■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[] is range over() |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.4. dio.reset_bits : 다중 디지털 출력 신호 비활성화(ON)

| | |
|---|---|
| ■ 기 능 | **다중 포트(Port)의 디지털 출력 신호를 비활성화 합니다.** |
| ■ 형 태 | dio.reset_bits(Port) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | uint32_t | - | 0x0000 | 0xFFFFFFFFFFFF | - |

■ **세부사항**

➢ **Port (복수 개 I/O를 출력하고자 하는 접점 list)**

- 인자의 경우, 16진수로 표현합니다.
- 1~16번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 17~48번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 16개의 접점으로 구성되어 있습니다.

➢ **16진수 표현 방법**

- 그림 2-67과 같이 16진수의 한자리는 4bit에 해당 됩니다.
- 16진수로 설정한 해당 bit의 값은 1로 설정되며, 해당 값은 디지털 출력 신호를 설정합니다.

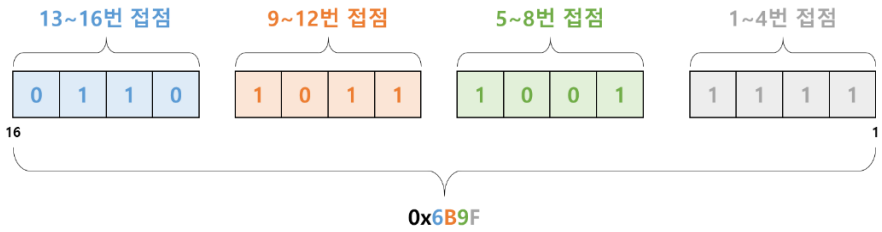



**그림 2-68 16진수와 bits 간의 관계**

■ 예 제

**[예제1] dio.set_bits 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | dio.set_bits(0x0031) | -- 디지털 출력 1번, 5번, 6번 활성화 |
| 2 | dio.reset_bits(0x0011) | -- 디지털 출력 1번, 5번 비활성화 |
| 3 | dio.set_bits(0x103000200000) | -- 디지털 출력 22번, 37번, 38번 45번 활성화 |
| 4 | | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[] is range over() |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.5. dio.get : 디지털 입력 값 가져오기

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 디지털 입력 값을 가져옵니다.** |
| ■ 형 태 | dio.get(Port) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 48 | - |

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 디지털 입력 값을 가져옵니다.
- 1~16번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 17~48번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 16개의 접점으로 구성되어 있습니다.

■ 반환값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Return | int | - | 0 | 1 | - |

➢ **Return (반환 값)**

- 해당 포트(Port) 입력된 값을 반환합니다. 신호가 입력되었을 경우 1, 그렇지 않을 경우 0을 반환합니다.

■ 예 제

**[예제1] dio.get 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | result = dio.get(2) | -- 입력 2번 신호 값 변수에 저장 |
| 2 | | |
| 3 | if result == 1 then | -- 입력 신호값 비교문 |
| 4 | dio.set(1) | -- 신호값이 1일 경우 출력 1번 On |
| 5 | else | |
| 6 | dio.set(5) | -- 신호값이 0일 경우 출력 5번 On |
| 7 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[50] is range over(1~48) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.5.6. dio.in_wait : 디지털 입력 신호 대기

| | |
|---|---|
| ■ 기 능 | **해당 입력 포트(Port)가 설정된 값(Value)이 될 때 까지 대기합니다.** |
| ■ 형 태 | dio.in_wait(Port, Value, [Time]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 48 | - |
| Value | int | - | 0 | 1 | - |
| [Time] | int | - | 0 | 100000 | 10ms |

* 단위 : ms (milliesecond)

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 디지털 입력 신호를 비교합니다.
- 1~16번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 17~48번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 16개의 접점으로 구성되어 있습니다.

➢ **Value (설정 값)**

- 입력 포트의 설정 값이며, 0 또는 1을 설정합니다.
- 0을 설정하는 경우, 입력 신호가 비활성화된 상태를 의미하며, 1을 설정하는 경우, 입력 신호가 활성화된 상태를 의미합니다.

➢ **[Time] (시간 값)**

- 해당 인자는 옵션 인자입니다.
- 시간을 지정하여 해당 시간 동안 입력 신호의 값을 기다립니다.
- 시간 값을 지정 할 경우, 입력 신호가 설정된 값에 도달하였는지 결과값을 반환합니다.
- 시간 지정을 하지 않을 경우, 해당 포트(Port)에 설정 값(Value)이 입력 될 때까지 무기한 대기합니다.
- 지정된 시간이 지난 경우, 디지털 이벽 신호 대기를 종료합니다.

■ 반환 값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Return | int | - | 0 | 1 | - |

➢ **Return (반환 값)**

- 지정 시간(Time) 내에 해당 포트(Port)에 설정한 값(Value)이 입력 되었는지에 대한 결과 값을 반환합니다.
- 해당 값은 지정 시간(Time) 옵션을 입력하였을 경우에만 결과 값을 반환합니다.

■ 예 제

**[예제1] dio.in_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | dio.in_wait(1, 1) | -- 입력 1번 포트에 신호가 활성화 될 때 까지 대기 |
| 2 | | |
| 3 | dio.set(1) | -- 입력 신호가 활성화 되면 출력 1번 포트 활성화 |
| 4 | | |

**[예제2] dio.in_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | ret = dio.in_wait(1, 1, 20) | -- 2초 동안 입력 1번 포트에 신호가 활성화 될 때 까지 대기 결과 값을 변수에 저장 |
| 2 | | |
| 3 | | |
| 4 | if ret == 1 then | |
| 5 | dio.set(1) | -- 입력 신호가 활성화 되었을 경우 출력 1번 활성화 |
| 6 | else | |
| 7 | dio.set(2) | -- 입력 신호가 활성화 되지 않았을 경우 출력 2번 활성화 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[50] is range over(1~48) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.5.7. dio.out_wait : 디지털 출력 신호 대기

| | |
|---|---|
| ■ 기 능 | **해당 출력 포트(Port)가 설정된 값(Value)이 될 때 까지 대기합니다.** |
| ■ 형 태 | dio.out_wait(Port, Value, [Time]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 48 | - |
| Value | int | - | 0 | 1 | - |
| [Time] | int | - | 0 | 100000 | 10ms |

* 단위 : ms (milliesecond)

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 디지털 출력 신호를 비교합니다.
- 1~16번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 17~48번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 16개의 접점으로 구성되어 있습니다.

➢ **Value (설정 값)**

- 출력 포트의 설정 값이며, 0 또는 1을 설정합니다.
- 0을 설정하는 경우, 출력 신호가 비활성화된 상태를 의미하며, 1을 설정하는 경우, 출력 신호가 활성화된 상태를 의미합니다.

➢ **[Time] (시간 값)**

- 해당 인자는 옵션 인자입니다.
- 시간을 지정하여 해당 시간 동안 출력 신호의 값을 기다립니다.
- 시간 값을 지정 할 경우, 출력 신호가 설정된 값에 도달하였는지 결과값을 반환합니다.
- 시간 지정을 하지 않을 경우, 해당 포트(Port)에 설정 값(Value)이 출력 될 때까지 무기한 대기합니다.
- 지정된 시간이 지난 경우, 디지털 이벽 신호 대기를 종료합니다.

■ 반환 값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Return | int | - | 0 | 1 | - |

➢ **Return (반환 값)**

- 지정 시간(Time) 내에 해당 포트(Port)에 설정한 값(Value)이 출력 되었는지에 대한 결과 값을 반환합니다.
- 해당 값은 지정 시간(Time) 옵션을 입력하였을 경우에만 결과 값을 반환합니다.

■ 예 제

**[예제1] dio.out_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | dio.out_wait(1, 1) | -- 출력 1번 포트에 신호가 활성화 될 때 까지 대기 |
| 2 | | |
| 3 | dio.set(5) | -- 출력 신호가 활성화 되면 출력 5번 포트 활성화 |
| 4 | | |

**[예제2] dio.out_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | ret = dio.out_wait(1, 1, 20) | -- 2초 동안 출력 1번 포트에 신호가 활성화 될 때 까지 대기 결과 값을 변수에 저장 |
| 2 | | |
| 3 | | |
| 4 | if ret == 1 then | |
| 5 | dio.set(5) | -- 출력 신호가 활성화 되었을 경우 출력 5번 활성화 |
| 6 | else | |
| 7 | dio.set(6) | -- 출력 신호가 활성화 되지 않았을 경우 출력 6번 활성화 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[50] is range over(1~48) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.5.8. dio.pulse : 디지털 단일 펄스 출력

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 출력 값(Value)를 지정 시간(Time)동안 단일 펄스 출력합니다.** |
| ■ 형 태 | dio.pulse(Port, Value, [Time]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 48 | - |
| Value | int | - | 0 | 1 | - |
| [Time] | int | - | 0 | 100000 | 10ms |

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 디지털 출력 신호를 비교합니다.
- 1~16번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 17~48번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 16개의 접점으로 구성되어 있습니다.

➢ **Value (설정 값)**

- 출력 포트의 설정 값이며, 0 또는 1을 설정합니다.
- 0을 설정하는 경우, 출력 신호가 비활성화된 상태를 의미하며
  1을 설정하는 경우, 출력 신호가 활성화된 상태를 의미합니다.

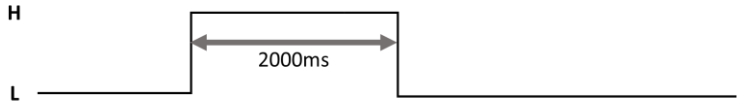
➢ **[Time] (시간 값)**

- 해당 인자는 옵션 인자입니다.
- 시간을 지정하여 해당 시간 동안 출력 값(Value)를 출력합니다.
- 시간을 0으로 지정할 경우, 출력 값(Value)를 상시 출력합니다.

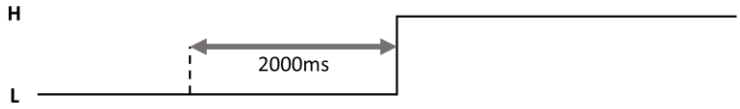
■ 설명

- pulse 명령어가 호출될 경우, 지정 시간(Time) 동안 해당 포트(Port)에 출력 값(Value)를 출력하며, 지정 시간 경과 후에 출력 값에 반대된 값으로 신호가 토글 됩니다.

