입력되면 내부적으로 "/dev/ttyUSB0"으로 변환되어 시리얼 포트를 오픈하게 됩니다.
- ***baud-rate*** : 통신속도를 지정합니다. 그 외의 나머지 값들은 기본값(databit=8, stop=1, parity=none)으로 설정됩니다. 예를 들어 9600 값이 입력되면 "baud=9600 databit=8 parity=none"으로 변환되어 시리얼 포트를 오픈합니다.

MBrtutcp.openTcp(*ip-addr, port-no*)

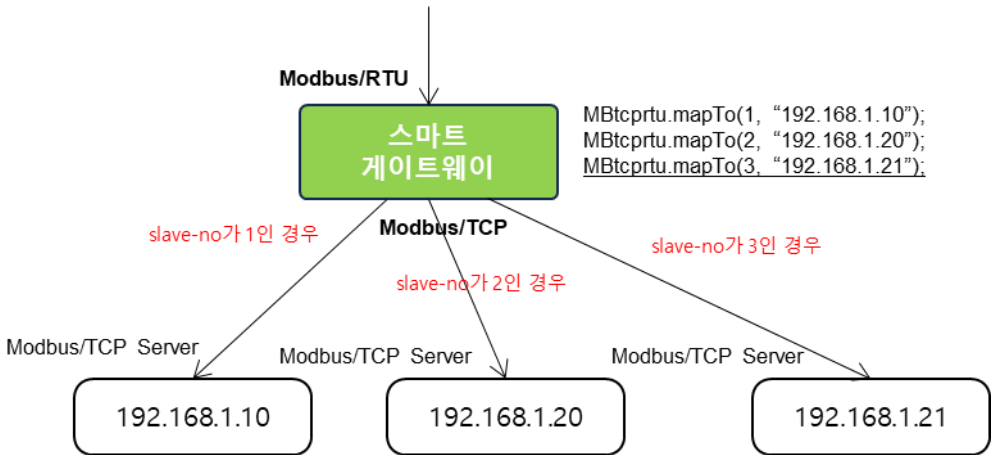
IP 주소와 Port 번호를 이용하여 Modbus Server 와의 연결을 합니다.

- ***ip-addr*** : 연결하고자 하는 Modbus/TCP Server IP 주소.
- ***port-no*** : 연결하고자 하는 Modbus/TCP port 번호로 주로 502 를 사용합니다.

MBrtutcp.mapTo(*slave-no, ip-address*)

slave-no 와 ***ip*** 주소를 연계하는 기능으로, MBrtutcp(1, "192.168.1.50")으로 설정된 경우 1 번으로 요청되는 메시지는 "192.168.1.50" 모드버스서버로 연결하여 데이터를 취득하거나 데이터를 전달하기 위함입니다.

- ***slave-no*** : 전달하고자 하는 Modbus/RTU Slave 번호.
- ***ip-address*** : 연결하고자 하는 Modbus/TCP 서버 주소



[그림 27] slave 번호에 따른 Modbus Server 연계

6.3.6. mbtcprtu

라이브러리 관리자를 이용하여 모드버스 통신 드라이버를 로드 하면 MBtcprtu 로 지정된 오브젝트명을 사용할 수 있으며, 해당 드라이버가 로드되고 설비와 연결되면 modbus/TCP client 를 이용하여 modbus/RTU slave 와의 데이터교환을 가능하게 합니다.

[그림 28] Modbus/TCP-RTU 게이트웨이

mbtrutcp 드라이버는 시리얼 통신망으로 연계된 Modbus/RTU master 에서 이더넷으로 연결된 Modbus/TCP Server 의 데이터를 연계하기 위하여 사용합니다.

MBrtutcp.openRtu(*port-name*, *communication-param*, *slave-no1*, *slave-no2*)

port-name 으로 지정된 시리얼포트를 오픈한 다음 ***communication-param*** 으로 지정된 설정 내용을 적용합니다. ***communication-param*** 은 통신속도, 패리티 등과 같은 정보를 문자열로 구성되어 있습니다.

- ***port-name*** : 연결하고자 하는 통신포트.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다.

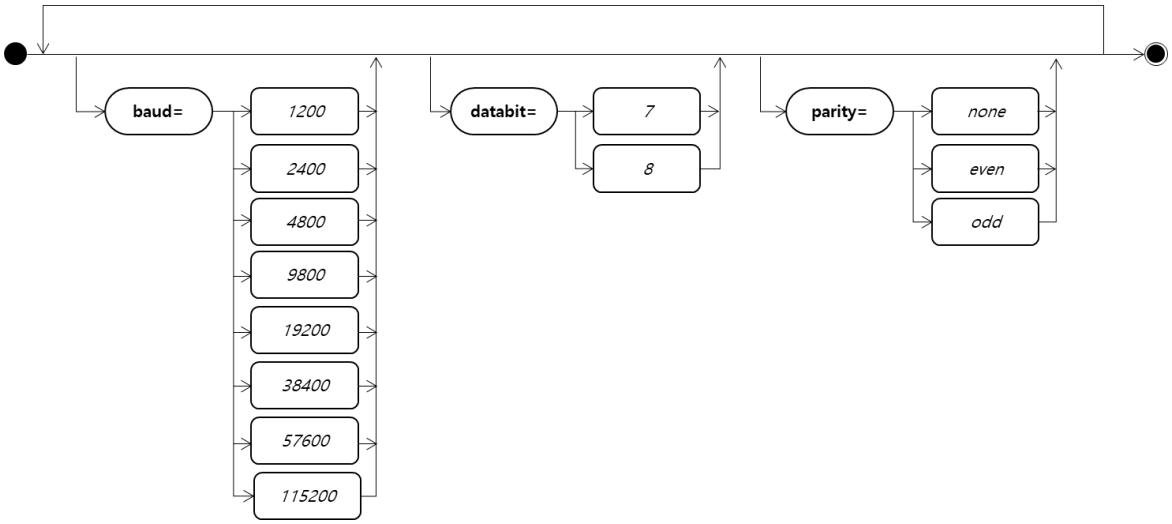
```
baud=2400 databit=8 parity=none
baud=9600 databit=8 parity=odd
baud=19200 databit=8 parity=even
```

[예시 14] MBrtutcp.openRtu(): communication-param

- ***slave-no1*** : 연결하고자 하는 Modbus/RTU 의 ***slave*** 번호로 지정하지 않으면 1 로 고정됩니다.

- ***slave-no2*** : 연결하고자 하는 Modbus/RTU 의 ***slave*** 번호로 ***slave-no1*** 이 지정되어 있으면 ***slave-no*** 와 동일한 번호가 ***slave-no1*** 도 지정되지 않았으면 254 로 고정됩니다.

시리얼 포트(openRtu)를 오픈하기 위한 통신 파라메터 구성은 다음과 같습니다.



[그림 29] 시리얼 통신 파라메터

MBrtutcp.openTcp(*port-no*)

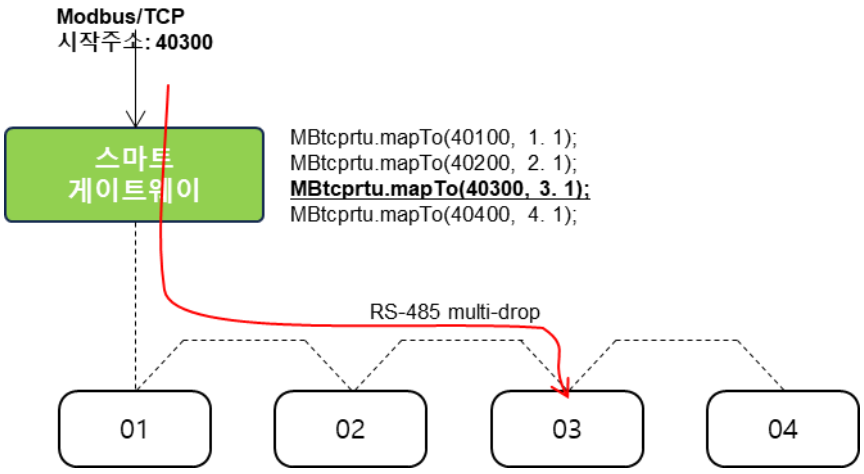
- ***port-no*** : 연결하고자 하는 Modbus/TCP port 번호로 주로 502 를 사용합니다.

MBrtutcp.mapTo(*addr1*, *slave-no*, *addr2*)

- ***addr1*** : 변환시작 주소
- ***slave-no*** : ***addr1*** 으로 요청되면 ***slave*** 번호를 지정된 ***slave*** 번호로 변환합니다.
- ***addr2*** : ***addr1*** 으로 요청하면 ***addr2*** 로 요청주소를 변환합니다.

Modbus/TCP 로 요청된 프레임을 ***slave*** 번호와 요청주소를 변경하고자 할 경우 사용합니다.

예를 들어 아래 그림과 같이 Modbus/RTU 가 Multi-drop 방식으로 구성된 경우



[그림 30] Modbus TCP-RTU 다중변환

Modbus/TCP 프로토콜을 이용하여 40100 번지의 값을 요청하면 스마트 게이트웨이는 1 번째 Slave Modbus/RTU 장치의 40001 위치의 값을 읽어 Modbus/TCP 로 변환합니다. 만일 Modbus/TCP 프로토콜을 이용하여 40300 번지의 값을 요청하면 스마트 게이트웨이는 3 번째 Slave Modbus/RTU 장치의 40001 위치의 값을 읽어 Modbus/TCP 로 변환합니다.

6.4. OPC UA 통신드라이버

6.4.1. uaclient

uaclient 드라이버는 OPC UA 서버와의 통신을 위한 드라이버로 OPC UA 서버로부터 데이터를 읽거나 쓰기 위한 통신 드라이버입니다.

OPC UA 는 NodeID 라고 하는 구조체를 이용하여 데이터통신을 하며 Node ID 구성은 다음과 같이 이루어져 있습니다.

- {N|S}nnn|{numeric|string|

예를 들면

- N1|3002 => Numeric 타입의 Namespace 1 번에 3002 번째 아이템
- S1|temp => String 타입의 Namespace 1 번에 temp 라는 라벨로 정해진 아이템

UAclient.addItem(*node-id*)

데이터를 읽어오기 위한 아이템(***node-id***로 구성)을 등록하기 위한 함수입니다. additem() 함수는 반드시 open() 함수를 호출하기 전에 호출되어야 합니다.

- ***node-id*** : 데이터를 읽어오기 위한 아이템 명으로 ***node-id***로 구성됩니다.

UAclient.open(*URL*)

URL로 지정된 OPC UA 서버와의 연결을 위한 함수입니다.

- ***URL*** : OPC UA 서버 address로 다음과 같은 형식으로 구성됩니다.

```
opc.tcp://192.168.1.135:53530/OPCUA/SimulationServer
```

[예시 15] UAclient.open()

UAclient.read()

addItem() 함수를 이용하여 추가된 아이템의 값을 읽어오기 위한 함수입니다.

아이템별 값, 이름의 배열로 반환됩니다.

```
[{"val":-1.628655, "name":"N3|1002"}, {"val":-1.6, "name":"N3|1003"}]
```

[예시 16] UAclient.read()

6.5. PLC 통신드라이버

6.5.1. LS ELECTRIC PLC: lsclient

lsclient 드라이버는 Enet(TCP/IP)를 지원하는 LS Electric PLC와의 통신을 위한 드라이버로 LS Electric PLC로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

LSclient.open(*ip-addr*, *port-no*)

ip-addr와 ***port-no***를 이용하여 설비와 연동합니다.

- ***ip-addr*** : 연결하고자 하는 IP 주소.
- ***port-no*** : 소켓 통신에 사용되는 포트 번호로 2004 번을 기본 포트로 사용합니다.

```
var sess = LSclient.open("192.168.1.191", 2004);
```

[예시 17] LSclient.open()

LSclient.read(*sess, variables*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***Variables*** : 값을 읽어오기 위한 직접변수를 배열로 전달합니다. 반환되는 값은 요청한 직접변수와 동일한 크기, 타입의 배열로 반환됩니다.

```
var sess = LSclient.open("192.168.1.191", 2004);
LSclient.read(sess, ["%MW101", "%MW102"]);
```

[예시 18] LSclient.read()

LSclient.write(*sess*, *tag-array*, *values*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***variables*** : 값을 쓰기 위한 직접변수로 배열 형식으로 전달되어야 합니다.
- ***values*** : 배열로 전달되야 하며 tag-array 크기와 일치되어야 합니다.

```
var sess = LSclient.open("192.168.1.191", 2004);
LSclient.write(sess, ["%MW101", "%MW102"], [45, 64]);
```

[예시 19] LSclient.write()

※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비 담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

lsclient 드라이버의 사용가능한 직접변수는 데이터 타입별로 다음과 같습니다.