**dio.pulse(1, 1, 2000)**




**dio.pulse(1, 0, 2000)**



ⓘ **pulse 명령어가 호출될 경우, 프로그램 실행, 일시정지, 정지와 관계없이 독립적으로 수행됩니다.**

ⓘ **pulse 명령어는 비상정지에 영향을 받지 않습니다.**

■ **예 제**

**[예제1] dio.pulse 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | ret = dio.in_wait(1, 1, 2000) | -- 2초 동안 출력 1번 포트에 신호가 활성화 될 때 까지 대기 결과 값을 변수에 저장 |
| 2 | | |
| 2 | if ret == 1 then | |
| 3 | dio.pulse(5, 1, 2000) | -- 입력 신호가 활성화 되었을 경우 출력 5번 2초 동안 펄스 출력 |
| 4 | else | |
| 5 | dio.pulse(6, 0, 2000) | -- 입력 신호가 활성화 되지 않았을 경우 출력 6번 2초 동안 펄스 출력 |
| 6 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[3] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[50] is range over(1~48) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.9. aio.set : 아날로그 출력 신호 설정

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 Analog I/O 신호 출력 전압 값(Voltage)을 설정합니다.** |
| ■ 형 태 | aio.set(Port, Voltage) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 6 | - |
| Voltage | double | - | 0.0 | 10.0 | V |

* 단위 : V (voltage)

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 아날로그 출력 전압 값을 설정 합니다.
- 1~2번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 3~6번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 2개의 접점으로 구성되어 있습니다.

➢ **Voltage (출력 값 설정)**

- 해당 포트의 출력 전압 값을 설정합니다.

■ 예 제

**[예제1] aio.set 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | aio.set(1, 5.2) | -- 1번 아날로그 5.2V 출력 |
| 2 | delay(1000) | -- 1000ms 대기 (=1sec) |
| 3 | aio.set(1, 0) | -- 1번 아날로그 0V 출력 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[7] is range over(1~6) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.5.10. aio.get : 아날로그 출력 신호 가져오기

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 Analog I/O 신호 출력 전압 값(Voltage)을 가져옵니다.** |
| ■ 형 태 | aio.get(Port) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 6 | - |

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 아날로그 출력 전압 값을 가져옵니다.
- 1~2번 포트는 기본 제공 포트이며, 확장 IO를 연결 할 경우, 3~10번 포트를 사용 할 수 있습니다.
- 확장 IO는 모듈당 2개의 접점으로 구성되어 있습니다.

■ 반환값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Voltage | double | - | 0 | 10 | V |

* 단위 : V (voltage)

➢ **Return (반환 값)**

- 해당 포트(Port) 입력된 아날로그 신호값을 반환합니다.

■ 예 제

**[예제1] aio.get 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | a=aio.get(2) | -- 아날로그 입력 2번 포트의 신호 |
| 2 | | 값을 변수에 저장 |
| 3 | if a > 5 then | -- 입력 신호가 5V 보다 큰지 비교 |
| 4 | dio.set(1) | -- 5V 보다 클 경우, 출력 1번 활성화 |
| 5 | else | |
| 6 | dio.set(2) | -- 5V 보다 작거나 같을 경우, 출력 |
| 7 | end | 2번 활성화 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[11] is range over(1~10) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.5.11. tool.set : 툴 출력 신호 활성화(ON)

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 Tool I/O 출력 신호를 활성화(ON)합니다.** |
| ■ 형 태 | tool.set(Port) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 4 | - |

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 툴 출력 신호를 활성화합니다.

■ 예 제

**[예제1] tool.set 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | tool.set(1) | -- 툴 출력 1번 활성화 |
| 2 | delay(1000) | -- 1000ms 대기 (=1sec) |
| 3 | tool.reset(1) | -- 툴 출력 1번 비활성화 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[5] is range over(1~4) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.12. tool.reset : 툴 출력 신호 비활성화(OFF)

| | |
|---|---|
| ■ 기 능 | **설정한 포트(Port)의 Tool I/O 출력 신호를 OFF 합니다.** |
| ■ 형 태 | tool.reset(Port) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 4 | - |

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 툴 출력 신호를 비활성화 합니다.

■ 예 제

**[예제1] tool.reset 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | tool.set(1) | -- 툴 출력 1번 활성화 |
| 2 | delay(1000) | -- 1000ms 대기 (=1sec) |
| 3 | tool.reset(1) | -- 툴 출력 1번 비활성화 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[5] is range over(1~4) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.13. tool.get : 툴 입력 값 가져오기

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 툴 입력 값을 가져옵니다.** |
| ■ 형 태 | tool.get(Port) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 2 | - |

■ **세부사항**

➢ **Port (Tool I/O 접점 번호)**

- 해당 포트의 툴 입력 값을 가져옵니다.

■ 반환값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Return | int | - | 0 | 1 | - |

➢ **Return (반환 값)**

- 해당 포트(Port) 입력된 값을 반환합니다. 신호가 입력되었을 경우 1, 그렇지 않을 경우 0을 반환합니다.

■ 예 제

**[예제1] tool.get 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | result = tool.get(2) | -- 입력 2번 신호 값 변수에 저장 |
| 2 | | |
| 3 | if result == 1 then | -- 입력 신호값 비교문 |
| 4 | tool.set(1) | -- 신호값이 1일 경우 출력 1번 On |
| 5 | else | |
| 6 | tool.set(2) | -- 신호값이 0일 경우 출력 2번 On |
| 7 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[3] is range over(1~2) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.5.14. tool.in_wait : 툴 입력 신호 대기

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)가 설정된 툴 신호 입력 값(Value)이 될 때까지 대기합니다.** |
| ■ 형 태 | tool.in_wait(Port, Value, [Time]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 4 | - |
| Value | int | - | 0 | 1 | - |
| [Time] | int | - | 0 | 100000 | 10ms |

* 단위 : ms (milliesecond)

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 툴 입력 신호를 비교합니다.

➢ **Value (설정 값)**

- 입력 포트의 설정 값이며, 0 또는 1을 설정합니다.
- 0을 설정하는 경우, 입력 신호가 비활성화된 상태를 의미하며, 1을 설정하는 경우, 입력 신호가 활성화된 상태를 의미합니다.

➢ **[Time] (시간 값)**

- 해당 인자는 옵션 인자입니다.
- 시간을 지정하여 해당 시간 동안 입력 신호의 값을 기다립니다.
- 시간 값을 지정 할 경우, 입력 신호가 설정된 값에 도달하였는지 결과값을 반환합니다.
- 시간 지정을 하지 않을 경우, 해당 포트(Port)에 설정 값(Value)이 입력 될 때까지 무기한 대기합니다.

■ 반환 값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Return | int | - | 0 | 1 | - |

➢ **Return (반환 값)**

- 지정 시간(Time) 내에 해당 포트(Port)에 설정한 값(Value)이 입력 되었는지에 대한 결과 값을 반환합니다.
- 해당 값은 지정 시간(Time) 옵션을 입력하였을 경우에만 결과 값을 반환합니다.

■ 예 제

**[예제1] tool.in_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | tool.in_wait(2, 1) | -- 툴 2번 포트에 입력 신호가 활성화 |
| 2 | | 될 때 까지 대기 |
| 3 | dio.set(1) | -- 툴 입력 신호가 활성화 되면 출력 |
| 4 | | 1번 포트 활성화 |

**[예제2] tool.in_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | ret = tool.in_wait(1, 1, 20) | -- 2초 동안 툴 입력 1번 포트에 신호가 |
| 2 | | 활성화 될 때 까지 대기 결과 값을 |
| 3 | | 변수에 저장 |
| 4 | if ret == 1 then | |
| 5 | dio.set(1) | -- 툴 입력 신호가 활성화 되었을 경우 |
| 6 | else | 출력 1번 활성화 |
| 7 | dio.set(2) | -- 툴 입력 신호가 활성화 되지 않았을 |
| 8 | end | 경우 출력 2번 활성화 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[5] is range over(1~2) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.15. tool.out_wait : 툴 출력 신호 대기

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)가 설정된 툴 신호 출력 값(Value)이 될 때까지 대기합니다.** |
| ■ 형 태 | tool.out_wait(Port, Value, [Time]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 1 | 4 | - |
| Value | int | - | 0 | 1 | - |
| [Time] | int | - | 0 | 100000 | 10ms |

* 단위 : ms (milliesecond)

■ **세부사항**

➢ **Port (제어기의 I/O 접점 번호)**

- 해당 포트의 툴 출력 신호를 비교합니다.

➢ **Value (설정 값)**

- 출력 포트의 설정 값이며, 0 또는 1을 설정합니다.
- 0을 설정하는 경우, 출력 신호가 비활성화된 상태를 의미하며, 1을 설정하는 경우, 출력 신호가 활성화된 상태를 의미합니다.

➢ **[Time] (시간 값)**

- 해당 인자는 옵션 인자입니다.
- 시간을 지정하여 해당 시간 동안 출력 신호의 값을 기다립니다.
- 시간 값을 지정 할 경우, 출력 신호가 설정된 값에 도달하였는지 결과값을 반환합니다.
- 시간 지정을 하지 않을 경우, 해당 포트(Port)에 설정 값(Value)이 출력 될 때까지 무기한 대기합니다.

■ 반환 값

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Return | int | - | 0 | 1 | - |

➢ **Return (반환 값)**

- 지정 시간(Time) 내에 해당 포트(Port)에 설정한 값(Value)이 입력 되었는지에 대한 결과 값을 반환합니다.
- 해당 값은 지정 시간(Time) 옵션을 입력하였을 경우에만 결과 값을 반환합니다.

■ 예 제

**[예제1] tool.out_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | tool.out_wait(2, 1) | -- 툴 2번 포트에 출력 신호가 활성화 |
| 2 | | 될 때 까지 대기 |
| 3 | dio.set(1) | -- 툴 출력 신호가 활성화 되면 출력 |
| 4 | | 1번 포트 활성화 |

**[예제2] tool.out_wait 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | ret = tool.out_wait(1, 1, 20) | -- 2초 동안 툴 출력 1번 포트에 신호가 |
| 2 | | 활성화 될 때 까지 대기 결과 값을 |
| 3 | | 변수에 저장 |
| 4 | if ret == 1 then | |
| 5 | dio.set(1) | -- 툴 출력 신호가 활성화 되었을 경우 |
| 6 | else | 출력 1번 활성화 |
| 7 | dio.set(2) | -- 툴 출력 신호가 활성화 되지 않았을 |
| 8 | end | 경우 출력 2번 활성화 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[5] is range over(1~2) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.16. tool.voltage : 툴 출력 전압 값 설정

| | |
|---|---|
| ■ 기 능 | **해당 포트(Port)의 툴 신호 출력 전압 값(Value)을 설정합니다.** |
| ■ 형 태 | tool.voltage(Value) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Value | int | - | 0 | 24 | V |

* 단위 : V (voltage)

■ **세부사항**

➢ **Value (출력 값 설정)**

- 해당 포트의 출력 전압 값을 설정합니다.
- 출력은 0, 12, 24V 세 가지 출력을 지원합니다.
- 기본 출력값은 24V입니다.