데이터 타입	사용 예
Bit	%MX0 %QX0.0.0 %IX0.0.0 %PX0 %LX0 %FX0
Byte	%MB10 %QB0.0.0 %IB0.0.0
Word	%MW10 %QW0.0.0 %IW0.0.0 %PW0 %LW0 %FW0 %DW0
Double Word	%MD10 %QD0.0.0 %ID0.0.0
Long Word	%ML10 %QL0.0.0 %IL0.0.0(GM1/2만 사용가능)

[표 8] 데이터 타입 및 사용 예

드라이버를 이용한 사용 예는 다음과 같습니다. 아래의 사용 예는 이해를 돕기 위한 예제로 실제 적용하기에는 적합하지 않으며 PLC 쓰기 함수가 있으므로 설비 담당자와의 충분한 협의 후 진행하여야 합니다.

```
// 드라이버를 로드
Lib.load("lsclient");
// PLC와의 연결
var sess = LSclient.open("192.168.1.191", 2004);
// PLC에 난수 값 쓰기
LSclient.write(sess, ["%MW100", "%MW101", "%MW102"], [Math.random() * 100, Math.random() * 200,
Math.random() * 300]);
// PLC로부터 값 읽기
var val = LSclient.read(sess, ["%MW100", "%MW101", "%MW102"]);
Log.add(val);
```

[예시 20] LSclient Driver

6.5.2. LS ELECTRIC PLC: lsmaster

Lsmaster 드라이버는 Cnet(시리얼)을 지원하는 LS ELECTRIC PLC 와의 통신을 위한 드라이버로 LS ELECTRIC PLC 로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

LSmaster.open(*port-name*, *communication-param*)

드라이버와 PLC 를 연결하기 위한 함수로 연결하고자 하는 시리얼 포트와 통신속성을 지정하여 연결하며 성공하면 세션번호(0 보다 큰 값)이 반환되며 반환된 세션번호를 이용하여 읽고, 쓰기를 처리하게 됩니다.

- ***port-name*** : 연결하고자 하는 통신포트로 /dev/ttyUSB0 와 같은 형식으로 지정되며 스마트 게이트웨이 초기화면에서 연결가능한 시리얼 포트를 확인할 수 있습니다.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다. 사용 예는 다음과 같습니다. 설정되는 통신 파라메터는 다음과 같습니다.

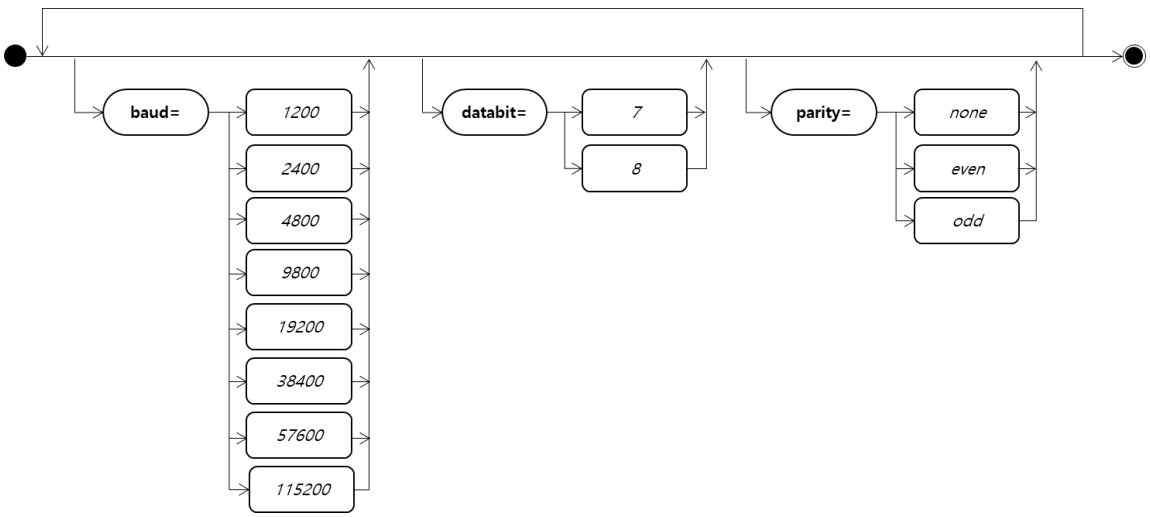
Parameter	설명
baud	통신속도를 지정하기 위한 항목으로 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 28400, 57600, 115200에서 선택하여 지정합니다.
data	데이터 비트(bit)값을 지정하기 위한 항목으로 7 혹은 8 값으로 지정합니다.
parity	에러 검사를 위한 패리티비트를 지정하는 항목으로 none, odd, even 중 한 값을 지정하여 선택합니다.

[표 9] Parameter 종류 및 설명

사용가능한 예제는 다음과 같습니다.

```
baud=2400 databit=8 parity=none
baud=9600 databit=8 parity=odd
baud=19200 databit=8 parity=even
```

[예시 21] LSmaster.open(): communication-param



[그림 31] 시리얼 통신 파라메터

LSmaster.read(*sess*, *variable*, *count*)

PLC 로부터 값을 읽어 바이트 배열로 반환합니다.

- ***sess*** : open 함수에서 반환된 세션 값.
- ***variable*** : 읽어오고자 하는 디바이스 변수 값을 지정합니다. 사용가능한 디바이스 변수는 '그림 30'에서 확인하실 수 있습니다.
- ***count*** : 읽어오고자 하는 데이터 개수.

영 역	범 위	크 기(Word)	비 고
P	P0 – P2047	2048	읽기/쓰기/모니터 가능
M	M0 – M2047	2048	읽기/쓰기/모니터 가능
K	K0 – K2047	2048	읽기/쓰기/모니터 가능
F	F0 – F2047	2048	읽기/모니터 가능
T	T0 – T2047	2048	읽기/쓰기/모니터 가능
C	C0 – C2047	2048	읽기/쓰기/모니터 가능
Z	Z0 – Z127	128	읽기/쓰기/모니터 가능
S	S0 – S127	128	읽기/쓰기/모니터 가능
L	L0 – L11263	11264	읽기/쓰기/모니터 가능
N	N0 – N21503	21504	읽기/쓰기/모니터 가능
D	D0 – D32767	32768	읽기/쓰기/모니터 가능, XGK-CPUH
	D0 – D19999	20000	읽기/쓰기/모니터 가능, XGK-CPUS
R	R0 – R32767	32768	읽기/쓰기/모니터 가능
ZR	ZR0 – ZR65535	65536	읽기/쓰기/모니터 가능, XGK-CPUH 에서만 사용가능

[그림 32] 사용가능한 디바이스 이름

LSmaster.write(*sess*, *tag-array*, *values*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***variables*** : 값을 쓰기위한 위한 직접변수로 배열형식으로 전달되어야 합니다.
- ***values*** : 배열로 전달되어야 하며 tag-array 크기와 일치되어야 합니다.

※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비 담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

드라이버를 이용한 사용 예는 다음과 같습니다. 아래의 사용 예는 이해를 돕기 위한 예제로 실제 적용하기에는 적합하지 않으며 PLC 쓰기 함수가 있으므로 설비 담당자와의 충분한 협의 후 진행하여야 합니다.

```
// 드라이버 로드
Lib.load("lsmaster");
// PLC와의 연결
var sess = LSmaster.open("dev/usbtty0", "baud=9600 data=8 parity=none");
// PLC에 난수 값 쓰기
LSmaster.write(sess, ["%MW100"], [Math.random() * 100]);
// PLC로부터 값 읽기
var val = LSmaster.read(sess, "%MW100", 1);
Log.add(val);
```

[예시 22] LSmaster Driver

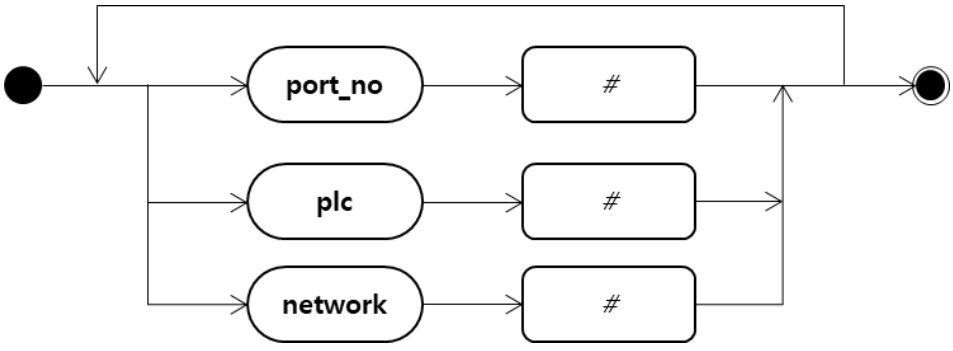
6.5.3. 미쓰비시 PLC: mqclient

mqclient 드라이버는 미쓰비시 Q 시리즈 PLC 와의 통신을 위한 드라이버로 미쓰비시 Q 시리즈 PLC 로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

MQclient.open(*ip-addr*, *parameter*)

ip-addr 와 ***port-no*** 를 이용하여 설비와 연동합니다.

- ***ip-addr*** : 연결하고자 하는 IP 주소.
- ***parameter*** : 연결에 필요한 연결정보를 지정합니다. PORT, PLC, NETWORK 번호를 지정할 수 있으며 생략될 경우 PORT 번호는 5010, PLC 및 NETWORK 는 각각 1 값이 기본값으로 할당됩니다.



[그림 33] MQclient 통신 파라메터

```
var sess = MQclient.open("192.168.1.191", "port_no=5010");
var sess = MQclient.open("192.168.1.191", "port_no=5010 plc=1");
var sess = MQclient.open("192.168.1.191", "port_no=5010 network=1");
```

[예시 23] MQclient.open()

MQclient.read(*sess*, *device*, *count*, *type*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***device*** : 값을 읽어오기 위한 디바이스 변수.
- ***count*** : 읽어오고자 하는 값의 개수.
- ***type*** : 읽어오고자 하는 변수의 타입으로 "short", "word", "int", "dword", "float" 값이 지정될 수 있습니다. 생략할 경우 "word"로 지정됩니다.

다음은 사용 예입니다.

```
MQclient.read(sess, "D*100", 100, 10);
MQclient.read(sess, "D*100", 100, 10, "dword");
```

[예시 24] MQclient.read(.., device, count, type)

MQclient.read(*sess*, *variables*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***variables*** : 값을 읽어오기 위한 직접변수를 배열로 전달합니다.

반환되는 값은 요청한 직접변수와 동일한 크기, 타입의 배열로 반환됩니다. 다음은 사용 예입니다.

```
MQclient.read(sess, ["D*100", "D*150"]);
```

[예시 25] MQclient.read(.., variables)

MQclient.write(*sess*, *device*, *value*, *type*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***device*** : 값을 읽어오기 위한 디바이스 변수.
- ***value*** : 쓰고자 하는 값으로 type 에 따라 변환되어 PLC 로 쓰게 됩니다.

- ***type*** : 읽고자 하는 변수의 타입으로 “short”, “word”, “int”, “dword”, “float” 값이 지정될 수 있습니다. 생략할 경우 “word”로 지정됩니다.

다음은 사용 예입니다.

```
MQclient.write(sess, “D*100”, 156);
MQclient.write(sess, “D*100”, 655398, “int”);
```

[예시 26] MQclient.write()

※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비 담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

mqclient 드라이버는 사용가능한 디바이스 변수는 다음과 같습니다.

분류	디바이스		디바이스 종류		디바이스 코드		디바이스 번호범위(디폴트 할당 시)		표 현		비 고
			비트	워드	ASCII 코드시	바이너리 코드시	Q02(H),Q06H,Q12H,Q25H, Q12PH,Q25PH,Q2A,Q2A-S1, Q2AS,Q2AS-S1,Q2ASH, Q2ASH-S1,Q3A,Q4A,Q4AR	Q00J,Q00,Q01	10 진수	16 진수	
내부 시스템	기능입력		○		——	——	000000~00000F	000000~00000F		○	액세스 불가
	기능출력		○		——	——	000000~00000F	000000~00000F		○	
	기능 레지스터			○	——	——	000000~000004	000000~000004	○		
	특수 릴레이		○		SM	91H	000000~002047	000000~001023	○		——
	특수 레지스터			○	SD	A9H	000000~002047	000000~001023	○		
내부 사용자	입력 릴레이		○		X*	9CH	000000~001FFF	000000~0007FF		○	디바이스의 할당 변경의 가능점수는 사용할 PLC CPU의 매뉴얼을 참조 할당 변경시에는 변경후의 최대 디바이스 번호까지 액세스 가능. 로컬 디바이스는 액세스 불가
	출력 릴레이		○		Y*	9DH	000000~001FFF	000000~0007FF		○	
	내부 릴레이 *2		○		M*	90H	000000~008191	000000~008191	○		
	래치 릴레이 *2		○		L*	92H	000000~008191	000000~002047	○		
	어넌시에이터		○		F*	93H	000000~002047	000000~001023	○		
	에지 릴레이		○		V*	94H	000000~002047	000000~001023	○		
	링크 릴레이		○		B*	A0H	000000~001FFF	000000~0007FF		○	
	데이터 레지스터			○	D*	A8H	000000~012287	000000~011135	○		
	링크 레지스터			○	W*	B4H	000000~001FFF	000000~0007FF		○	
	타이머 *3	접점	○		TS	C1H	000000~002047	000000~000511	○		
		코일	○		TC	C0H			○		
		현재값		○	TN	C2H			○		
	적산 타이머 *3	접점	○		SS	C7H			○		
		코일	○		SC	C6H			○		
		현재값		○	SN	C8H			○		
	카운터 *3	접점	○		CS	C4H	000000~001023	000000~000511	○		
		코일	○		CC	C3H			○		
		현재값		○	CN	C5H			○		
	링크 특수 릴레이		○		SB	A1H	000000~0007FF	000000~0003FF		○	
	링크 특수 레지스터			○	SW	B5H	000000~0007FF	000000~0003FF		○	
	스텝 릴레이 *2		○		S*	98H	000000~008191	000000~002047	○		Q00J/Q00/Q01CPU로는 액세스 불가
	직접 입력		○		DX	A2H	000000~001FFF	000000~0007FF		○	입력 릴레이, 출력 릴레이와 동일 (직접 액세스용)
	직접 출력		○		DY	A3H	000000~001FFF	000000~0007FF		○	
	인덱스 레지스터			○	Z*	CCH	000000~000015	000000~000009	○		——
——	파일 *4, *5, *6 레지스터			○	R*	AFH	000000~032767	000000~032767	○		블록 전환에 의한 일반 액세스 용
					ZR	B0H	000000~0FE7FF	000000~00FFFF		○	연속번호 액세스용

[그림 34] 디바이스 일람

드라이버를 이용한 사용 예는 다음과 같습니다. 아래의 사용 예는 이해를 돕기 위한 예제로 실제 적용하기에는 적합하지 않으며, PLC 쓰기 함수가 있으므로 설비 담당자와의 충분한 협의 후 진행하여야 합니다.

```
// 드라이버 로드
Lib.load("mqclient");
// PLC와의 연결
var sess = MQclient.open("192.168.1.191", "port_no=5010");
// PLC에 난수 값 쓰기
MQclient.write(sess, "D*100", Math.random() * 200);
// PLC로부터 값 읽기
var val = MQclient.read(sess, "D*100", 1);
Log.add(val);
```

[예시 27] MQclient Driver

6.5.4. 미쓰비시 PLC: fxmaster

Fxmaster 드라이버는 미쓰비시 FX1s PLC 와의 통신을 위한 드라이버로 미쓰비시 FX1s PLC 로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

FXmaster.open(*port-name*, *communication-param*)

드라이버와 PLC 를 연결하기 위한 함수로 연결하고자 하는 시리얼 포트와 통신속성을 지정하여 연결하며 성공하면 세션번호(0 보다 큰 값)이 반환되며 반환된 세션번호를 이용하여 읽고, 쓰기를 처리하게 됩니다.

- ***port-name*** : 연결하고자 하는 통신포트로 /dev/ttyUSB0 와 같은 형식으로 지정되며 스마트 게이트웨이 초기화면에서 연결가능한 시리얼 포트를 확인할 수 있습니다.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다. 사용 예는 다음과 같습니다. 설정되는 통신 파라메터는 다음과 같습니다.

Parameter	설명
baud	통신속도를 지정하기 위한 항목으로 300, 600, 1200, 2400, 4800, 9600, 14400,

	19200에서 선택하여 지정합니다.
data	데이터 비트(bit)값을 지정하기 위한 항목으로 7 혹은 8 값으로 지정합니다.
parity	에러 검사를 위한 패리티비트를 지정하는 항목으로 none, odd, even 중 한 값을 지정하여 선택합니다.
stop	데이터 패킷의 끝을 알리기 위한 정지 비트로 1 또는 2를 선택하여 지정합니다.

[표 10] 파라메터 종류 및 설명

사용가능한 예제는 다음과 같습니다.

```
baud=2400 databit=8 stop=1 parity=none
baud=9600 databit=8 stop=1 parity=odd
baud=19200 databit=8 stop=2 parity=even
```

[예시 28] FXmaster.open(): communication-param

FXmaster.read(*sess*, *variable*, *count*)

PLC 로부터 값을 읽어 바이트 배열로 반환합니다.

- ***sess*** : open 함수에서 반환된 세션 값.
- ***variable*** : 읽어오고자 하는 디바이스 변수 값을 지정합니다. 사용가능한 디바이스 변수는 '그림 33'에서 확인하실 수 있습니다.
- ***count*** : 읽어오고자 하는 데이터 개수.

1) Bit devices

Device		Device specification characters					Decimal/octal expression	Usable command	
		FX0N	FX1S	FX, FX2C	FX1N	FX2N, FX2NC		BR, BW, BT	WR, WW, WT
Inputs	(X)	X0000~X0177	X0000~X0017	X0000~X0337	X0000~X0177	X0000~X0267	Octal	○	○
Outputs	(Y)	Y0000~Y0177	Y0000~Y0015	Y0000~Y0337	Y0000~Y0177	Y0000~Y0267			
Auxiliary relays	(M)	M0000~M0511		M0000~M1535		M0000~M3071	Decimal		
States	(S)	S0000~S0127		S0000~S0999					
Special auxiliary relays	(M)	M8000~M8254		M8000~M8255					
Timer contacts	(T)	TS000~TS063		TS000~TS255					×
Counter contacts	(C)	CS000~CS031 CS235~CS254		CS000~CS255					

In FX series, the timer coil (TC) and counter coil (CC) are not supported.

2) Word devices

Device		Device specification characters				Decimal/ octal expression	Usable command		
		FX0N	FX1S	FX, FX2C	FX1N, FX2N, FX2NC		BR, BW, BT	WR, WW	WT
Timer current value	(T)	TN000~TN063		TN000~TN255		Decimal	×	○	○
Counter current value	(C)	CN000~CN031 CN235~CN254		CN000~CN255					○*
Data registers	(D)	D0000~D0255		D0000~D0999	D0000~D7999				○
File registers	(D)	D1000~D2499	—	D1000~D2999	—				
RAM file registers	(D)	—		D6000~D7999	—				
Special data registers	(D)	D8000~D8255		D8000~D8255					

* Only CN000 to CN199 can be used.
(C200 to C255 of 32 bit counter or high speed counter cannot be used.)

[그림 35] 사용 가능한 디바이스 이름

FXmaster.write(*sess*, *variable*, *array-values*)

- ***sess*** : open() 함수에서 반환된 세션 ID
- ***variables*** : 값을 쓰기 위한 디바이스 변수 명.
- ***array-values*** : 배열로 전달되야 합니다. Bit 디바이스에는 0 과 1 의 값이, Word 디바이스에는 정수 값이 들어가야 합니다.

※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

드라이버를 이용한 사용 예는 다음과 같습니다. 아래의 사용 예는 이해를 돕기 위한 예제로 실제 적용하기에는 적합하지 않으며 PLC 쓰기 함수가 있으므로 설비 담당자와의 충분한 협의 후 진행하여야 합니다.

```
Lib.load("fxmaster")  // 드라이버 로드

var sess = FXmaster.open("/dev/ttyUSB0", "baud=9600 data=8 stop=1 parity=N")  //PLC 연결
Log.add(sess); //PLC 연결 확인

FXmaster.write(sess, "X0040", [0, 1, 1, 0, 1]); //Bit 디바이스에 쓰기
var bytes = FXmaster.read(sess, "X0040", 5); //Bit 디바이스 읽기
Log.add(bytes) //읽은 값 로그에 띄우기 (결과: 0,1,1,0,1)

//Word 디바이스에 쓰기
FXmaster.write(sess, "D0020", [1234, -28492, 3330]);
//Word 디바이스 읽기
var wbytes = FXmaster.read(sess, "D0020", 4);
//읽은 값 로그에 띄우기 (결과: 1234,-28492,3330,0)
Log.add(wbytes);
```

[예시 29] FXmaster Driver

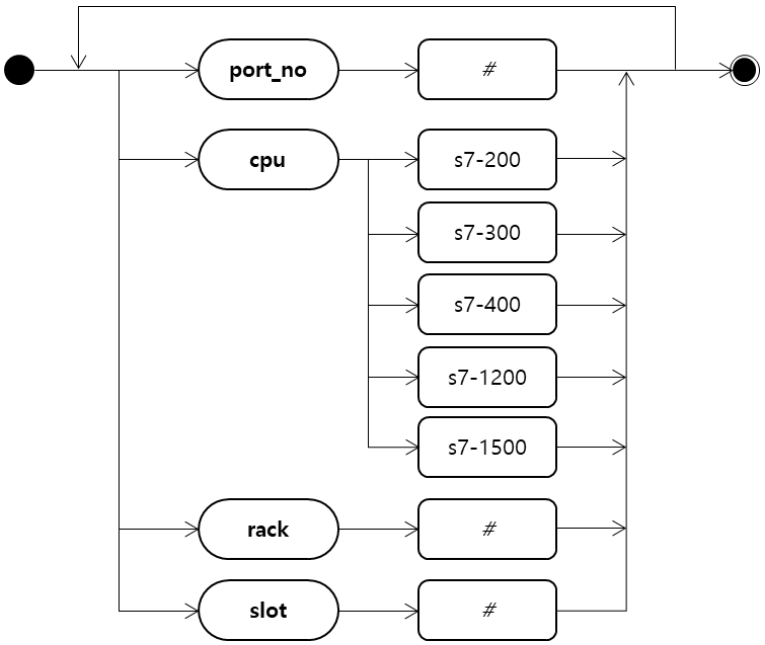
6.5.5. 지멘스 PLC: s7client

s7client 드라이버는 OPEN TCP 프로토콜을 지원하는 지멘스 PLC 와의 통신을 위한 드라이버로 지멘스 PLC 로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

S7client.open(*ip-addr*, *parameter*)

ip-addr *와* ***port-no*** 를 이용하여 설비와 연동합니다.

- ***ip-addr*** : 연결하고자 하는 IP 주소.
- ***parameter*** : 연결에 필요한 연결 정보를 지정합니다. CPU, RACK, SLOT 정보를 지정할 수 있으며 생략될 경우 CPU 는 s7-300, Port 번호는 102, Rack 는 0, Slot 은 1 값이 기본값으로 할당됩니다.



[그림 36] S7client 통신 파라메터

다음은 사용 예입니다.

```
var sess = S7client.open("192.168.1.201", "port_no=102");
var sess = S7client.open("192.168.1.201", "port_no=102 cpu=s7-200");
var sess = S7client.open("192.168.1.201", "port_no=5010 cp=s7-200 rack=0 slot=1");
```

[예시 30] S7client.open()

S7client.read(*sess*, *device*, *count*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***devices*** : 값을 읽어오기 위한 디바이스 변수.
- ***count*** : 읽어오고자 하는 값의 개수.

다음은 사용 예입니다.

```
S7client.read(sess, "DB1.DBX0.0", 5);
S7client.read(sess, "DB1.DBW10", 100);
```

[예시 31] S7client.read()

S7client.write(*sess*, *device*, *value*)

- ***sess*** : open() 함수에서 반환된 세션 ID.

- ***device*** : 값을 읽어오기 위한 디바이스 변수.
- ***value*** : 쓰고자 하는 값으로 type 에 따라 변환되어 PLC 로 쓰게 됩니다.

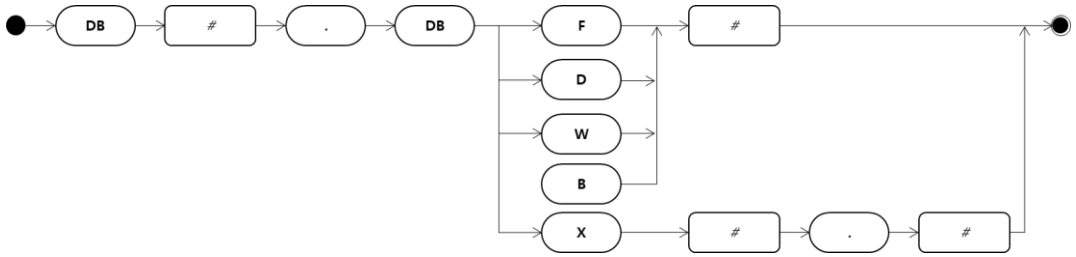
※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

다음은 사용 예입니다.

```
S7client.write(sess, "DB1.DBW50", 10);
S7client.write(sess, "DB1.DBF100", 3.14);
```

[예시 32] S7client.write()

smclient 드라이버는 사용가능한 디바이스 변수는 다음과 같습니다.



[그림 37] 지멘스 PLC 디바이스변수 구성

위의 구성형식에 따른 변수명의 예는 다음표와 같습니다.