■ 예 제

**[예제1] tool.voltage 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | tool.voltage(24) | -- 툴 출력 전압 24V 설정 |
| 2 | | |
| 3 | dio.voltage(12) | -- 툴 출력 전압 12V 설정 |
| 4 | | |
| 5 | dio.voltage(0) | -- 툴 출력 전압 0V 설정 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Invalid output voltage |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.17. tool.modbus_write_register : Holding Register에 값 쓰기

■ 기 능

**해당 주소의 Holding Register에 값을 씁니다.**

■ 형 태

tool.modbus_write_register(Address, Value)

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsigned short | - | 0 | 65,535 | |
| Value | unsigned short | - | 0 | 65,535 | |

■ **세부사항**

➢ **Address (Holding Register 주소)**

- 값을 쓸 Holding Register의 주소를 입력합니다.

➢ **Value (입력 값)**

- 지정한 주소의 Holding Register에 쓸 값을 입력합니다.

■ 예 제

**[예제1] tool.modbus_write_register 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1<br>2 | tool.modbus_write_register(0x00, 700) | -- 0x00(16진수) 주소에 있는 Holding Register에 700 값 쓰기 |
| 3<br>4 | tool.modbus_write_register (3, 1) | -- 3(10진수) 주소에 있는 Holding Register에 1 값 쓰기 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| dio.set_bits | | tool.get | | |
| dio.reset_bits | | tool.in_wait | | |
| dio.get | | tool.out_wait | | |
| dio.in_wait | | tool.voltage | | |
| dio.out_wait | | | | |
| dio.pulse | | | | |

## 2.5.18. tool.modbus_read_register : Holding Register에 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Holding Register 값 1개 읽기** |
| ■ 형 태 | Value = tool.modbus_read_register(Address) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsgiend short | - | 0 | 65,535 | |

■ **세부사항**

➢ **Address (Holding Register 주소)**

- 값을 읽어 올 Holding Register의 주소를 입력합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0~65,535 | unsgiend short | Holding Register의 값 |

■ 예 제

**[예제1] tool.modbus_read_register 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | val1 = | -- 0x1000(16진수) 주소에 있는 |
| 2 | tool.modbus_read_register(0x1000) | Holding Register 값 읽기 |
| 3 | | |
| 4 | val2 = tool.modbus_read_register(257) | -- 257(10진수) 주소에 있는 Holding |
| 5 | | Register 값 읽기 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | dio.set | aio.set | tool.set | tool.modbus_write_regiseter | |
| | dio.reset | aio.get | tool.reset | tool.modbus_read_regitster | |
| | dio.set_bits | | tool.get | | |
| | dio.reset_bits | | tool.in_wait | | |
| | dio.get | | tool.out_wait | | |
| | dio.in_wait | | tool.voltage | | |
| | dio.out_wait | | | | |
| | dio.pulse | | | | |

## 2.6. 다중처리(Thread) 스크립트

다중처리(Thread) 스크립트는 메인 스크립트와 병렬으로 동작하는 스크립트를 실행할 때 사용됩니다.

다중처리 스크립트 사용에 있어서 다음과 같은 제약 조건이 있습니다.

- 다중처리 스크립트는 최대 3개까지만 실행 가능합니다.
- 다중처리 스크립트 내에서는 동작 스크립트(movep, movel, trans, tool.transX, rot 등)의 명령어는 사용 불가능합니다.

**표 2-62 다중처리 스크립트 명령어 리스트**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 생성 | thread_name =thread(Script_name, Loop) | 스크립트를 실행할 Thread를 선언 | 2.6.1 |
| 2 | 실행 | thread_name.start() | 스레드 시작 | 2.6.2 |
| 3 | | thread_name.stop() | 스레드 정지 | 2.6.3 |
| 4 | | thread_name.puase() | 스레드 일시 정지 | 2.6.4 |
| 5 | | thread_name.resume() | 스레드 재시작 | 2.6.5 |
| 6 | 상태 | thread_name.state() | 스레드 동작 상태 가져오기 | 2.6.6 |
| 7 | 제어 | lock(Index) | 스레드 잠금 | 2.6.7 |
| 8 | | unlock(Index) | 스레드 잠금 해제 | 2.6.8 |

표 2-62 는 다중처리 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.6.1. thread_name = thread : 스레드 선언

■ 기 능

**스크립트를 실행할 Thread를 선언합니다.**

■ 형 태

thread_name =thread(Script_name, Loop)

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Script_name | string | - | 1 | 50 | byte |
| Loop | boolean | - | true, false | | |

* 단위 : 1byte = 알파벳 1글자 (최대 50 글자)

■ **세부사항**

➢ **Script_name (스크립트 이름)**

- 스레드에서 실행할 스크립트 이름을 작성합니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

➢ **loop (반복실행)**

- 스레드 반복 실행 여부를 설정합니다. False인 경우 한번만 실행합니다.

■ 예 제

**[예제1] thread_name = thread 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | th_1=thread(*"thread_1"*, true) | -- 반복 스레드 선언 (스크립트: thread_1) |
| 2 | th_1.start() | -- th_1 스레드 시작 |
| 3 | th_2=thread(*"thread_2"*, false) | -- 비반복 스레드 선언 (스크립트: thread_2) |
| 4 | th_2.start() | -- th_2 스레드 시작 |
| 5 | th_1.stop() | -- th_1 스레드 정지 |
| 6 | th_2.stop() | -- th_2 스레드 정지 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| | 스크립트 이름 | File doesn't exist.. |
| 실행오류 | 데이터 유효범위 | Value[script_name_length*] is range over(1~50) |
| | 스크립트 이름 | File doesn't exist.. |
| | 루아 스택 | [LThread] Failed getting script pointer. |
| | 스레드 생성 | Failed to create thread. |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.2. thread_name.start : 스레드 시작

| | |
|---|---|
| ■ 기 능 | 스레드를 시작합니다. |
| ■ 형 태 | thread_name.start() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ 세부사항

➢ 스레드 실행 명령어로 인자가 필요하지 않습니다.

■ 예 제

**[예제1] thread start 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | th_1=thread(*"thread_1"*, true) | -- 스레드 선언 (스크립트 : thread_1) |
| 2 | th_1.start() | -- th_1 스레드 시작 |
| 3 | th_1.stop() | -- th_1 스레드 정지 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 스레드 ID | Failed to start thread.[invalid thread id] |
| | 스레드 유효성 | Failed to start thread.[invalid thread] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.3. thread_name.stop : 스레드 정지

| | |
|---|---|
| ■ 기 능 | 스레드를 정지 시킵니다. |
| ■ 형 태 | thread_name.stop() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ 세부사항

➢ 스레드 실행 명령어로 인자가 필요하지 않습니다.

■ 예 제

**[예제1] thread stop 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | th_1=thread(*"thread_1"*, true) | -- 스레드 선언 (스크립트: thread_1) |
| 2 | th_1.start() | -- th_1 스레드 시작 |
| 3 | th_1.stop() | -- th_1 스레드 정지 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.4. thread_name.pause : 스레드 일시 정지

| | |
|---|---|
| ■ 기 능 | **스레드를 일시 정지 시킵니다.** |
| ■ 형 태 | thread_name.pause() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **스레드 실행 명령어로 인자가 필요하지 않습니다.**

■ 예 제

**[예제1] thread pause 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | th_1=thread(*"thread_1"*, true) | -- 스레드 선언 (스크립트: thread_1) |
| 2 | th_1.start() | -- th_1 스레드 시작 |
| 3 | th_1.pause() | -- th_1 스레드 일시 정지 |
| 4 | th_1.resume() | -- th_1 스레드 다시 시작 |
| 5 | th_1.stop() | -- th_1 스레드 정지 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.5. thread_name.resume : 스레드 재시작

| | |
|---|---|
| ■ 기 능 | **스레드를 재시작 합니다.** |
| ■ 형 태 | thread_name.resume() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **스레드 실행 명령어로 인자가 필요하지 않습니다.**

■ 예 제

**[예제1] thread resume 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | th_1=thread(*"thread_1"*, true) | -- 스레드 선언 (스크립트: thread_1) |
| 2 | th_1.start() | -- th_1 스레드 시작 |
| 3 | th_1.pause() | -- th_1 스레드 일시 정지 |
| 4 | th_1.resume() | -- th_1 스레드 다시 시작 |
| 5 | th_1.stop() | -- th_1 스레드 정지 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.6. thread_name.state: 스레드 상태 확인

| | |
|---|---|
| ■ 기 능 | **스레드를 상태를 확인합니다.** |
| ■ 형 태 | thread_name.state() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **스레드 실행 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | int | 정지 |
| 1 | Int | 동작중 |
| 2 | int | 일시 정지 |
| -1 | int | 스레드가 유효하지 않음 |

➢ **Return (반환 값)**

- 생성된 스레드의 상태값을 반환합니다.

■ 예 제

**[예제1] thread state 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | th_1=thread(*"thread_1"*, true) | -- 스레드 선언 (스크립트: thread_1) |
| 2 | th_1.start() | -- th_1 스레드 시작 |
| 3 | th_1.pause() | -- th_1 스레드 일시 정지 |
| 4 | th_1.resume() | -- th_1 스레드 다시 시작 |
| 5 | th_1.stop() | -- th_1 스레드 정지 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.7. lock: 스레드 잠금

| | |
|---|---|
| ■ 기 능 | **동일한 자원에 대한 동시 접근을 방지하기 위해 스레드 잠금을 실행합니다.** |
| ■ 형 태 | lock(Index) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 4 | - |

■ **세부사항**

➢ **Index (mutex 번호)**

- lock을 실행 할 mutex 번호를 설정합니다.
- 여러 스레드 내에서 정수형 데이터와 같은 공용으로 사용되는 변수의 다중 접근, 방지, 동기화를 위해 사용됩니다.
- 스레드에서 동일한 mutex 번호로 잠금을 실행한 경우, lock 호출에서 멈춥니다. 다른 경우는 잠금을 실행 후, 계속해서 스레드를 실행합니다.
- 스레드 잠금이 더 이상 필요 없는 경우 반드시 unlock을 호출하여야 합니다.