변수 명	설명
DB1.DBX10.9	DB1 영역의 10번째 위치의 워드 값(16bit)에서 9번째 비트정보 값을 저장하고 있는 변수
DB1.DBW100	DB1영역의 100번째 위치의 워드 값
DB1.DBB59	DB1영역의 59번째 바이트 값(8 bit)
DB100.DBD6	DB100영역의 6번째 위치의 더블 워드 값(32 bit)
DB2.DBF6	DB2영역의 6번째 위치의 실수 값(32 bit)

[표 11] siemens 변수 구성방법

드라이버를 이용한 사용 예는 다음과 같습니다. 아래의 사용 예는 이해를 돕기 위한 예제로 실제 적용하기에는 적합하지 않으며 PLC 쓰기 함수가 있으므로 설비 담당자와의 충분한 협의 후 진행하여야 합니다.

```
// 드라이버 로드
Lib.load("s7client");
// PLC와의 연결
var sess = S7client.open("192.168.1.201", "port_no=102 cpu=S7-300 rack=0 slot=2");
// PLC에 난수 값 쓰기
S7client.write(sess, "DB1.DBW0", Math.random() * 200);
// PLC로부터 값 읽기
var val = S7client.read(sess, "DB1.DBW0", 1);
Log.add(val);
```

[예시 33] S7client 통신드라이버 사용예

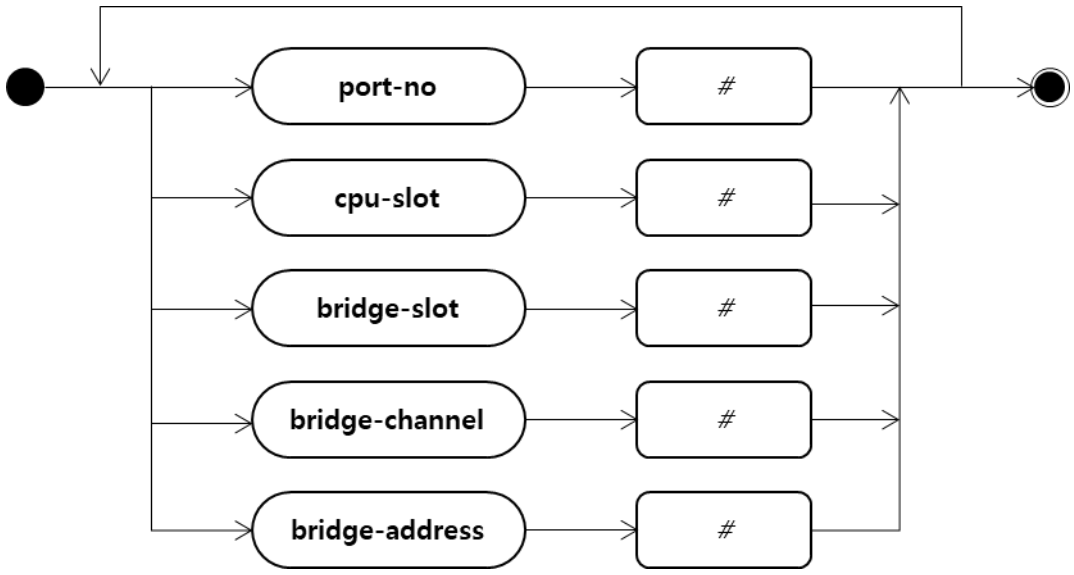
6.5.6. ABB PLC: eipclient

eipclient 드라이버는 Ethernet/IP 프로토콜을 지원하는 ABB PLC 와의 통신을 위한 드라이버로 ABB PLC 로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

EIPclient.open(*ip-addr*, *parameter*)

ip-addr *와* ***port-no*** 를 이용하여 설비와 연동합니다.

- ***ip-addr*** : 연결하고자 하는 IP 주소.
- ***parameter*** : 연결에 필요한 연결 정보를 지정합니다. port-no, cpu-slot, bridge-slot, bridge-channel, bridge-address 를 지정할 수 있으며 생략될 경우 port-no 는 44818, cpu-slot 은 0 값으로 지정되며, 통신카드가 CPU 와 다른 rack 에 설치된 경우에는 bridge 정보를 추가적으로 입력하여야 합니다.



[그림 38] S7client 통신 파라메터

다음은 사용 예입니다.

```
var sess = EIPclient.open("192.168.1.201", "port_no=44818");
var sess = EIPclient.open("192.168.1.201", "cpu-slot=1");
```

[예시 34] EIPclient.open()

EIPclient.read(*sess*, *tags*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***tags*** : 값을 읽어오기 태그이름 배열

다음은 사용 예입니다.

```
EIPclient.read(sess, ["EIP_FLOW01", "EIP_ANALOG1"]);
```

[예시 35] EIPclient.read()

EIPclient.write(*sess*, *tag*, *value*)

- ***sess*** : open() 함수에서 반환된 세션 ID.
- ***tag*** : 값을 쓰기 위한 태그명.
- ***value*** : 쓰고자 하는 값으로 type 에 따라 변환되어 PLC 로 쓰게 됩니다.

※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

다음은 사용 예입니다.

```
EIPclient.write(sess, "EIP_FLOW01", 3.65);
```

[예시 36] EIPclient.write()

드라이버를 이용한 사용 예는 다음과 같습니다. 아래의 사용 예는 이해를 돕기 위한 예제로 실제 적용하기에는 적합하지 않으며 PLC 쓰기 함수가 있으므로 설비 담당자와의 충분한 협의 후 진행하여야 합니다.

```
// 드라이버 로드
Lib.load("eipclient");
// PLC와의 연결
var sess = EIPclient.open("192.168.1.101", "port_no=44818");
// PLC에 난수 값 쓰기
EIPclient.write(sess, " EIP_FLOW01", Math.random() * 200);
// PLC로부터 값 읽기
var val = EIPclient.read(sess, ["EIP_FLOW01"]);
Log.add(val);
```

[예시 37] EIPclient Driver 사용 예제

6.5.7. Panasonic 조명제어 PLC: NCUmaster

NCUmaster 드라이버는 NCU 프로토콜을 지원하는 Panasonic 조명제어 PLC 와의 통신을 위한 드라이버로 Panasonic 조명제어 PLC 로부터 데이터를 읽어오거나 쓰기 위한 통신 드라이버입니다.

NCUmaster.open(*port-name*, *communication-param*)

드라이버와 PLC 를 연결하기 위한 함수로 연결하고자 하는 시리얼 포트와 통신속성을 지정하여 연결하며 성공하면 세선번호(0 보다 큰 값)이 반환되며 반환된 세션번호를 이용하여 읽고, 쓰기를 처리하게 돕니다.

- ***port-name*** : 연결하고자 하는 통신포트로 /dev/ttyUSB0 와 같은 형식으로 지정되며 스마트 게이트웨이 초기화면에서 연결가능한 시리얼 포트를 확인할 수 있습니다.
- ***communication-param*** : 통신포트를 오픈할 때 사용되는 통신 파라메터 값으로 통신속도, 패리티 등을 지정합니다. 사용 예는 다음과 같습니다. 설정되는 통신 파라메터는 다음과 같습니다.

Parameter	설명
baud	통신속도를 지정하기 위한 항목으로 300, 600, 1200, 2400, 4800, 9600, 14400, 19200에서 선택하여 지정합니다.
data	데이터 비트(bit)값을 지정하기 위한 항목으로 7 혹은 8 값으로 지정합니다.
parity	에러 검사를 위한 패리티비트를 지정하는 항목으로 none, odd, even 중 한 값을 지정하여 선택합니다.
stop	데이터 패킷의 끝을 알리기 위한 정지 비트로 1 또는 2를 선택하여 지정합니다.

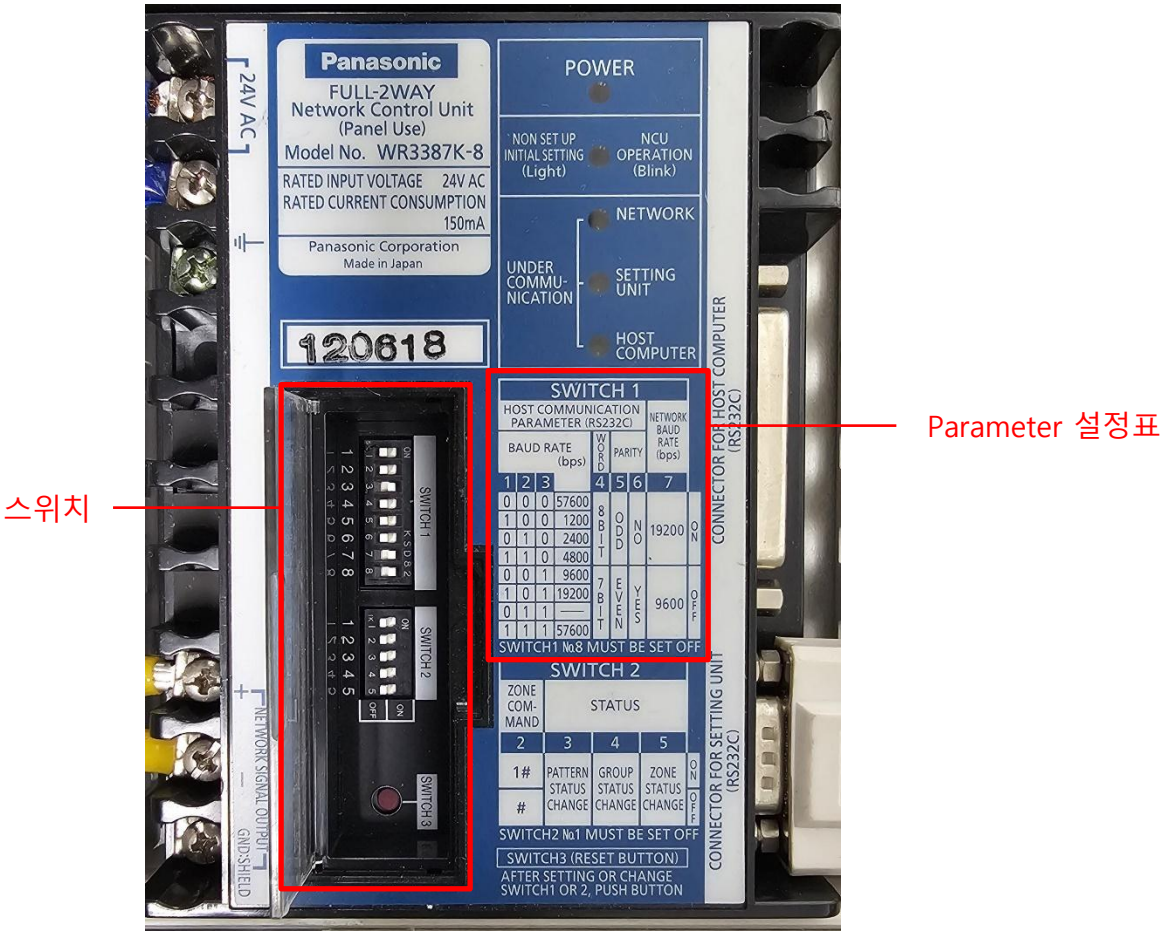
[표 12] Parameter 종류 및 설명

사용가능한 예제는 다음과 같습니다.

```
baud=2400 databit=8 stop=1 parity=none
baud=9600 databit=7 stop=1 parity=even
baud=19200 databit=8 stop=2 parity=even
```

[예시 38] NCUmaster.open(): communication-param

※ NCUmaster 드라이버의 open() parameter 설정은 조명제어 PLC 제품에 설정된 값과 동일해야 합니다. PLC parameter 값은 아래 사진에서 확인 가능합니다.



[그림 39] Panasonic 조명제어 PLC

각 스위치의 ON/OFF 상태를 보고 Parameter 설정표에 따라 open() 함수의 parameter 값을 설정해주시면 됩니다.

NCUmaster.read(*sess*, *command*)

- ***sess*** : open() 함수에서 반환된 세션 값.
- ***command***: 읽어오고자 하는 데이터 종류를 지정합니다. 사용가능한 입력 값은 아래 표에서 확인 가능합니다.