ⓘ **해당 명령어는 메인 스크립트가 아닌 스레드 동작 내부 스크립트에서 사용되는 명령어입니다.**

■ 예 제

**[예제1] thread lock 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | a=10 | -- 변수 a에 10 저장 |
| 2 | lock(1) | -- mutex 1번 접근 금지 설정 |
| 3 | var.i(1,a) | -- 정수형 전역변수 1번에 10을 저장 |
| 4 | unlock(1) | -- mutex 1번 접근 금지 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[5] is range over(1~4) |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.6.8. unlock: 스레드 잠금 해제

| | |
|---|---|
| ■ 기 능 | **스레드 잠금을 해제합니다.** |
| ■ 형 태 | unlock(Index) |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 4 | - |

■ **세부사항**

➢ **Index (mutex 번호)**

- unlock을 실행 할 mutex 번호를 설정합니다.
- lock을 한 후 반드시 unlock을 수행하여야만 스레드에서 해당 mutex 번호로 스레드 잠금 해제를 실행 할 수 있습니다.

ⓘ **해당 명령어는 메인 스크립트가 아닌 스레드 동작 내부 스크립트에서 사용되는 명령어입니다.**

■ 예 제

**[예제1] thread unlock사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | a=10 | -- 변수 a에 10 저장 |
| 2 | lock(1) | -- mutex 1번 접근 금지 설정 |
| 3 | var.i(1,a) | -- 정수형 전역변수 1번에 10을 저장 |
| 4 | unlock(1) | -- mutex 1번 접근 금지 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[5] is range over(1~4) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| th=thread | | | | |
| th.start | | | | |
| th.stop | | | | |
| th.pause | | | | |
| th.resume | | | | |
| th.state | | | | |
| lock | | | | |
| unlock | | | | |

## 2.7. 사용자 메시지 출력 스크립트

메시지 출력 스크립트는 화면에 설정한 사용자 메시지창을 출력하는 기능입니다. 메시지 종류에 따라 에러, 경고, 정보로 구분하여 사용할 수 있습니다.

**표 2-63 메시지 출력 스크립트 명령어 리스트**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 에러 | msg.error(Message) | 에러 메시지창 출력 | 2.7.1 |
| 2 | 경고 | msg.warn(Message) | 경고 메시지창 출력 | 2.7.2 |
| 3 | 정보 | msg.info(Message) | 정보 메시지창 출력 | 2.7.3 |

표 2-63 은 메시지 출력 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.7.2. msg.error : 에러 메시지

| | |
|---|---|
| ■ 기 능 | **에러 메시지창에 사용자 메시지를 출력합니다.** |
| ■ 형 태 | msg.error(Message) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Message | string | - | 1 | 256 | - |

■ **세부사항**

➢ **Message (메시지 출력 문자)**

- 출력 문자는 사용자 변수 또는 직접 메시지을 작성할 수 있습니다.
- 직접 메시지 작성 시 **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.
- 에러 알림 기능으로 사용하시기 바랍니다.

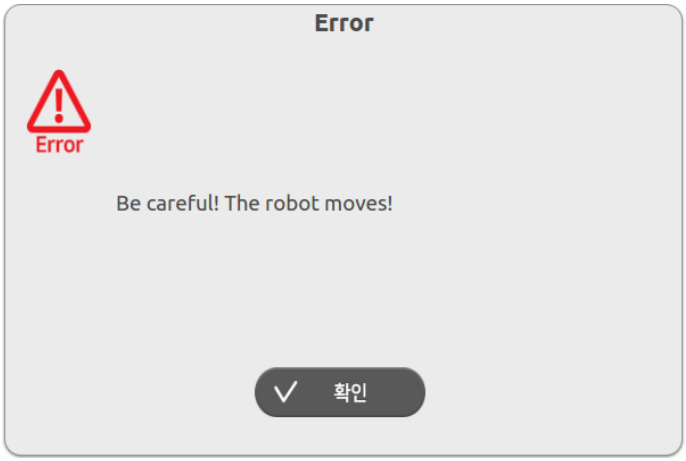



**그림 2-69 Error message 팝업 창**

- 사용자 에러 메시지를 그림 2-69과 같이 화면에 출력합니다.

■ 예 제

**[예제1] msg.error 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | msg.error(*"Oh! The robot moves"*) | -- 직접 메시지 출력 |
| 2 | | |
| 3 | str1 = *"Be careful! The robot moves"* | -- 사용자 변수 정의 |
| 4 | msg.error(str1) | -- 에러 메시지 출력 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[257] is range over(1~256) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| msg.error | | | | |
| msg.warn | | | | |
| msg.info | | | | |

## 2.7.3. msg.warn : 경고 메시지

| | |
|---|---|
| ■ 기 능 | **경고 메시지창에 사용자 메시지를 출력합니다.** |
| ■ 형 태 | msg.warn(Message) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Message | string | - | 1 | 256 | - |

■ **세부사항**

➢ **Message (메시지 출력 문자)**

- 출력 문자는 사용자 변수 또는 직접 메시지을 작성할 수 있습니다.
- 직접 메시지 작성 시 **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.
- 경고 알림 기능으로 사용하시기 바랍니다.

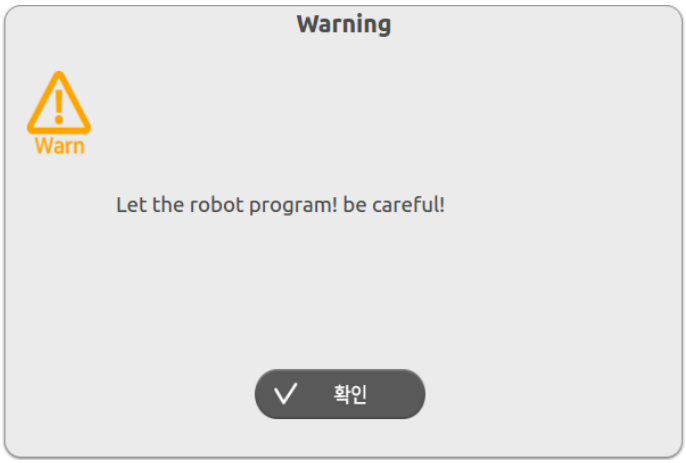



**그림 2-70 Warning message 팝업 창**

- 사용자 에러 메시지를 그림 2-70과 같이 화면에 출력합니다.

■ 예 제

**[예제1] msg.warn 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | msg.warn(*"Let the robot program! be* | -- 직접 메시지 출력 |
| 2 | *careful!"*) | |
| 3 | | |
| 4 | str1 = *"Let the robot program! be* | -- 사용자 변수 정의 |
| 5 | *careful!"* | |
| 6 | msg.warn(str1) | -- 경고 메시지 출력 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[257] is range over(1~256) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| msg.error | | | | |
| msg.warn | | | | |
| msg.info | | | | |

## 2.7.4. msg.info : 정보 메시지

| | |
|---|---|
| ■ 기 능 | **정보 메시지창에 사용자 메시지를 출력합니다.** |
| ■ 형 태 | msg.info(Message) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Message | string | - | 1 | 256 | - |

■ **세부사항**

➢ **Message (메시지 출력 문자)**

- 출력 문자는 사용자 변수 또는 직접 메시지을 작성할 수 있습니다.
- 직접 메시지 작성 시 **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.
- 정보 알림 기능으로 사용하시기 바랍니다.

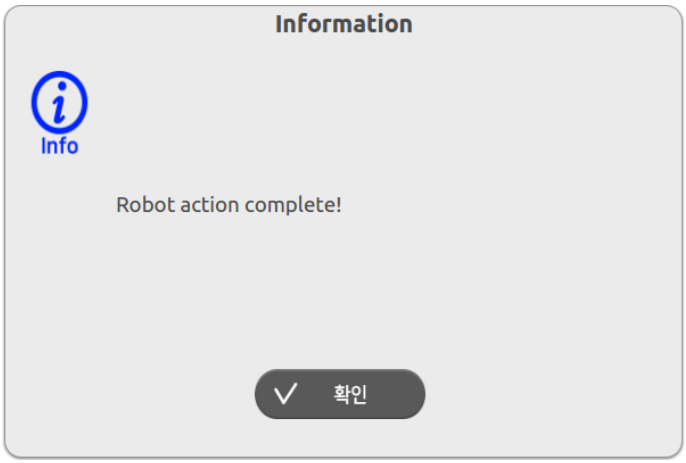



**그림 2-71 Information message**

- 사용자 정보 메시지를 그림 2-71과 같이 화면에 출력합니다.

■ 예 제

**[예제1] msg.info 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | msg.info(*"Robot action complete!"*) | -- 직접 메시지 출력 |
| 2 | | |
| 3 | str1 = "Robot action complete!" | -- 사용자 변수 정의 |
| 4 | msg.info(str1) | -- 정보 메시지 출력 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[257] is range over(1~256) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| msg.error | | | | |
| msg.warn | | | | |
| msg.info | | | | |

7

## 2.8. 수학 함수 스크립트

Lua 언어의 math 라이브러리의 수학함수를 사용할 수 있는 기능을 제공합니다. 삼각함수(sin, cos, tan, ...), 제곱, 제곱근 등 표준 수학함수들로 구성되어 있습니다. 수학함수 스크립트 명령어를 통해 데이터를 환산 및 관리할 수 있습니다.

**표 2-64 수학함수 스크립트 항목**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 기본함수 | math.pi | pi 값 반환 (3.14159265) | 2.8.1 |
| 2 | | math.deg(X) | X radian의 degree 값 반환 | 2.8.2 |
| 3 | | math.rad(X) | X degree의 radian 값 반환 | 2.8.3 |
| 4 | | math.sqrt(X) | X의 제곱근 반환 | 2.8.4 |
| 5 | | math.pow(X, Y) | X의 거듭제곱 반환 | 2.8.5 |
| 6 | | math.abs(X) | X의 절대값 반환 | 2.8.6 |
| 7 | | math.ceil(X) | X의 소수점 자리의 올림 값 반환 | 2.8.7 |
| 8 | | math.floor(X) | X의 소수점 자리의 내림 값 반환 | 2.8.8 |
| 9 | 삼각함수 | math.sin(X) | X의 sine 값 반환 | 2.8.9 |
| 10 | | math.cos(X) | X의 cosine 값 반환 | 2.8.10 |
| 11 | | math.tan(X) | X의 tangent 값 반환 | 2.8.11 |
| 12 | | math.asin(X) | X의 arc sine 값 반환 | 2.8.12 |
| 13 | | math.acos(X) | X의 arc cosine 값 반환 | 2.8.13 |
| 14 | | math.atan(X) | X의 arc tangent 값 반환 | 2.8.14 |
| 15 | | math.atan2(Y, X) | Y/X의 arc tangent 값 반환 | 2.8.15 |

표 2-64 는 수학 함수 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.8.1. math.pi : 파이

| | |
|---|---|
| ■ 기 능 | **pi (3.141582...)를 반환합니다.** |
| ■ 형 태 | math.pi |

■ 인 자

| 인자명 | 자료형 | 기본값 | 범위 | 단위 |
|---|---|---|---|---|
| - | - | - | - | - |

■ **세부사항**

- pi 명령어는 인자가 입력되지 않습니다.
- $\pi$ (3,141593)을 반환합니다.
- 사칙연산을 할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 3.141593 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.pi 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | r=math.pi | -- pi 변수 정의 |
| 2 | | |
| 3 | s1 = math.sin(r) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 0 |
| 5 | s2 = math.sin(math.pi) | -- 입력 인자 pi 사용 |
| 6 | | ※ 결과 : 0 |
| 7 | r1 = r + 3.14 | -- 사칙연산 |
| 8 | | ※ 결과 : 6.281593 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | attempt to call a number value (field 'pi') |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt | math.sin | math.asin | |
| math.deg | math.pow | math.cos | math.acos() | |
| math.rad | math.abs | math.tan | math.atan() | |
| | math.ceil | | math.atan2() | |
| | math.floor | | | |

## 2.8.2. math.deg : Degree 각도

| | |
|---|---|
| ■ 기 능 | **radian(호도법)을 degree(육십분법) 값으로 반환합니다.** |
| ■ 형 태 | math.deg(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | rad |

* 단위 : rad (radian)

■ **세부사항**

➢ **X (거듭 제곱 입력 인자)**

- x(호도법)을 degree(육십분법)값으로 반환합니다.
- 사칙연산이 가능합니다.