기능	Script 명령어	Script 예시
부하 감시	LOAD-LCU-Group-Point	"LOAD-1-50-1"
부하 일괄 감시	LOADBATCH-LCU-Group	"LOADBATCH-1-50"
접속 전송장치 종별 감시	UNITKIND-LCU	"UNITKIND-1"
부하 조광 Level 일괄 감시 (Level 7)	DIMBATCH7-LCU-Group	"DIMBATCH7-1-59"
부하 조광 Level 감시 (Level 7)	DIM7-LCU-Group-Point	"DIM7-1-59-1"

부하 조광 Level 일괄 감시 (Level 128)	DIMBATCH128-LCU-Group	"DIMBATCH128-1-50-1"
부하 조광 Level 감시 (Level 128)	DIM128-LCU-Group-Point	"DIM128-1-50-1"
Zone 감시	ZONE-LCU	"ZONE-1"
개별 부하 감시	INDIVLOAD-LCU	"INDIVLOAD-1"
조광 부하 감시	DIMLOAD-LCU	"DIMLOAD-1"
Pattern 감시	PAT-LCU	"PAT-1"
Group 감시	GRP-LCU	"GRP-LCU"
Group 조광 Level 감시 (Level 7)	GRPDIM7-LCU	"GRPDIM7-1"
개별 부하 조광 Level 감시 (Level 7)	INDIVDIM7-LCU	"INDIVDIM7-1"
Group 조광 Level 감시 (Level 128)	GRPDIM128-LCU-Packet	"GRPDIM128-1-2"
개별 부하 조광 Level 감시 (Level 128)	INDIVDIM128-LCU-Packet	"INDIVDIM128-1-2"

[표 13] NCUmaster.read() Command 목록

```
var data = NCUmaster.read(sess, "LOAD-1-50-1");
```

[예시 39] NCUmaster.read() 예시

NCUmaster.write(*sess*, *command*, *values*)

- ***sess*** : open() 함수에서 반환된 세션 ID
- ***command*** : 사용하고자 하는 기능 명령어
- ***values*** : 제어하고자 하는 기능의 주소 값과 제어 값. 기능에 따라 IO 또는 DIMMER 설정 등을 할 수 있습니다.

※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

기능	Script 명령어	Script 예시	설명
부하 제어 요구	CTRL-LCU-Group-Point, ON/OFF/HALF1/HALF2	"CTRL-1-50-7", 0	0 : OFF / 1 : ON / 2 : 반등1 / 3 : 반등2

부하 제어 요구(Global)	CTRL-Group-Point, [스위치 번호 / "ALL"], ON/OFF/HALF1/HALF2	"CTRLGL-50-1". [1, 2, 3], 1	[1, 2, 3,] : 유효한 스위치 번호 입력 (1~30) 0 : OFF / 1 : ON / 2 : 반등1 / 3 반등2
		"CTRLGL-50-1", ["ALL"], 1	["ALL"] : 스위치 전체 유효
조광 Level 설정 제어 요구	DIMLV-LCU, [Level / "Level", 스위치 번호],	"DIMLV-1", [7]	[7] : 스위치 전체 Level 7 설정 (Level 1~ Level 7)
		"DIMLV-1", ["L3", 3, 10, "L4", 7]	["L3", 3, 10, "L4", 7"] : 3, 10번 스위치 Level 3 설정 7번 스위치 Level 4 설정
조광 Level UP•DOWN 제어 요구	DIMIO-LCU-Group-Point, ON/OFF/HALF1/HALF2	"DIMIO-1-59-1", 1	0 : 정지 1 : UP 연속 2 : DOWN 연속

[표 14] NCUmaster.write() Command 목록

```
NCUmaster.write(sess, "CTRL-1-50-7", 1);
```

[예시 40] NCUmaster.write() 예시

NCUmaster.check(*sess*, *command*)

NCUmaster.check() 함수는 현재 조명제어 PLC 에 입력된 설정 값을 읽는 기능을 수행합니다.

- ***sess*** : open() 함수에서 반환된 세션 ID
- ***command*** : 사용하고자 하는 기능 명령어

기능	Script 명령어	Script 예시
패턴•그룹 Data 확인	CHPATGRP-LCU-Group-Point	"CHPATGRP-1-60-1"
패턴•그룹 Data 확장 설정 확인 (Level 7)	CHPATGRP7-LCU-Group-Point	"CHPATGRP7-1-70-1"
패턴•그룹 Data 확정 설정 확인 (Level 128)	CHPATGRP128-LCU-Group-Point	"CHPATGRP128-1-70-1"
LCU 접속 Unit 설정 Monitor	CHUNITNO	"CHUNITNO"

동시 제어 Unit 설정 Monitor	CHSIMUNIT-LCU	"CHSIMUNIT-1"
Zone 설정 Monitor	CHZONE-LCU-Classification	"CHZONE-1-1"

[표 15] NCUmaster.check() Command 목록

```
var data = NCUmaster.check(sess, "CHPATGRP-1-60-1");
```

[예시 41] NCUmaster.check() 예시

NCUmaster.set(*sess*, *command*, *setting-variables*)

NCUmaster.set() 함수는 조명제어 PLC 와 연결된 장치들의 초기 설정을 바꾸는 기능을 수행합니다.

- ***sess*** : open() 함수에서 반환된 세션 ID
- ***command*** : 사용하고자 하는 기능 명령어
- ***setting-variables*** : 설정하고자 하는 기능의 값을 입력해줍니다. 기능에 따라 입력할 값의 개수나 형식이 상이합니다.
- ※ PLC 는 설비와 직접적으로 연동되어 설비를 제어하기 위하여 사용하고 있으므로 PLC 내의 특정 주소 값을 변경하게 되면 설비에 영향을 줄 수 있으므로 write() 명령은 설비담당자와의 충분한 협의를 거친 후 전송하여야 하며 협의하지 않고 전송할 경우 PLC 와 연동된 설비에 영향을 줄 수도 있으므로 주의하여 사용하여야 합니다.

기능	Script 명령어	Script 예시
패턴•그룹 Data 확장 설정	SETPATGRP-LCU-Group-Point, [Indiv Setting], [Dimmer Setting], [Delay Setting]	"SETPATGRP-1-60-1", ["ON"], ["ON", 1], ["ALL", "TIMER,DIM,Y,1s,0,15"]
패턴•그룹 Data 확정 설정 (Level 7)	SETPATGRP7-LCU-Group-Point, Packet, Packet = 0 : [Indiv Setting], [Dimmer Setting] Packet = 1 : [Delay Setting], Fade Setting, [Level Setting]	"SETPATGRP-1-70-1", 0, ["ON", 5, 6, 7, 8, 9, 10, 11, 12], ["ON-7", 1, 2]
패턴•그룹 Data 확장 설정 (Level 128)	SETPATGRP128-LCU-Group-Point, Packet, Packet = 0 : [Indiv Setting], [Dimmer Setting] Packet = 1 : [Delay Setting], Fade Setting, [Level Setting] Packet = 2 / 3 / 4 : [Level Setting]	"SETPATGRP128-1-70-1", 0, ["ON", 5, 6, 7, 8, 9, 10, 11, 12], ["ON-7", 1, 2]
LCU 접속 Unit 설정	SETUNITNO, [Switch No.]	"SETUNITNO", [1, 19]
동시 제어 Unit 설정	SETSIMUNIT-LCU, [Switch No.]	"SETSIMUNIT-1", ["DEFAULT"]

Zone 설정	SETZONE-LCU-Classification, [Pattern]	"SETZONE-1-1", ["ON", 1]

[표 16] NCUmaster.set() Command 목록

```
NCUmaster.set(sess, "SETUNITNO", [1, 19]);
```

[예시 42] NCUmaster.set() 예시

Command	Packet	Variable	Example	설명	
SETPATGRP	-	1. [Individual Setting] Max 256 Switches (총 256개의 스위치 설정)	["DEFAULT"] ["ON"] ["OFF"] ["TD"]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = Override ON = 전체 On 설정 OFF = 전체 Off 설정 TD = 전체 Timer/Delay 설정	
			["ON", 2, 3, "OFF", 32, 33, "TD", 85, 90]	각 스위치에 대한 설정 2, 3번 = ON 32, 33 = OFF 85, 90 = TD 스위치는 1~256번까지 사용 가능하며 입력하지 않은 스위치 번호는 Default 설정 값을 가지게 됩니다.	
		2. [Dimmer Setting] Max 16 Switches (총 16개의 스위치 설정) 조광 Level 1 ~ 7	["DEFAULT"] ["ON", 1"] [OFF", 2] ["TD", 7]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = 전체 Override, Level 0 설정 "ON", 1 = 전체 On Level 1 설정 "OFF", 2 = 전체 Off Level 2 설정 "TD", 7 = 전체 Timer/Delay Level 7 설정	
			["ON-4", 3, 7, "OFF-1", 10, 11, "TD-2", 14, 16]	각 스위치에 대한 설정 3, 7번 = ON, Level 4 10, 11번 = OFF, Level 1 14, 16번 = Timer/Delay, Level 2	
		3. Delay Setting Max 8 Inputs (총 8개의 설정 값)	["DEFAULT"] ["ALL", "TIMER,DIM,Y,1s,0,15"]	-. TIMER / DELAY (일시) / (지연) DIM / INDIV (조광) / 개별) -. Y / N (지연 설정 있음) / (없음) -. 1s / 30s / 1m / 5m / 1h / 2h	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = 지연, 개별, 지연 설정 없음, 1초, 스위치 0번, 채널 0번으로 설정 "ALL" = 다음 들어오는 설정 값으로 전체 스위치 설정

			["TIMER,DIM,Y,1m,0,15", "DELAY,INDIV,Y,2h,1,60"]	(시간 설정) -. 0~3 (스위치 번호 : 조광은 0) -. 지정 Channel : 개별 : 0~63 조광 : 0~15 " " 사이에는 6개의 설정 값이 반드시 모두 입력되어야 합니다.	최소 1개~ 최대 8개의 설정 값 입력 가능 8개 이하 입력 시 나머지 값은 DEFAULT 설정 값으로 통일됩니다.

[표 17] NCUmaster.set() 설정 값 설명 1

Command	Packet	Variable	Example	설명	
SETPATGRP7	0	1. [Individual Setting] Max 256 Switches (총 256개의 스위치 설정)	["DEFAULT"] ["ON"] ["OFF"] ["TD"]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = Override ON = 전체 On 설정 OFF = 전체 Off 설정 TD = 전체 Timer/Delay 설정	
			["ON", 2, 3, "OFF", 32, 33, "TD", 85, 90]	각 스위치에 대한 설정 2, 3번 = ON 32, 33 = OFF 85, 90 = TD	
		2. [Dimmer Setting] Max 16 Switches (총 16개의 스위치 설정) 조광 Level 1 ~ 7	["DEFAULT"] ["ON", 1"] [OFF", 2] ["TD", 7]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = 전체 Override, Level 0 설정 "ON", 1 = 전체 On Level 1 설정 "OFF", 2 = 전체 Off Level 2 설정 "TD", 7 = 전체 Timer/Delay Level 7 설정	
			["ON-4", 3, 7, "OFF-1", 10, 11, "TD-2", 14, 16]	각 스위치에 대한 설정 3, 7번 = ON, Level 4 10, 11번 = OFF, Level 1 14, 16번 = Timer/Delay, Level 2	
	1	1. Delay Setting Max 8 Inputs (총 8개의 설정 값)	["DEFAULT"] ["ALL", "TIMER,DIM,Y,1s,0,15"]	-. TIMER / DELAY (일시) / (지연) DIM / INDIV (조광) / 개별) -. Y / N (지연 설정 있음) / (없음) -. 1s / 30s / 1m / 5m / 1h / 2h (시간 설정) -. 0~3 (스위치 번호 :	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = 지연, 개별, 지연 설정 없음, 1초, 스위치 0번, 채널 0번으로 설정 "ALL" = 다음 들어오는 설정 값으로 전체 스위치 설정

			["TIMER,DIM,Y,1m,0,15", "DELAY,INDIV,Y,2h,1,60"]	조광은 0) -. 지정 Channel : 개별 : 0~63 조광 : 0~15 " " 사이에는 6개의 설정 값이 반드시 입력되어야 합니다.	최소 1개~ 최대 8개의 설정 값 입력 가능 8개 이하 입력 시 나머지 값은 DEFAULT 설정 값으로 통일됩니다.
		2. Fade Setting Max 1 Input	0~3	0 : Fade 없음 또는 그룹 Point 지정 시 1 : 3초 2 : 6초 3 : 1분	

[표 18] NCUmaster.set() 설정 값 설명 2

Command	Packet	Variable	Example	설명
SETPATGRP7	1	3. [Individual Level Setting] Max 256 Switches (총 256개의 스위치 설정)	["DEFAULT"] [1] [2]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = OFF, T/U 없음 1 = ON Level 1 2 = ON Level 2 1~7 사이의 수 입력 시 ON Level 입력 값으로 설정됩니다.
			["L7", 1, 5, 10, 14, "L4", 11, 15, 30, "L1", 19, 25, 81]	각 스위치에 대한 설정 1, 5, 10, 14번 = ON Level 7 11, 15, 30번 = ON Level 4 19, 25, 81 = ON Level 1 스위치는 1~256번까지 입력 가능하며 조광 Level은 1~7까지 "L숫자" 형식으로 입력하여야 합니다. 입력하지 않은 스위치는 Default 설정 값을 가지게 됩니다.
SETPATGRP128	0	1. [Individual Setting] Max 256 Switches (총 256개의 스위치 설정)	["DEFAULT"] ["ON"] ["OFF"] ["TD"]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = Override ON = 전체 On 설정 OFF = 전체 Off 설정 TD = 전체 Timer/Delay 설정
			["ON", 2, 3, "OFF", 32, 33, "TD", 85, 90]	각 스위치에 대한 설정 2, 3번 = ON 32, 33 = OFF 85, 90 = TD
		2. [Dimmer Setting] Max 16 Switches (총 16개의 스위치 설정) 조광 Level 1 ~ 7	["DEFAULT"] ["ON", 1"] [OFF", 2] ["TD", 7]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = 전체 Override, Level 0 설정 "ON", 1 = 전체 On Level 1 설정 "OFF", 2 = 전체 Off Level 2 설정 "TD", 7 = 전체 Timer/Delay Level 7 설정

			["ON-4", 3, 7, "OFF-1", 10, 11, "TD-2", 14, 16]	각 스위치에 대한 설정 3, 7번 = ON, Level 4 10, 11번 = OFF, Level 1 14, 16번 = Timer/Delay, Level 2

[표 19] NCUmaster.set() 설정 값 설명 3

Command	Packet	Variable	Example	설명	
SETPATGRP128	1	1. Delay Setting Max 8 Inputs (총 8개의 설정 값)	["DEFAULT"] ["ALL", "TIMER,DIM,Y,1s,0,15"]	-. TIMER / DELAY (일시) / (지연) DIM / INDIV (조광) / 개별) -. Y / N (지연 설정 있음) / (없음) -. 1s / 30s / 1m / 5m / 1h / 2h (시간 설정) -. 0~3 (스위치 번호 : 조광은 0) -. 지정 Channel : 개별 : 0~63 조광 : 0~15 " " 사이에는 6개의 설정 값이 반드시 입력되어야 합니다.	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = 지연, 개별, 지연 설정 없음, 1초, 스위치 0번, 채널 0번으로 설정 "ALL" = 다음 들어오는 설정 값으로 전체 스위치 설정
			["TIMER,DIM,Y,1m,0,15", "DELAY,INDIV,Y,2h,1,60"]		최소 1개~ 최대 8개의 설정 값 입력 가능 8개 이하 입력 시 나머지 값은 DEFAULT 설정 값으로 통일됩니다.
		2. Fade Setting Max 1 Input	0~3	0 : Fade 없음 또는 그룹 Point 지정 시 1 : 3초 2 : 6초 3 : 1분	
		3. [Individual Level Setting] Max 64 Switches (총 64개의 스위치 설정)	["DEFAULT"] [1] [2]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = OFF, T/U 없음 1 = ON Level 1 2 = ON Level 2 1~7 사이의 수 입력 시 ON Level 입력 값으로 설정됩니다.	
			["L7", 1, 5, 10, 14, "L4", 11, 15, 30, "L1", 19, 25]	각 스위치에 대한 설정 1, 5, 10, 14번 = ON Level 7 11, 15, 30번 = ON Level 4 19, 25 = ON Level 1 스위치는 1~256번까지 입력 가능하며 조광 Level은 1~7까지 "L숫자" 형식으로 입력하여야 합니다. 입력하지 않은 스위치는 Default 설정 값을 가지게 됩니다.	

[표 20] NCUmaster.set() 설정 값 설명 4

Command	Packet	Variable	Example	설명
SETPATGRP128	2 / 3 / 4	1. [Individual Level Setting] Max 64 Switches (총 64개의 스위치 설정)	["DEFAULT"] [1] [2]	전체 스위치를 한 가지 설정 값으로 통일 DEFAULT = OFF, T/U 없음 1 = ON Level 1 2 = ON Level 2 1~7 사이의 수 입력 시 ON Level 입력 값으로 설정됩니다.
			["L7", 1, 5, 10, 14, "L4", 11, 15, 30, "L1", 19, 25]	각 스위치에 대한 설정 1, 5, 10, 14번 = ON Level 7 11, 15, 30번 = ON Level 4 19, 25 = ON Level 1 스위치는 1~256번까지 입력 가능하며 조광 Level은 1~7까지 "L숫자" 형식으로 입력하여야 합니다. 입력하지 않은 스위치는 Default 설정 값을 가지게 됩니다.
SETSIMUNIT	-	1. [Simultaneous Connection Setting] Max 31 Units (총 31 개의 LCU 설정)	["DEFAULT"] ["ALL"]	전체 LCU 설정을 한 가지 설정으로 통일 DEFAULT = 전체 Disconnect 설정 (전체 무효) ALL = 전체 Connect 설정 (전체 유효)
SETUNITNO			[1, 3, 21, 30]	각 LCU 번호에 대한 무효/유효 설정 1, 3, 21, 30번 LCU = Connect 설정 (유효) 입력되지 않은 LCU 번호는 무효 설정됩니다.

[표 21] NCUmaster.set() 설정 값 설명 5

Command	Classification	Variable	Example	설명
SETZONE	1 : Pattern No. 1~36	1. [Zone Setting] Max 36 Patterns	["ON", 1, 3, 5, "OFF", 7, 10, 15, "HALF1", 20, 25]	각 패턴에 대한 상태 설정 1, 3, 5번 패턴 = ON 7, 10, 15번 패턴 = OFF 20, 25번 패턴 = HALF1 입력된 패턴 번호가 없을 경우 기본 설정인 '미 할당'으로 설정됩니다. 설정할 수 있는 상태 값은 다음과 같습니다. ON = 전등 OFF = 소등 HALF1 = 반등 1 HALF2 = 반등 2
	2 : Pattern No. 37~72			

[표 22] NCUmaster.set() 설정 값 설명 6

```
Lib.load("ncumaster");

var sess = NCUmaster.open("/dev/ttyUSB0", "baud=9600 databit=7 stop=1 parity=even");
/* read() */
var data = NCUmaster.read(sess, "LOAD-1-50-1");
var data = NCUmaster.read(sess, "LOADBATCH-1-50");
var data = NCUmaster.read(sess, "UNITKIND-1");
var data = NCUmaster.read(sess, "ZONE-1");
var data = NCUmaster.read(sess, "INDIVLOAD-1");
var data = NCUmaster.read(sess, "DIMLOAD-1");
var data = NCUmaster.read(sess, "PAT-1");
var data = NCUmaster.read(sess, "GRP-1");
var data = NCUmaster.read(sess, "DIM7-1-59-1");
var data = NCUmaster.read(sess, "DIMBATCH7-1-59");
var data = NCUmaster.read(sess, "GRPDIM7-1");
var data = NCUmaster.read(sess, "INDIVDIM7-1");
var data = NCUmaster.read(sess, "GRPDIM128-1-2");
var data = NCUmaster.read(sess, "INDIVDIM128-1-2");
var data = NCUmaster.read(sess, "DIM128-1-50-1");
var data = NCUmaster.read(sess, "DIMBATCH128-1-50");
/* write() */
NCUmaster.write(sess, "CTRL-1-50-7", 0);
NCUmaster.write(sess, "CTRLGL-50-1", [1], 1);
NCUmaster.write(sess, "DIMLV-1", [7]);
NCUmaster.write(sess, "DIMLV-1", ["L7", 1]);
NCUmaster.write(sess, "DIMIO-1-5"-2", 0);
NCUmaster.write(sess, "DIMIO-1-59-1", 0);
NCUmaster.write(sess, "CTRLGL-50-1", [1], 1);
/* check() */
var data = NCUmaster.check(sess, "CHUNITNO");
var data = NCUmaster.check(sess, "CHSIMUNIT-1");
var data = NCUmaster.check(sess, "CHZONE-1-1");
var data = NCUmaster.check(sess, "CHPATGRP-1-60-1");
var data = NCUmaster.check(sess, "CHPATGRP7-1-70-1");
var data = NCUmaster.check(sess, "CHPATGRP128-1-70-1");
/* set() */
NCUmaster.set(sess, "SETPATGRP-1-60-1", ["ON"], ["ON", 1], ["ALL", "TIMER,DIM,Y,1s,0,15"]);
NCUmaster.set(sess, "SETPATGRP7-1-70-1", 0, ["ON", 5, 6, 7, 8, 9, 10, 11, 12], ["ON-7", 1, 2]);
NCUmaster.set(sess, "SETPATGRP128-1-70-1", 0, ["ON", 5, 6, 7, 8, 9, 10, 11, 12], ["ON-7", 1, 2]);
NCUmaster.set(sess, "SETUNITNO", [1, 19]);
NCUmaster.set(sess, "SETSIMUNIT-1", [1, 19]);
NCUmaster.set(sess, "SETZONE-1-1", ["ON", 1]);
```

[예시 43] NCUmaster 드라이버 Script 작성 예시

6.5.8. Panasonic OPIU 조명제어 PLC: LCSRest

LCSRest 드라이버는 TCP/IP 프로토콜을 지원하는 Panasonic OPIU 조명제어 PLC 와의 통신을 위한 드라이버로 OPIU 로부터 데이터를 읽어오거나 제어하기 위한 통신 드라이버입니다.

LCSRest.open(*IP addr, ID, Password*)

드라이버와 OPIU 를 연결하기 위한 함수로 연결하고자 하는 OPIU 의 IP 주소 입력하여 연결합니다. 연결에 성공하면 세션번호(0 보다 큰 값)이 반환되며, 반환된 세션번호를 이용하여 데이터 읽기와 제어할 수 있게 합니다.

- ***IP addr*** : 연결하고자 하는 OPIU IP 주소입니다. OPIU 의 초기 IP 주소는 192.168.0.1 이며, OPIU IP 주소는 PLC 의 웹 페이지 또는 자사 조명제어 시스템을 통하여 변경할 수 있습니다.
- ***ID***, ***Password*** : 연결하고자 하는 OPIU 의 웹페이지 ID 와 비밀번호입니다. 초기 ID 와 비밀번호는 "admin", "admin"입니다. ID 와 비밀번호는 PLC 웹페이지 혹은 자사 조명제어 시스템을 통하여 변경할 수 있습니다.

LCSRest.read(*sess, parameter*)

- ***sess***: open 함수에서 반환된 세션 값입니다.
- ***parameter***: 읽어오고자 하는 데이터의 범위를 지정합니다. 사용 가능한 입력 값은 아래의 표에서 확인 가능합니다.

Parameter	설명	비고
INDIV	개별조명 데이터 256개를 읽어옵니다.	
PG	패턴, 그룹조명 데이터 199개를 읽어옵니다.	0~126번 : Group 데이터 127~71 : Pattern 데이터

[표 23] LCSRest.read() Parameter 설명

Parameter	반환 값	설명
INDIV	-1	사용 안 하는 개별 조명
	0	OFF
	1	ON
	2	Time-delay
PG	-1	사용 안 하는 패턴, 그룹 조명
	0	OFF
	1	ON

[표 24] LCSRest.read() 반환 값 설명

LCSRest.write(*sess*, *Tag-name*, *IO*)

- ***sess*** : open 함수에서 반환된 세션 값입니다.
- ***Tag-name*** : 제어하고자 하는 개별, 패턴, 그룹 조명과 맵핑 된 태그이름을 입력합니다.
- ***IO*** : 입력한 조명을 키고자 하면 1 을, 끄고자 하면 0 을 입력합니다.

```
Lib.load("lcsrest");

var sess = LCSRest.open("192.168.1.xxx", "admin", "admin");

var indivData = LCSRest.read(sess, "INDIV");
var pgData = LCSRest.read(sess, "PG");

var now = new Date();

for(var j = 0; j < indivData.length; j++) {
  var ptName = "01-IO-" + (j + 1).pad(3);
  if(indivData[j] === -1) Db.set(ptName, '-', now, 0xC0, "string");
  else Db.set(ptName, indivData[j], now, 0xC0, "int");
}

for(var j = 0; j < pgData.length; j++) {
  if(j < 127) {
    var groupName = "01-G-" + (j + 1).pad(3);
    if(pgData[j] === -1) Db.set(groupName, '-', now, 0xC0, "string");
    else Db.set(groupName, pgData[j], now, 0xC0, "int");
  }
  else {
    var patternName = "01-P-" + (j - 126).pad(3);
    if(pgData[j] === -1) Db.set(patternName, '-', now, 0xC0, "string");
    else Db.set(groupName, pgData[j], now, 0xC0, "int");
  }
}

LCSRest.write(sess, "01-IO-001", 1);
LCSRest.write(sess, "01-G-010", 0);
```

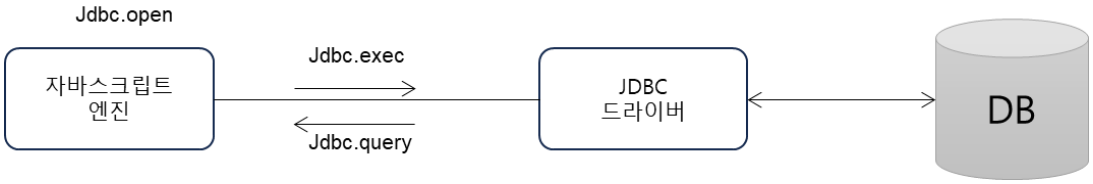
[예시 44] LCSRest Driver 사용예제

7. 스마트 게이트웨이 구성사례

7.1. 범용

데이터베이스 인터페이스

스마트 게이트웨이는 JDBC 가 지원되는 데이터베이스 관리 시스템과의 연결을 통하여 데이터를 조회하거나, 특정 값을 데이터베이스에 저장할 수 있는 기능을 제공합니다.