➢ **deg 함수**

- y = math.deg(x) 함수는 $y = x \times (\frac{180^{\circ}}{\pi})$ 식과 같은 변환 공식을 가집니다.
- math.deg(x)의 결과 값을 y의 변수로 반환을 하는 경우,

  y = math.deg(0) // y = 0.0

  y = math.deg(0.523599) // y = 30.0

  y = math.deg(0.785247) // y = 44.991348

  y = math.deg(1.570796) // y = 90.0 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | deg |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.deg 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | x=0.5 | -- x 변수 정의 |
| 2 | | |
| 3 | s1=math.asin(x) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 0.534599 |
| 5 | d1=math.deg(s1) | ※ 결과 : 30 |
| 6 | | |
| 7 | s2=math.asin(0.5) | -- 입력 인자 0.5 |
| 8 | | ※ 결과 : 0.534599 |
| 9 | d2=math.deg(s2) | ※ 결과 : 30 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(0~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.3. math.rad : Radian 각도

| | |
|---|---|
| ■ 기 능 | **degree(육십분법)을 radian(호도법) 값으로 반환합니다.** |
| ■ 형 태 | math.rad(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | deg |

* 단위 : deg (degree)

■ **세부사항**

➢ **X (거듭 제곱 입력 인자)**

- x(육십분법)을 radian(호도법)값으로 반환합니다.
- 사칙연산이 가능합니다.

➢ **rad 함수**

- y = math.rad(x) 함수는 $y = x \times (\frac{\pi}{180^{\circ}})$ 식과 같은 변환 공식을 가집니다.
- math.rad(x)의 결과 값을 y의 변수로 반환을 하는 경우,

  y = math.rad(0) // y = 0.0

  y = math.rad(45) // y = 0.785398

  y = math.rad(90) // y = 1.570796

  y = math.rad(180) // y = 3.141593 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | rad |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.rad 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | d=math.rad(90) | -- radian 변수 정의 |
| 2 | | |
| 3 | s1 = math.sin(d) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 1 |
| 5 | s2 = math.sin(math.rad(90)) | -- 입력 인자 rad(90) 사용 |
| 6 | | ※ 결과 : 1 |
| 7 | s3 = math.sin(math.pi/2) | -- 입력 인자 pi 함수 사용 |
| 8 | | ※ 결과 : 1 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(0~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.4. math.sqrt : 제곱근

| | |
|---|---|
| ■ 기 능 | **제곱근(Square Root) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.sqrt(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | -100000 | 100000 | - |

■ **세부사항**

➢ **X (제곱근 식 입력)**

- 입력 인자의 제곱근을 반환합니다.
- 입력 값이 음수인 경우, -nan이 반환됩니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **sqrt 함수**

- y = math.sqrt(x) 함수는 $y = \sqrt{x}$ 식과 같은 변환 공식을 가집니다.
- math.sqrt(x)의 결과 값을 y의 변수로 반환을 하는 경우,
  y = math.sqrt(1) // y = 1.0
  y = math.sqrt(9) // y = 3.0
  y = math.sqrt(30) // y = 5.477226
  y = math.sqrt(90) // y = 9.486833 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.sqrt 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | i=9 | |
| 2 | | |
| 3 | i1 = math.sqrt(9) | -- sqrt 함수에 직접 입력 |
| 4 | | ※ 결과 : 3.0 |
| 5 | i2 = math.sqrt(i) | -- sqrt 함수에 변수 입력 |
| 6 | | ※ 결과 : 3.0 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[2001] is range over(0~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.5. math.pow : 거듭제곱

| | |
|---|---|
| ■ 기 능 | **거듭제곱(Power) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.pow(X, N) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |
| N | double | - | - | - | - |

■ **세부사항**

➢ **X, N (거듭 제곱 입력 인자)**

- x(밑수)와 n(지수)의 거듭제곱을 반환합니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **pow함수**

- y = math.pow(x, n) 함수는 $y = x^n$ 식과 같은 변환 공식을 가집니다.
- math.pow(x, n)의 결과 값을 y의 변수로 반환을 하는 경우,

  y = math.pow(1, 2) // y = 1.0

  y = math.pow (-3, 5) // y = -243.0

  y = math.pow (5.6, 3) // y = 175.616

  y = math.pow (11, 2) // y = 121 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.pow 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | i=9 | |
| 2 | | |
| 3 | i1 = math.pow(9,2) | -- pow 함수에 직접 입력 |
| 4 | | ※ 결과 : 81.0 |
| 5 | i2 = math.pow(i,2) | -- pow 함수에 변수 입력 |
| 6 | | ※ 결과 : 81.0 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[2001] is range over(0~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.6. math.abs : 절대 값

| | |
|---|---|
| ■ 기 능 | **절대 값(Absolute) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.abs(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |

■ **세부사항**

➢ **X (절대 값 입력 인자)**

- 입력 인자의 절대 값을 반환합니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **abs함수**

- y = math.abs(x) 함수는 $y = |x|$ 식과 같은 변환 공식을 가집니다.
- math.abs(x)의 결과 값을 y의 변수로 반환을 하는 경우,
  y = math.abs(1) // y = 1
  y = math.abs (-3) // y = 3
  y = math.abs (-5.63) // y = 5.63 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.pow 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | i = -78.09 | |
| 2 | | |
| 3 | i1 = math.abs(-7) | -- abs 함수에 직접 입력 |
| 4 | | ※ 결과 : 7 |
| 5 | i2 = math.abs(i) | -- abs 함수에 변수 입력 |
| 6 | | ※ 결과 : 78.09 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[2001] is range over(0~2000) |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.7. math.ceil : 올림

| | |
|---|---|
| ■ 기 능 | **소수부를 올림하여(Ceil) 정수 값을 반환합니다.** |
| ■ 형 태 | math.ceil(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |

■ **세부사항**

➢ **X (실수형 입력 인자)**

- 1. 소수부를 올림하여 정수값을 반환할 입력 값입니다.
- 2. 입력 인자는 소수부를 올림하여 정수 값을 반환합니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **ceil 함수**

- math.ceil(x)의 결과 값을 y의 변수로 반환을 하는 경우,
  y = math.ceil(1.01) // y = 2
  y = math.ceil(-3.394) // y = -3
  y = math.ceil(9.93) // y = 10 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.ceil 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | i = -78.09 | |
| 2 | | |
| 3 | i1 = math.ceil(7.321) | -- ceil 함수에 직접 입력 |
| 4 | | ※ 결과 : 7 |
| 5 | i2 = math.ceil(i) | -- ceil 함수에 변수 입력 |
| 6 | | ※ 결과 : -78 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.8. math.floor : 내림

| | |
|---|---|
| ■ 기 능 | **소수부를 내림하여(Floor) 정수 값을 반환합니다.** |
| ■ 형 태 | math.floor(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |

■ **세부사항**

➢ **X (실수형 입력 인자)**

- 1. 소수부를 내림하여 정수값을 반환할 입력 값입니다.
- 2. 입력 인자는 소수부를 내림하여 정수 값을 반환합니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **floor 함수**

- math.floor(x)의 결과 값을 y의 변수로 반환을 하는 경우,
  y = math.floor(1.01) // y = 1
  y = math.floor(-3.394) // y = -4
  y = math.floor(9.93) // y = 9 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.floor 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | i = -78.09 | |
| 2 | | |
| 3 | f1 = math.floor(7.321) | -- floor 함수에 직접 입력 |
| 4 | | ※ 결과 : 7 |
| 5 | f2 = math.floor(-78.09) | -- floor 함수에 변수 입력 |
| 6 | | ※ 결과 : -79 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.9. math.sin : 사인함수

| | |
|---|---|
| ■ 기 능 | **사인(Sine) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.sin(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | rad |

* 단위 : rad (radian)

■ **세부사항**

➢ **X (입력 radian 값)**

- 사인 함수의 반환 값의 범위는 $-1 \le \sin(x) \le 1$ 입니다.
- 인자의 단위는 degree 입니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **sin 함수**

- y = math.sin(x) 함수는 $y = \sin(x)$ 식과 같은 변환 공식을 가집니다.
- sin 함수의 입력 값의 단위는 radian 입니다.
- math.sin(x)의 결과 값을 y의 변수로 반환을 하는 경우,

  y = math.sin(math.rad(0)) // y = 0.0

  y = math.sin(math.rad(30)) // y = 0.5

  y = math.sin(math.rad(45)) // y = 0.7071068

  y = math.sin(math.rad(60)) // y = 0.8660254

  y = math.sin(math.rad(90)) // y = 1.0 의 결과를 확인할 수 있습니다.