[그림 40] JDBC 드라이버를 이용한 데이터베이스 연결

```
Lib.load("jdbc");
var sess = Jdbc.open("org.postgresql.Driver", "jdbc:postgresql://192.168.1.100:5432/lcs-db", "lcs-
admin", "lcs-admin");
if(sess !== undefined) {
   var records = Jdbc.query(sess, 'SELECT "month", "option", "value", "time" FROM  serve.schedule
WHERE "useYN" = \'Y\''); }
   Log.add(records);
}
```

[예시 45] JDBC 사용예제

Restful 드라이버를 이용한 공공데이터 포털 데이터 인터페이스

공공 포털 데이터서버에서 제공하는 데이터를 인터페이스 하는 예제로, 공공데이터 포털 서버에서 지원되는 Restful API 를 이용하여 데이터를 수집하여 태그저장소에 저장합니다.

스크립트 예제는 다음과 같습니다.

```
Lib.load("rest");

var now = new Date();
now.setDate(now.getDate() - 1);

var url = "http://apis.data.go.kr/1360000/AsosHourlyInfoService/getWthrDataList";
var serviceKey = "?serviceKey=servicekey";
var options = "&numOfRows=50&pageNo=1&dataCd=ASOS&dateCd=HR&stnIds=108&dataType=JSON";
var param = "&startDt=" + now.format("yyyyMMdd") + "&startHh=" + now.format("HH") + "&endDt=" +
now.format("yyyyMMdd") + "&endHh=" + now.format("HH");

url += serviceKey + options + param;

// 서울시 기상청 관측자료료
var ret = Rest.get(url);
if(ret.response.header.resultCode == '00') {
    for(var i = 0; i < ret.response.body.items.item.length; i++) {
```

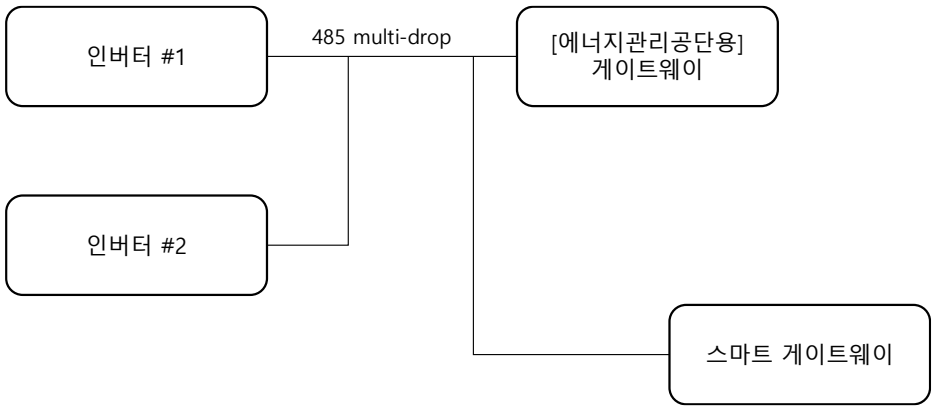
```
        var item = ret.response.body.items.item[i];
        var tm = item.tm;

        Db.set("ASOS."+item.stnId + ".TEMP", Number(item.ta), item.tm, 0xC0, "float");
        Db.set("ASOS."+item.stnId + ".HUMI", Number(item.hm), item.tm, 0xC0, "float");
        Db.set("ASOS."+item.stnId + ".RAINFALL", Number(item.rn), item.tm, 0xC0, "float");
        Db.set("ASOS."+item.stnId + ".WINDSPEED", Number(item.ws), item.tm, 0xC0, "float");
        Db.set("ASOS."+item.stnId + ".WINDDIR", Number(item.wd), item.tm, 0xC0, "float");
        Db.set("ASOS."+item.stnId + ".HPA", Number(item.pv), item.tm, 0xC0, "float");
        Db.set("ASOS."+item.stnId + ".DEWPOINT", Number(item.td), item.tm, 0xC0, "float");
    }
}
```

[예시 46] Restful API: 공공데이터 포털 데이터 연계

7.2. Serial

시리얼통신을 이용하여 신재생 에너지(태양광 발전설비) 연계

신재생에너지 설비의 데이터는 시리얼 통신을 통하여 환경부 산하 에너지 관리공단 서버로 전송되고 있으며 이 전송되는 시리얼 포트에서 데이터를 취득하여 다른 서버로 전송하도록 구성합니다.



[그림 41] 에너지 관리공단 게이트웨이 연계

에너지 관리공단에서 제공하는 신재생에너지 표준 프로토콜은 다음 그림과 같습니다.

2) 응답 패킷 형태(각 에너지원 별 데이터 형태는 아래내용 참조)

가) 태양광, 태양열 생산 측, 태양열 자연 순환식, 지열 히트펌프 측, 풍력, 연료전지, ESS

① SOP	② 국번	③ Command	④ 데이터 총 길이 (length)		⑤ Data (byte)	⑥ CRC	
0x7e (고정 값)	0x01 ~ 0xnn	0x02 (응답 값)	Data Length High Byte	Data Length Low Byte	데이터	CRC High	CRC Low

나) 태양열 부하 측, 지열 부하 측

① SOP	② 국번	③ Command	④ 데이터 총 길이 (length)		⑤ Data (byte)	⑥ CRC	
0x7e (고정 값)	0x01 ~ 0xnn	0x04 (응답 값)	Data Length High Byte	Data Length Low Byte	데이터	CRC High	CRC Low

다) 지열 축열조 측

① SOP	② 국번	③ Command	④ 데이터 총 길이 (length)		⑤ Data (byte)	⑥ CRC	
0x7e (고정 값)	0x01 ~ 0xnn	0x06 (응답 값)	Data Length High Byte	Data Length Low Byte	데이터	CRC High	CRC Low

※ CRC는 SOP부터 Data까지의 CRC16-Modbus 계산을 합니다.

※ CRC16 Modbus 테이블(맨 왼쪽상단의 번호가 0번부터 ~ 255번까지입니다.)

[그림 42] 신재생에너지 표준 프로토콜

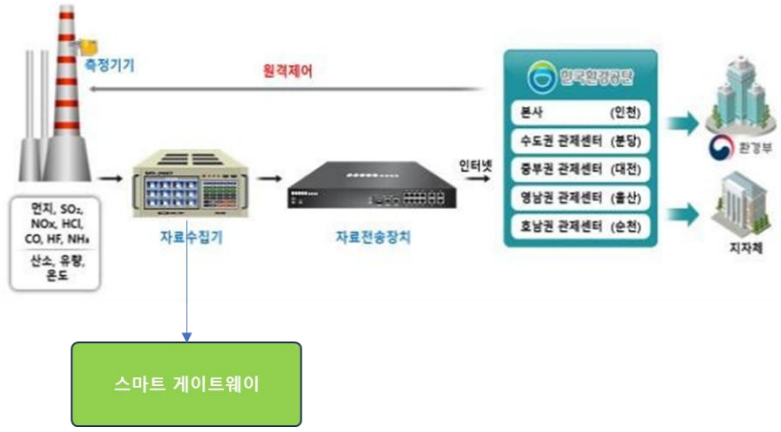
스크립트 예제는 다음과 같습니다.

```
Lib.load("serial");

var sess = Serial.open("/dev/ttyS0", "baud=9600 databit=8 parity=none");
var data = Serial.read(sess, "type=stxLen stx=0x7E len=3:7:word1");

if(typeof(data) !='undefined') {
    //Log.add(data);
    var unit = Arr.get(data, 1, "byte");
    Db.set("INV" + unit + ".PV", Arr.get(data,  5, "word1"));
    Db.set("INV" + unit + ".PA", Arr.get(data,  7, "word1"));
    Db.set("INV" + unit + ".PW", Arr.get(data,  9, "word1"));
    Db.set("INV" + unit + ".VLT", Arr.get(data, 11, "word1"));
    Db.set("INV" + unit + ".AMP", Arr.get(data, 13, "word1"));
    Db.set("INV" + unit + ".WATT", Arr.get(data, 15, "word1"));
    Db.set("INV" + unit + ".PF", Arr.get(data, 17, "word1") / 10);
    Db.set("INV" + unit + ".FREQ", Arr.get(data, 19, "word1") / 10);
    Db.set("INV" + unit + ".KWH", Arr.get(data, 21, "long1") / 1000);
    Db.set("INV" + unit + ".FAULT", Arr.get(data, 29, "word1") / 1000);
}
```

[예시 47] Serial: 신생 에너지 연계

시리얼통신을 이용하여 환경부 대기오염관리 시스템 연계

대기환경 관리를 위하여 오염 발생현장의 설비로부터 값을 측정하여 환경부로 전송하기 위한 시스템에서 별도의 경로를 이용하여 데이터를 취득하기 위한 구성은 다음과 같습니다.



[그림 43] 대기 오염 관리 시스템으로부터 데이터 취득

환경부에서 관리하는 대기오염 측정설비의 프로토콜을 다음과 같습니다.

STX (1)	사업장정보						
	report-ID (4)	*지역코드 (2)	*사업장코드 (4)	*굴뚝번호 (3)	DCD (1)	측정항목수 (2)	*측정일시 (12)
측정자료							
측정자료0							
평균자료0							
*측정항목 (3)	측정값(보정후) (7)	측정값(보정전) (7)	측정기기 status (1)	DL status (1)	보정여부 (1)		

측정자료									
측정자료0									
상태정보 자료수 (2)	상태정보0		상태정보1	…	알람정보 자료수 (2)	알람정보0		알람정보1	…
	상태정보명 (4)	상태정보 (7)				알람정보명 (4)	알람정보 (1)		

측정자료		*보안코드 (12)	ETX (1)	CHK (2)	CR (1)
측정자료1	…				

( ) : byte수, * : 공통코드 적용항목

[그림 44] 대기오염 시스템 표준 프로토콜

환경부 대기오염 시스템과의 연결 스크립트는 다음과 같습니다.

```
var updateValue = function(arr, key, tag) {
    var pos = Arr.find(arr, "key");
    if(pos != -1) {
    var val = Arr.getAsc(arr, pos + 3, 7);
        Db.set(tag, Number(val));
    }
}

Lib.load("serial");

var sess = Serial.open("/dev/ttyS0", "baud=9600 databit=8 parity=none");
```

```
var data = Serial.read(sess, "type=none");

if(typeof(data) !='undefined') {
    if(data[0] == 0x05) {            // ENQ수신하면 ACK전송
        Serial.write(sess, [0x06]);   // ACK전송
        var nFail = 0;                // 데이터 수신 루프
        while(true) {
            Serial.wait(500);          // 500 msec대기 후 데이터 읽기 시도
            data = Serial.read(sess, "type=none");
            if(typeof(data) !='undefined') {
                if(data[0] == 0x04 || data[0] == 0x05) break;
                // ENQ 혹은 EOT 수신하면 세션 종료
                updateValue (data, "TSP", "SLC.1.TSP");
                updateValue (data, "NOX", "SLC.1.NOX");
                Serial.write(sess, [0x06]); // ACK 전송
            }
            else {
               if(++nFail >= 5) break;
               Serial.write(sess, [0x15]); // NAK 전송
            }
        }
    }
}
```

[예시 48] Serial: 환경부 대기오염관리 시스템 연계

7.3. Modbus

시리얼통신을 이용하여 이더넷 Modbus 서버연계

Modbus/TCP 로 운영하고 있는 Modbus 설비를 시리얼 통신을 이용하여 연계하고자 할 경우 사용 가능한 구성으로 시리얼 구간에는 Modbus/RTU master 로 연결되고 이더넷 구간은 Modbus/TCP Server 로 연결된 경우 아래와 같은 스크립트만으로 두 설비 간의 연동이 가능하게 됩니다.

[그림 45] Modbus/RTU 게이트웨이

```
Lib.load("mbrtutcp");
MBrtutcp.openTcp("192.168.10.1", 502);
MBrtutcp.openTcp("192.168.10.10", 502);
MBrtutcp.openRtu("/dev/ttyUSB0", "baud=9600 databit=8 parity=none");
MBrtutcp.mapTo(1, "192.168.1.1");
MBrtutcp.mapTo(2, "192.168.1.10");
```

[예시 49] MBrtutcp: Serial 통신을 이용한 이더넷 Modbus 서버 연계

위의 예제는 하나의 스마트 게이트웨이를 이용하여 두 대의 Modbus/TCP 서버를 연결하기 위한 예제로 슬레이브 1 번의 요청은 IP 주소 192.168.1.1 서버로 전송되고 슬레이브 2 번의 요청은 IP 주솧 192.168.1.10 으로 전달되도록 구성됩니다.