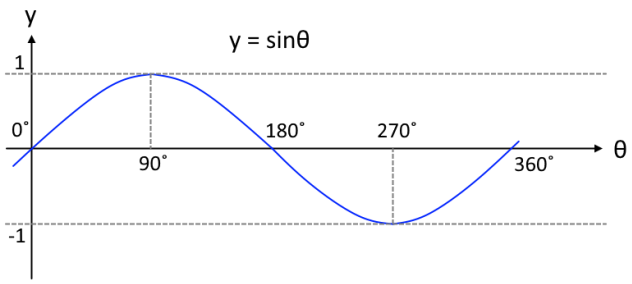



**그림 2-72 sin함수**

■ **반환 값**

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.sin 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | d=math.rad(90) | -- radian 변수 정의 |
| 2 | | |
| 3 | s1 = math.sin(d) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 1 |
| 5 | s2 = math.sin(math.rad(90)) | -- 입력 인자 rad(90) 사용 |
| 6 | | ※ 결과 : 1 |
| 7 | s3 = math.sin(math.pi/2) | -- 입력 인자 pi 함수 사용 |
| 8 | | ※ 결과 : 1 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.10. math.cos : 코사인 함수

| | |
|---|---|
| ■ 기 능 | **코사인(Cosine) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.cos(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |

* 단위 : rad (radian)

■ **세부사항**

➢ **X (입력 radian 값)**

- 코사인 함수의 반환 값의 범위는 $-1 \le \cos(x) \le 1$ 입니다.
- 인자의 단위는 degree 입니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **tan 함수**

- y = math.cos(x) 함수는 $y = \cos(x)$ 식과 같은 변환 공식을 가집니다.
- cos 함수의 입력 값의 단위는 radian 입니다.
- math.cos(x)의 결과 값을 y의 변수로 반환을 하는 경우

  y = math.cos(math.rad(0)) // y = 1.0

  y = math.cos(math.rad(30)) // y = 0.8660254

  y = math.cos(math.rad(45)) // y = 0.7071068

  y = math.cos(math.rad(60)) // y = 0.5

  y = math.cos(math.rad(90)) // y = 0.0 의 결과를 확인할 수 있습니다.

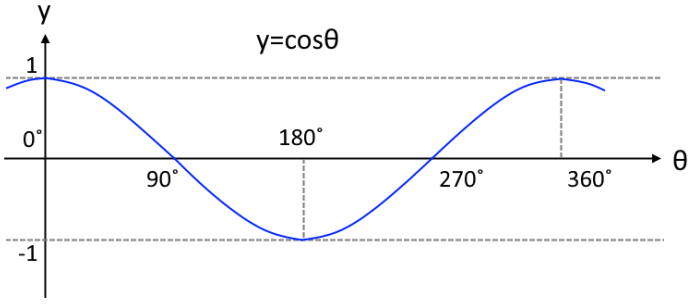



**그림 2-73 cos함수**

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.cos 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | d=math.rad(90) | -- radian 변수 정의 |
| 2 | | |
| 3 | c1=math.cos(d) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 0 |
| 5 | c2=math.cos(math.rad(90)) | -- 입력 인자 rad(90) 사용 |
| 6 | | ※ 결과 : 0 |
| 7 | c3=math.cos(math.pi/2) | -- 입력 인자 pi 함수 사용 |
| 8 | | ※ 결과 : 0 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.11. math.tan : 탄젠트 함수

| | |
|---|---|
| ■ 기 능 | **탄젠트(Tangent) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.tan(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |

* 단위 : rad (radian)

■ **세부사항**

➢ **X (입력 radian 값)**

- 인자의 단위는 degree 입니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **tan 함수**

- y = math.tan(x) 함수는 $y = \tan(x)$ 식과 같은 변환 공식을 가집니다.
- tan 함수의 입력 값의 단위는 radian 입니다.
- math.tan(x)의 결과 값을 y의 변수로 반환을 하는 경우

  y = math.tan(math.rad(0)) // y = 0.0

  y = math.tan(math.rad(30)) // y = 0.5773503

  y = math.tan(math.rad(45)) // y = 1.0

  y = math.tan(math.rad(60)) // y = 1.7320508

  y = math.tan(math.rad(90)) // y = 1.6331239 의 결과를 확인할 수 있습니다.

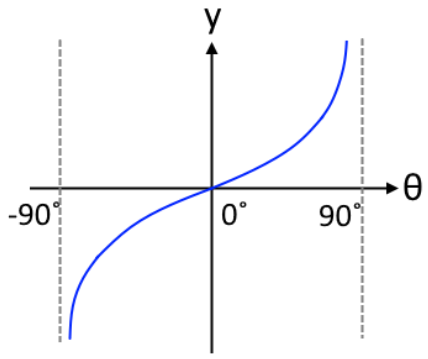



**그림 2-74 tan 함수**

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.tan 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | d=math.rad(45) | -- radian 변수 정의 |
| 2 | | |
| 3 | t1=math.tan(d) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 1 |
| 5 | t2=math.tan(math.rad(45)) | -- 입력 인자 rad(45) 사용 |
| 6 | | ※ 결과 : 1 |
| 7 | t3=math.tan(math.pi/4) | -- 입력 인자 pi 함수 사용 |
| 8 | | ※ 결과 : 1 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.12. math.asin : 아크사인 함수

| | |
|---|---|
| ■ 기 능 | **아크사인(Arcsine) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.asin(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | -1 | 1 | - |

■ **세부사항**

➢ **X (입력 값)**

- 아크사인 함수의 반환 값의 범위는 $-\frac{\pi}{2} \le \mathrm{asin}(x) \le \frac{\pi}{2}$ 입니다.
- 입력인자의 범위를 벗어나는 경우 nan 값을 반환합니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **asin 함수**

- y = math.asin(x) 함수는 sin 함수의 역함수이며, $y = a\sin(x)$ 식과 같은 변환 공식을 가집니다.
- math.asin(x)의 결과 값을 y의 변수로 반환을 하는 경우,
  y = math.asin(0) // y = 0.0 (deg : 0)
  y = math.asin(0.5) // y = 0.523599 (deg : 30)
  y = math.asin(0.707) // y = 0.785247 (deg : 44.991348)
  y = math.asin(1) // y = 1.570796 (deg : 90) 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.asin 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | x=0.5 | -- x 변수 정의 |
| 2 | | |
| 3 | s1=math.asin(x) | -- 입력 인자 사용자 변수 사용 |
| 4 | | ※ 결과 : 0.534599 |
| 5 | d1=math.deg(s1) | ※ 결과 : 30 |
| 6 | | |
| 7 | s2=math.asin(0.5) | -- 입력 인자 0.5 |
| 8 | | ※ 결과 : 0.534599 |
| 9 | d2=math.deg(s2) | ※ 결과 : 30 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.13. math.acos : 아크코사인 함수

| | |
|---|---|
| ■ 기 능 | **아크코사인(Arccosine) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.acos(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | -1 | 1 | - |

■ **세부사항**

➢ **X (입력 값)**

- 아크코사인 함수의 반환 값의 범위는 $0 \le \mathrm{acos}(x) \le \pi$ 입니다.
- 입력인자의 범위를 벗어나는 경우 nan 값을 반환합니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **asin 함수**

- y = math.acos(x) 함수는 cos 함수의 역함수이며, $y = \mathrm{acos}(x)$ 식과 같은 변환 공식을 가집니다.
- math.acos(x)의 결과 값을 y의 변수로 반환을 하는 경우,
  y = math.acos(0) // y = 1. 570796 (deg : 90)
  y = math.acos(0.5) // y = 1.047198 (deg : 60)
  y = math.acos(0.707) // y = 0.785247 (deg : 45.008652)
  y = math.acos(1) // y = 0 (deg : 0) 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.asin 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | x=0.5 | -- x 변수 정의 |
| 2 | | |
| 3 | s1=math.acos(x) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 1.047198 |
| 5 | d1=math.deg(s1) | ※ 결과 : 60 |
| 6 | | |
| 7 | s2=math.acos(0.5) | -- 입력 인자 0.5 |
| 8 | | ※ 결과 : 1.047198 |
| 9 | d2=math.deg(s2) | ※ 결과 : 60 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.14. math.atan : 아크탄젠트 함수

| | |
|---|---|
| ■ 기 능 | **아크탄젠트(Arctangent) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.atan(X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |

■ **세부사항**

➢ **X (입력 값)**

- 아크탄젠트 함수의 반환 값의 범위는 $-\frac{\pi}{2} \leq \mathrm{atan}(x) \leq \frac{\pi}{2}$ 입니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **atan 함수**

- y = math.atan(x) 함수는 $y = \alpha\tan(x)$ 식과 같은 변환 공식을 가집니다.
- math.atan(x)의 결과 값을 y의 변수로 반환을 하는 경우
  y = math.atan(0) // y = 0.0
  y = math.atan(0.5773503) // y = 0.524637 (deg : 30.059470)
  y = math.atan(1) // y = 1.021365 (deg : 45.0) 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.tan 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | r=math.rad(1) | -- radian 변수 정의 |
| 2 | | |
| 3 | t1=math.tan(r) | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 1.021365 |
| 5 | d1=math.deg(t1) | ※ 결과 : 1 |
| 6 | | |
| 7 | t2=math.tan(1) | -- 입력 인자 1 |
| 8 | | ※ 결과 : 1.021365 |
| 9 | d2=math.deg(t2) | ※ 결과 : 1 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.8.15. math.atan2 : 제2아크탄젠트 함수

| | |
|---|---|
| ■ 기 능 | **제2아크탄젠트(Arctangent2) 함수의 결과 값을 반환합니다.** |
| ■ 형 태 | math.atan2(Y, X) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| X | double | - | - | - | - |
| Y | double | - | - | - | - |

■ **세부사항**

➢ **X, Y (직각 삼각형의 너비 및 높이 값)**

- 제2아크탄젠트 함수의 반환 값의 범위는 $-\pi \leq \mathrm{atan2}(Y,\ X) \leq \pi$ 입니다.
- 사용자 변수로 정의할 수 있으며, 사칙연산이 가능합니다.

➢ **atan 함수**

- a = math.atan2(Y, X) 함수는 $a = atan2(Y, X)$ 식과 같은 변환 공식을 가집니다.
- math.atan2(Y, X)의 결과 값을 a의 변수로 반환을 하는 경우

  a = math.atan2(1.0, 1.0) // Y = 0.7854

  a = math.atan2(2.0, 1.0) // Y = 1.1071

  a = math.atan2(1.0, 2.0) // Y = 0.4636 의 결과를 확인할 수 있습니다.

■ 반환 값

| 구분 | 반환 값 | 설명 |
|---|---|---|
| 성공 | double | 입력 인자의 반환 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] math.tan 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | r=math.rad(1) | -- radian 변수 정의 |
| 2 | | |
| 3 | t1=math.tan(r)7 | -- 입력 인자 변수 사용 |
| 4 | | ※ 결과 : 1.021365 |
| 5 | d1=math.deg(t1) | ※ 결과 : 1 |
| 6 | | |
| 7 | t2=math.tan(1) | -- 입력 인자 1 |
| 8 | | ※ 결과 : 1.021365 |
| 9 | d2=math.deg(t2) | ※ 결과 : 1 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| math.pi | math.sqrt() | math.sin() | math.asin() | |
| math.deg() | math.pow() | math.cos() | math.acos() | |
| math.rad() | math.abs() | math.tan() | math.atan() | |
| | math.ceil() | | math.atan2() | |
| | math.floor() | | | |

## 2.9. TCP/IP 통신 서버 스크립트

제어기에 TCP/IP 서버(Server)를 구성하여 이더넷(Ethernet) 기반의 외부기기와 통신을 수행할 수 있도록 지원하는 명령어입니다.

**표 2-65 TCP/IP 통신 서버 스크립트 항목**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 연결 / 해제 | serv = server(Port) | 서버 선언 | 2.9.1 |
| 2 | | serv.open() | 서버 열기 | 2.9.2 |
| 3 | | serv.close() | 서버 닫기 | 2.9.3 |
| 4 | 통신 상태 | serv.state() | 서버 상태 확인 | 2.9.4 |
| 5 | 버퍼 초기화 | serv.flush() | 송/수신 데이터 버퍼 초기화 | 2.9.5 |
| 6 | 데이터 수신 | serv.read(Length, Timeout) | 데이터 읽기 | 2.9.6 |
| 7 | 데이터 송신 | serv.write(Data, Length) | 데이터 쓰기 | 2.9.7 |
| 8 | | serv.set_label(Data) | 송신 데이터의 접미사 지정 | 2.9.8 |

표 2-65 는 TCP/IP 통신 서버 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

**표 2-66 TCP/IP 통신 서버 오류 반환값**

| 반환 값 | 항 목 | 설 명 |
|---|---|---|
| -1 | ERROR_CLIENT_NONE | 오류 코드 초기값 |
| 0 | SUCCESS_ SERVER | 서버 관련 명령어 성공 |
| -201 | ERROR_SERVER_FAIL | 서버의 소켓 파일 디스크립터가 유효하지 않은 경우 |
| -202 | ERROR_ SERVER _BAD_ARGUMENT | 서버의 데이터 인자 타입이 맞이 않은 상태 |
| -203 | ERROR_ SERVER _OPENED | 서버가 연결된 상태 |
| -204 | ERROR_ SERVER _OPENED_FAIL | 서버 연결 실패 |
| -205 | ERROR_SERVER_CLOSED | 서버가 이미 닫혀 있는 상태 |
| -206 | ERROR_ SERVER _WRITE_FAIL | 데이터 송신 실패 |
| -207 | ERROR_ SERVER _READ_FAIL | 데이터 수신 실패 |
| -208 | ERROR_ SERVER _MEM_ALLOC | 수신 데이터 메모리 할당 실패 |
| -209 | ERROR_ SERVER _TIMEOUT | 수신 데이터 타임아웃(Timeout) |
| -210 | ERROR_ SERVER _DISCONNECTED | 서버가 연결되지 않은 상태 |

표 2-66 은 TCP/IP 통신 서버 오류 반환값에 대해 간략히 설명한 표이며, 오류 발생에 대한 조치사항은 알람 매뉴얼을 참조하시기 바랍니다.

## 2.9.1. serv = server : 서버 선언

| | |
|---|---|
| ■ 기 능 | **TCP/IP 서버(Server)를 설정한 Port 번호로 선언합니다.** |
| ■ 형 태 | serv = server(Port) |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | int | - | 0 | 65535 | - |

■ **세부사항**

➢ **Port (포트 번호)**

- 서버 소켓의 경우 등록된 포트의 번호인 0~65535번 영역을 사용하시기 바랍니다.
- 포트 번호가 중첩되지 않도록 확인 후 사용하시기 바랍니다.

■ 예 제

**[예제1] serv = server 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | state=serv.state() | -- 클라이언트 접속 유/무 판단 |
| 4 | serv.close() | -- 서버 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[Port*] is range over(0~65535) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| serv=server() | | | | |
| serv.open() | | | | |
| serv.close() | | | | |
| serv.state() | | | | |
| serv.flush() | | | | |
| serv.read() | | | | |
| serv.write() | | | | |
| serv.set_label() | | | | |

## 2.9.2. serv.open : 서버 열기

| | |
|---|---|
| ■ 기 능 | **클라이언트(Client) 연결이 가능하도록 TCP/IP 서버(Server)를 열어줍니다.** |
| ■ 형 태 | serv.open() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **서버 열기 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERVER | 소켓 생성 성공 |
| -201 | ERROR_SERVER_FAILED | 소켓 파일 디스크립터가 유효하지 않음 |
| -202 | ERROR_SERVER_BAD_ARGUMENT | 서버의 데이터 인자 타입 에러 |
| -203 | ERROR_SERVER_OPENED | 서버가 열려있는 상태 |
| -204 | ERROR_SERVER_OPEN_FAIL | 서버 열기 실패 |

※ 로봇 제어기가 서버(Server) 소켓을 생성하여, 클라이언트(Client) 연결 시, 연결된 소켓을 반환합니다.

■ 예 제

**[예제1] serv.open 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | state=serv.state() | -- 클라이언트 접속 유/무 판단 |
| 4 | serv.close() | -- 서버 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | unexpected symbol near ']' |

| | |
|---|---|
| ■ 관련 함수 | serv=server()<br>serv.open()<br>serv.close()<br>serv.state()<br>serv.flush()<br>serv.read()<br>serv.write()<br>serv.set_label() |

## 2.9.3. serv.close : 서버 종료

| | |
|---|---|
| ■ 기 능 | **서버(Server)와 클라이언트(Client)간 통신을 종료합니다.** |
| ■ 형 태 | serv.close() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **서버 통신 종료 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERVER | 서버 종료 성공 |
| -205 | ERROR_SERVER_CLOSED | 서버가 이미 닫혀있는 상태 |

■ 예 제

**[예제1] serv.close 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | state=serv.state() | -- 클라이언트 접속 유/무 판단 |
| 4 | serv.close() | -- 서버 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | unexpected symbol near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| serv=server() | | | | |
| serv.open() | | | | |
| serv.close() | | | | |
| serv.state() | | | | |
| serv.flush() | | | | |
| serv.read() | | | | |
| serv.write() | | | | |
| serv.set_label() | | | | |

## 2.9.4. serv.state : 클라이언트 접속 상태 확인

| | |
|---|---|
| ■ 기 능 | **클라이언트(Client)의 접속 유/무 상태를 반환합니다.** |
| ■ 형 태 | serv.state() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **서버 상태 확인 명령어로 인자가 필요하지 않습니다.**

- 운용 중인 소켓 서버가 닫힌 경우 serv.state의 반환 값이 -205이며, 이와 같은 경우 serv.open() 스크립트를 통해 소켓을 재 생성해 주시기 바랍니다.
- 클라이언트(Client)가 서버에 정상적으로 연결된 상태에서 데이터 송/수신을 수행하시기 바랍니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 1 | SERVER_CLIENT_CONNECTED | 클라이언트 접속 상태 |
| 0 | SERVER_CLIENT_DISCONNECTED | 클라이언트 미접속 상태 |
| -205 | ERROR_SERVER_CLOSED | 서버가 닫혀있는 상태 |

■ 예 제

**[예제1] serv.state 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | | |
| 4 | state= serv.state() | -- 클라이언트 접속 유/무 판단 |
| 5 | if state == 0 then | -- 클라이언트 미접속 상태 |
| 6 | serv.open() | -- 서버 생성 |
| 7 | end | |
| 8 | | |
| 9 | if state == 1 then | -- 클라이언트 접속 상태 |
| 10 | result, data=serv.read(1,3) | -- 클라이언트 데이터 수신 |
| 11 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | unexpected symbol near ']' |

| | |
|---|---|
| ■ 관련 함수 | serv=server()<br>serv.open()<br>serv.close()<br>serv.state()<br>serv.flush()<br>serv.read()<br>serv.write()<br>serv.set_label() |

## 2.9.5. serv.flush : 송/수신 버퍼 초기화

| | |
|---|---|
| ■ 기 능 | **서버(Server)의 송/수신 버퍼 데이터를 제거합니다.** |
| ■ 형 태 | serv.flush() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **버퍼를 초기화하는 명령어로 인자가 필요하지 않습니다.**

- 통신 연결 시, 이전에 사용된 데이터를 송/수신 받지 않도록 사용됩니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERVER | 초기화 완료 |
| -201 | ERROR_SERVER_FAILED | 소켓 파일 디스크립터가 유효하지 않음 |

■ 예 제

**[예제1] serv.flush 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | | |
| 4 | serv.flush() | -- 송/수신 버퍼의 데이터 초기화 |
| 5 | result, data=serv.read(1,3) | -- 데이터 수신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | unexpected symbol near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| serv=server() | | | | |
| serv.open() | | | | |
| serv.close() | | | | |
| serv.state() | | | | |
| serv.flush() | | | | |
| serv.read() | | | | |
| serv.write() | | | | |
| serv.set_label() | | | | |

## 2.9.6. serv.read : 데이터 수신

| | |
|---|---|
| ■ 기 능 | **클라이언트(Client)로부터 데이터를 수신합니다.** |
| ■ 형 태 | result, data = serv.read(Length, Timeout) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Length | int | - | 0 | 1024 | byte |
| Timeout | int | - | 0 | 600 | sec |

* 단위 : 1byte = 알파벳 1글자 (최대 1024 글자), sec (second)

■ **세부사항**

➢ **Length (수신 데이터 갯수)**

- 수신할 데이터 갯수를 지정합니다.