이더넷 통신을 이용하여 시리얼 Modbus Slave 연계

Modbus/RTU 로 운영하고 있는 Modbus 설비를 이더넷 통신을 이용하여 연계하고자 할 경우 사용 가능한 구성으로 시리얼구간에는 Modbus/RTU slave 로 연결되고 이더넷 구간은 Modbus/TCP client 로 연결된 경우 아래와 같은 스크립트만으로 두 설비 간의 연동이 가능하게 됩니다.

[그림 46] Modbus/TCP 게이트웨이

```
Lib.load("mbtcprtu");
MBtcprtu.openTcp(502);
MBtcprtu.openRtu("/dev/ttyUSB0", "baud=9600 databit=8 parity=none", 2);
```

[예시 50] MBtcprtu: 이더넷 통신을 이용한 Serial Modbus Slave 연계

위의 예제에서 Modbus/RTU 디바이스가 slave-no 2 를 갖고 있는 것으로 가정하여 구성한 사례이며 Modbus/RTU 디바이스가 1 인 경우 openRtu() 함수의 3 번째 값을 1 로 지정하여 운용합니다.

Modbus/TCP server 로부터 데이터 수집

Modbus/TCP server 장치와 연계하여 데이터를 읽어오거나 데이터를 전송하기 위한 예제입니다.

```
Lib.load("mbclient");

// modbus/tcp server와의 연결
var sess = MBclient.open("192.168.10.50", 502);

// 데이터 읽기
var values = MBclient.read(sess, 40100, 50);

// 정수형(2 bytes) 데이터 쓰기
MBclient.write(sess, 16, 0, [256, 564, 678]);

// 정수형(4 bytes) 데이터 쓰기
MBclient.write(sess, 16, 0, [655256, 5248564, 789678], "int");

// 실수형(4 bytes) 데이터 쓰기
MBclient.write(sess, 16, 0, [1.5, 3.8, 6.5], "float");
```

[예시 51] MBclient: Modbus/TCP server 로부터 데이터 수집

Modbus/RTU slave 로부터 데이터 수집

Modbus/RTU slave 장치와 연계하여 데이터를 읽어오거나 데이터를 전송하기 위한 예제입니다.

```
Lib.load("mbmaster");

// modbus/rtu slave와의 연결
var sess = MBmaster.open("/dev/ttyUSB0", "baud=9600 databit=8 parity=none");

// 데이터 읽기-Input register 30101번지부터 50개의 WORD자료를 읽어오기
var values = MBmaster.read(sess, 1, 40100, 50);

// 정수형(2 bytes) 데이터 쓰기-Holding Register 40017부터 3 word값 쓰기
MBmaster.write(sess, 1, 16, 0, [256, 564, 678]);

// 정수형(4 bytes) 데이터 쓰기 – Holding Register 40017부터 6 word값 쓰기
MBmaster.write(sess, 2, 16, 0, [655256, 5248564, 789678], "int");

// 실수형(4 bytes) 데이터 쓰기 – Holding Register 40017부터 IEEE754로 변환된 6 word값 쓰기
MBmaster.write(sess, 5, 16, 0, [1.5, 3.8, 6.5], "float");
```

[예시 52] MBmaster: Modbus/RTU slave 로부터 데이터 수집

Modbus/RTU slave 서비스

데이터 저장소에 저장된 데이터는 modbus/RTU slave 서비스를 제공할 수 있습니다.

```
Lib.load("mbslave");

// modbus/rtu slave 시작
MBslave.open("/dev/ttyUSB0", "baud=2400 databit=8 parity=none");

// 난수를 발생하여 Modbus/RTU서비스 제공
// 발생된 난수는 40001부터 4바이트씩 실수형 데이터가 저장
// 저장된 값은 Modbus/RTU 연계를 통하여 서비스 진행
for(var i = 0; i < 10; i++) {
   var val = Math.random() * 100;
   Db.setModbus(40001 + i * 2, "float", val);
}
```

[예시 53] MBslave: Modbus/RTU slave 서비스

Modbus/TCP server 서비스

데이터 저장소에 저장된 데이터는 modbus/TCP Server 서비스를 제공할 수 있습니다.

```
Lib.load("mbserver");

// modbus/tcp server 시작
MBserver.open(502);

// 난수를 발생하여 Modbus/TCP서비스 제공
// 발생된 난수는 40001부터 4바이트씩 실수형 데이터가 저장
// 저장된 값은 modbus/tcp 연계를 통하여 서비스 진행
for(var i = 0; i < 10; i++) {
   var val = Math.random() * 100;
   Db.setModbus(40001 + i * 2, "float", val);
}
```

[예시 54] MBserver: Modbus/TCP server 서비스

7.4. 표준 프로토콜

OPC-UA to Modbus/RTU

OPC UA Server 와의 연결을 통하여 데이터를 수집하고 수집된 데이터를 Modbus/RTU 로 서비스하기 위한 구성 예제입니다.

[그림 47] Modbus/RTU-OPC UA 게이트웨이

```
var mapper = [];        // OPC UA nodeID를 모드버스주소로 변환하기 위한 매퍼

// OPC UA로부터 데이터 수집을 위한 NodeId 등록과 modbus 주소 연계작업
var init = function() {
    var nodeIds = [ "N3|1002", "N3|1003", "N3|1006" ];
    var modbusAddrs = [ 40001, 40003, 40005 ];

    for(var idx in nodeIds)  {
        UAclient.addItem(nodeIds[idx]);
        mapper[nodeIds[idx]] = modbusAddrs[idx];
    }
}

// Modbus/RTU slave 드라이버를 로딩하고 서비스를 시작
Lib.load("mbslave");
MBslave.open("/dev/ttyUSB0", "baud=2400 databit=8 parity=none");

// OPC UA client 드라이버를 로딩
Lib.load("uaclient");
init();

// OPC UA서버와 연결
UAclient.open("opc.tcp://192.168.1.135:53530/OPCUA/SimulationServer");

// 데이터 읽어오기
var values = UAclient.read();

// 수신된 데이터를 Modbus 영역으로 전송
for(var idx in values) {
    var val = values[idx];
    Db.setModbus(mapper[val.name], "float", val.val );
}
```

[예시 55] UAclient: OPC-UA to Modbus/RTU

그림 목차

[그림 1] 스마트 게이트웨이 역할 5
[그림 2] 스마트 게이트웨이 모듈 구성 6
[그림 3] 로그인 화면 8
[그림 4] 사용자-시스템 상태감시 9
[그림 5] 사용자-태그관리 10
[그림 6] 사용자-스크립트관리 11
[그림 7] 사용자-스크립트관리 11
[그림 8] 관리자-시스템 상태감시화면 12
[그림 9] 관리자-스마트게이트웨이 연결주소 변경 13
[그림 10] 관리자-동기화 정보 변경 13
[그림 11] 관리자-동기화 정보 변경 14
[그림 12] 태그관리 화면 15
[그림 13] 스크립트 관리 화면 16
[그림 14] 관리자 모드 - 스크립트 등록 화면 17
[그림 15] 관리자 모드 - 스크립트 삭제 화면 18
[그림 16] 로그관리 화면 18
[그림 17] 장치로부터 읽기 19
[그림 18] 장치로 쓰기 19
[그림 19] 기능 – 간접쓰기 20
[그림 20] Big-endian vs. Little-Endian 29
[그림 21] 시리얼 통신 파라메터 34
[그림 22] 파서 및 항목정의 35
[그림 23] 시리얼 통신 파라메터 39
[그림 24] 시리얼 통신 파라메터 40
[그림 25] Modbus/RTU-TCP 게이트웨이 42
[그림 26] 시리얼 통신 파라메터 43
[그림 27] slave 번호에 따른 Modbus Server 연계 44
[그림 28] Modbus/TCP-RTU 게이트웨이 45
[그림 29] 시리얼 통신 파라메터 46
[그림 30] Modbus TCP-RTU 다중변환 47
[그림 31] 시리얼 통신 파라메터 51
[그림 32] 사용가능한 디바이스 이름 52
[그림 33] MQclient 통신 파라메터 53
[그림 34] 디바이스 일람 55
[그림 35] 사용 가능한 디바이스 이름 58

[그림 36] S7client 통신 파라메터 60
[그림 37] 지멘스 PLC 디바이스변수 구성 61
[그림 38] S7client 통신 파라메터 63
[그림 39] Panasonic 조명제어 PLC 66
[그림 40] JDBC 드라이버를 이용한 데이터베이스 연결 79
[그림 41] 에너지 관리공단 게이트웨이 연계 80
[그림 42] 신재생에너지 표준 프로토콜 81
[그림 43] 대기 오염 관리 시스템으로부터 데이터 취득 82
[그림 44] 대기오염 시스템 표준 프로토콜 82
[그림 45] Modbus/RTU 게이트웨이 83
[그림 46] Modbus/TCP 게이트웨이 84
[그림 47] Modbus/RTU-OPC UA 게이트웨이 86

표 목차

[표 1] 로그인 암호 9
[표 2] 동기화 모드 14
[표 3] 스크립트 실행간격 예제 17
[표 4] 스크립트 실행간격 예제 21
[표 5] 날짜 포맷 함수 26
[표 6] Type 종류 및 설명 28
[표 7] 파서 종류 및 설명 35
[표 8] 데이터 타입 및 사용 예 50
[표 9] Parameter 종류 및 설명 51
[표 10] 파라메터 종류 및 설명 57
[표 11] siemens 변수 구성방법 61
[표 12] Parameter 종류 및 설명 65
[표 13] NCUmaster.read() Command 목록 67
[표 14] NCUmaster.write() Command 목록 68
[표 15] NCUmaster.check() Command 목록 69
[표 16] NCUmaster.set() Command 목록 71
[표 17] NCUmaster.set() 설정 값 설명 1 72
[표 18] NCUmaster.set() 설정 값 설명 2 73
[표 19] NCUmaster.set() 설정 값 설명 3 74
[표 20] NCUmaster.set() 설정 값 설명 4 74
[표 21] NCUmaster.set() 설정 값 설명 5 75

[표 22] NCUmaster.set() 설정 값 설명 6....75
[표 23] LCSRest.read() Parameter 설명....77
[표 24] LCSRest.read() 반환 값 설명....77

예시 목차

[예시 1] pad()....26
[예시 2] Arr.get()....26
[예시 3] Arr.find()....27
[예시 4] Arr.get()....27
[예시 5] Arr.getAsc()....27
[예시 6] Jdbc.open()....30
[예시 7] Jdbc Driver....31
[예시 8] Jdbc.open()....32
[예시 9] Serial.open(): Communication-param....33
[예시 10] Serial Driver....34
[예시 11] MBmaster.open(): Communication-param....37
[예시 12] MBslave.open(): communication-param....39
[예시 13] MBrtutcp.openRtu(): communication-param....43
[예시 14] MBrtutcp.openRtu(): communication-param....45
[예시 15] UAclient.open()....48
[예시 16] UAclient.read()....48
[예시 17] LSclient.open()....48
[예시 18] LSclient.read()....49
[예시 19] LSclient.write()....49
[예시 20] LSclient Driver....50
[예시 21] LSmaster.open(): communication-param....51
[예시 22] LSmaster Driver....53
[예시 23] MQclient.open()....54
[예시 24] MQclient.read(.., device, count, type)....54
[예시 25] MQclient.read(.., variables)....54
[예시 26] MQclient.write()....55
[예시 27] MQclient Driver....56
[예시 28] FXmaster.open(): communication-param....57
[예시 29] FXmaster Driver....59
[예시 30] S7client.open()....60

[예시 31] S7client.read() 60
[예시 32] S7client.write() 61
[예시 33] S7client 통신드라이버 사용예 62
[예시 34] EIPclient.open() 63
[예시 35] EIPclient.read() 63
[예시 36] EIPclient.write() 64
[예시 37] EIPclient Driver 사용 예제 64
[예시 38] NCUmaster.open(): communication-param 65
[예시 39] NCUmaster.read() 예시 67
[예시 40] NCUmaster.write() 예시 68
[예시 41] NCUmaster.check() 예시 69
[예시 42] NCUmaster.set() 예시 71
[예시 43] NCUmaster 드라이버 Script 작성 예시 76
[예시 44] LCSRest Driver 사용예제 78
[예시 45] JDBC 사용예제 79
[예시 46] Restful API: 공공데이터 포털 데이터 연계 80
[예시 47] Serial: 신생 에너지 연계 81
[예시 48] Serial: 환경부 대기오염관리 시스템 연계 83
[예시 49] MBrtutcp: Serial 통신을 이용한 이더넷 Modbus 서버 연계 84
[예시 50] MBtcprtu: 이더넷 통신을 이용한 Serial Modbus Slave 연계 84
[예시 51] MBclient: Modbus/TCP server 로부터 데이터 수집 85
[예시 52] MBmaster: Modbus/RTU slave 로부터 데이터 수집 85
[예시 53] MBslave: Modbus/RTU slave 서비스 86
[예시 54] MBserver: Modbus/TCP server 서비스 86
[예시 55] UAclient: OPC-UA to Modbus/RTU 87