➢ **Timeout (수신 데이터 시간)**

- 데이터를 수신받는 시간 설정합니다.
- Timeout이 0으로 설정된 경우, 설정된 Length의 수 만큼의 데이터가 수신 될 때까지 무기한 대기합니다.
- 설정한 시간 내 데이터가 수신되지 않은 경우, 다음 스텝의 명령을 수행합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | string | 수신 데이터 문자(열) |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 데이터 수신 결과 |
| 0 | SUCCESS_SERVER | 수신 완료 |
| -201 | ERROR_SERVER_CLOSED | 서버가 닫혀있는 상태 |
| -202 | ERROR_SERVER_BAD_ARGUMENT | 서버와 관련된 명령어 인자 타입 에러 |
| -207 | ERROR_SERVER_WRITE_FAIL | 서버에 데이터 수신 실패 |
| -208 | ERROR_SERVER_MEM_ALLOC | 서버에 수신할 메모리 할당 실패 |
| -209 | ERROR_SERVER_TIMEOUT | 서버에 데이터 수신 시 타임아웃 |
| -210 | ERROR_SERVER_DISCONNECTED | 서버에 클라이언트가 연결되지 않음 |

■ 예 제

**[예제1] serve.read 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | serv_state=serv.state() | -- 서버 상태 확인 |
| 4 | if serv_state == 1 then | -- 클라이언트가 연결된 상태 |
| 5 | result, data=serv.read(10,3) | -- 대기시간 3초동안 문자열 10개 수신 |
| 6 | end | |
| 7 | if serv_state == -205 then | -- 반환 값이 -205인 경우 |
| 8 | serv.open() | -- 서버 생성 |
| 9 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[601] is range over(0~600) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| serv=server() | | | | |
| serv.open() | | | | |
| serv.close() | | | | |
| serv.state() | | | | |
| serv.flush() | | | | |
| serv.read() | | | | |
| serv.write() | | | | |
| serv.set_label() | | | | |

## 2.9.7. serv.write : 데이터 송신

| | |
|---|---|
| ■ 기 능 | **클라이언트(Client)로 데이터를 송신합니다.** |
| ■ 형 태 | serv.write(Data, Length, Timeout) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | string | - | 1 | 1024 | byte |
| Length | int | - | 1 | 1024 | - |

* 단위 : 1byte = 알파벳 1글자 (최대 1024 글자)

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 문자열 데이터를 작성합니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

➢ **Length (송신 데이터 갯수**

- 송신할 문자열 데이터의 길이를 작성합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERVER | 송신 완료 |
| -201 | ERROR_SERVER_CLOSED | 서버가 닫혀있는 상태 |
| -202 | ERROR_SERVER_BAD_ARGUMENT | 서버와 관련된 명령어 인자 타입 에러 |
| -207 | ERROR_SERVER_WRITE_FAIL | 서버에 데이터 송신 실패 |

■ 예 제

**[예제1] serv.write 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | if serv.open()==1 then | -- 클라이언트 접속 |
| 4 | data_msg="send test" | -- 전송할 문자 데이터 선언 및 초기화 |
| 5 | serv.write(data_msg, 9) | -- 데이터 송신 |
| 6 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[1025] is range over(0~1024) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| serv=server() | | | | |
| serv.open() | | | | |
| serv.close() | | | | |
| serv.state() | | | | |
| serv.flush() | | | | |
| serv.read() | | | | |
| serv.write() | | | | |
| serv.set_label() | | | | |

## 2.9.8. serv.set_label : 송신 데이터 접미사 설정

| | |
|---|---|
| ■ 기 능 | **송신 데이터 끝에 접미사를 설정합니다.** |
| ■ 형 태 | serv.set_label(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | string | - | 0 | 255 | byte |

* 단위 : 1byte = 알파벳 1글자 (최대 255 글자)

■ **세부사항**

➢ **Data (추가 송신 데이터)**

- 송신할 문자열 데이터의 접미사를 설정합니다.
- 데이터 줄 바꿈 또는 단위 설정과 같이 송신 데이터의 접미사로 활용됩니다.
- 표 2-67과 같이 설정된 추가 송신 데이터는 해제 전까지 송신 데이터 끝에 적용됩니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

| 항목 | set_label | |
|---|---|---|
| | 사용 | 미사용 |
| 송신 명령어 (write) | "send" | "send" |
| 추가 송신 명령어 (set_lable) | - | "add" |
| 실제 송신 데이터 | "send" | "sendadd" |

**표 2-67 serv.set_label 적용 예시**

※ 접미사 해제 시 인자에 ""를 입력하시기 바랍니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERVER | 설정 완료 |
| -202 | ERROR_SERVER_BAD_ARGUMENT | 서버의 데이터 인자 타입 에러 |

■ 예 제

**[예제1] serv.set_label 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | serv=server(20000) | -- 서버 선언 (Port : 20000) |
| 2 | serv.open() | -- 서버 생성 |
| 3 | | |
| 4 | serv_state=serv.state() | -- 서버 상태 확인 |
| 5 | if serv_state ==1 then | -- 서버 연결 |
| 6 | serv.set_label("₩r₩n") | -- 추가 송신 데이터 설정(₩r₩n) |
| 7 | data_msg1 = "send_test" | -- 송신 문자 데이터 선언 및 초기화 |
| 8 | serv_write=serv.write(data_msg1, 9) | -- 데이터 송신 |
| 9 | | ※ 실제 송신 데이터 : "send_test₩r₩n" |
| 10 | serv.set_label("") | -- 추가 송신 데이터 해제 |
| 11 | data_msg2 = "end" | -- 송신 문자 데이터 선언 및 초기화 |
| 12 | serv_write=serv.write(data_msg2, 7) | -- 데이터 송신 |
| 13 | end | ※ 실제 송신 데이터 : "end" |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행 오류 | 데이터 유효범위 | Value[256] is range over(0~255) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| serv=server() | | | | |
| serv.open() | | | | |
| serv.close() | | | | |
| serv.state() | | | | |
| serv.flush() | | | | |
| serv.read() | | | | |
| serv.write() | | | | |
| serv.set_label() | | | | |

## 2.10. TCP/IP 통신 클라이언트 스크립트

제어기가 TCP/IP 클라이언트(Client)로 이더넷(Ethernet) 기반의 외부기기와 통신을 수행할 수 있도록 지원하는 명령어입니다.

**표 2-68 TCP/IP 통신 클라이언트 스크립트 항목**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 연결 / 해제 | cli = client(Ip, Port) | 클라이언트 소켓 선언 | 2.10.1 |
| 2 | | cli.open() | 서버 연결 | 2.10.2 |
| 3 | | cli.close() | 서버 연결 종료 | 2.10.3 |
| 4 | 통신 상태 | cli.state() | 클라이언트 상태 확인 | 2.10.4 |
| 5 | 버퍼 초기화 | cli.flush() | 송/수신 데이터 버퍼 초기화 | 2.10.5 |
| 6 | 데이터 송신 | cli.write(Data, Timeout) | 문자열 데이터 송신 | 2.10.6 |
| 7 | | cli.set_label(Data) | 송신 문자열 데이터 접미사 설정 | 2.10.7 |
| 8 | | cli.send(Timeout) | 버퍼에 쌓인 데이터 송신 | 2.10.8 |
| 9 | | cli.add_byte(Data) | send 버퍼에 부호없는 8비트 정수형(byte) 데이터 쓰기 | 2.10.9 |
| 10 | | cli.add_short(Data) | send 버퍼에 16비트 정수형(short) 데이터 쓰기 | 2.10.10 |
| 11 | | cli.add_ushort(Data) | send 버퍼에 부호없는 16비트 정수형(unsigned short) 데이터 쓰기 | 2.10.11 |
| 12 | | cli.add_int(Data) | send 버퍼에 32비트 정수형(integer) 데이터 쓰기 | 2.10.12 |
| 13 | | cli.add_uint(Data) | send 버퍼에 부호없는 32비트 정수형(unsigned integer) 데이터 쓰기 | 2.10.13 |
| 14 | | cli.add_float(Data) | send 버퍼에 32비트 실수형(float) 데이터 쓰기 | 2.10.14 |
| 15 | | cli.add_double(Data) | send 버퍼에 64비트 실수형(double) 데이터 쓰기 | 2.10.15 |
| 16 | 데이터 수신 | cli.read(Length, Timeout) | 문자열 데이터 수신 | 2.10.16 |
| 17 | | cli.recv(Length, Timeout) | 서버로부터 수신버퍼에 데이터 수신 | 2.10.17 |
| 18 | | cli.get_byte() | recv 버퍼로부터 부호없는 8비트 정수형(byte) 데이터 읽기 | 2.10.18 |
| 19 | | cli.get_short() | recv 버퍼로부터 16비트 정수형(short) 데이터 읽기 | 2.10.19 |
| 20 | | cli.get_ushort() | recv 버퍼로부터 부호없는 16비트 정수형(unsigned short) 데이터 읽기 | 2.10.20 |
| 21 | | cli.get_int() | recv 버퍼로부터 32비트 정수형(integer) 데이터 읽기 | 2.10.21 |
| 22 | | cli.get_uint() | recv 버퍼로부터 부호없는 32비트 정수형(unsigned integer) 데이터 읽기 | 2.10.22 |
| 23 | | cli.get_float() | recv 버퍼로부터 32비트 실수형(float) 데이터 | 2.10.23 |

| | | | | |
|---|---|---|---|---|
| | | | 읽기 | |
| 24 | | cli.get_double() | recv 버퍼로부터 64비트 실수형(double) 데이터 읽기 | 2.10.24 |

표 2-68 은 TCP/IP 통신 클라이언트 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

**표 2-69 TCP/IP 통신 클라이언트 오류 반환값**

| 반환 값 | 항 목 | 설 명 |
|---|---|---|
| 0 | S_OK | 클라이언트 관련 명령어 성공 |
| 0x80004004 | E_ABORT | Stop버튼 클릭으로인한 전송 중단 |
| 0x80004006 | E_TIMEDOUT | 연결 시간 초과 |
| 0x80005004 | E_CONNREFUSED | 서버 연결 거절 |
| 0x80006003 | E_GETFCNTLFAILED | 클라이언트 소켓 설정 실패 |
| 0x80006006 | E_NONBLOCKFAILED | Non-block 설정 실패 |
| 0x80010001 | E_CONNECTIONFAILED | 서버 연결 실패 |
| 0x80010003 | E_CLIENTSOCKCLOSE | 클라이언트가 이미 종료된 상태 |
| 0x80010005 | E_INVALIDSNDPACKET | 버퍼에 데이터가 없음 |
| 0x80010009 | E_INVALIDDATASIZE | 잘못된 data 크기 |
| 0x80010014 | E_IOCTLERROR | 초기화 실패 |
| 0x8000FFFF | E_UNEXPECTED | 알 수 없는 에러 발생 |

표 2-69 는 TCP/IP 통신 클라이언트 오류 반환값에 대해 간략히 설명한 표이며, 오류 발생에 대한 조치사항은 알람 매뉴얼을 참조하시기 바랍니다.

## 2.10.1. cli = client : 서버 선언

| | |
|---|---|
| ■ 기 능 | **TCP/IP 클라이언트(Client) 소켓을 설정한 IP와 Port 번호로 선언합니다.** |
| ■ 형 태 | cli = client(Ip, Port) |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| IP | string | - | - | - | - |
| Port | int | - | 0 | 65535 | - |

■ **세부사항**

➢ **IP (IP 주소)**

- 연결할 서버의 IP 주소를 입력합니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

➢ **Port (포트 번호)**

- 연결할 서버의 Port 번호를 입력합니다.

■ 예 제

**[예제1] cli = client 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | state=cli.state() | -- 서버와 접속 유/무 판단 |
| 4 | cli.close() | -- 서버 연결 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[Port*] is range over(0~65535) |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | cli=client() | cli.add_byte() | cli.get_byte() | | |
| | cli.open() | cli.add_short() | cli.get_short() | | |
| | cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| | cli.state() | cli.add_int() | cli.get_int() | | |
| | cli.flush() | cli.add_uint() | cli.get_uint() | | |
| | cli.write() | cli.add_float() | cli.get_float() | | |
| | cli.set_label() | cli.add_double() | cli.get_double() | | |
| | cli.send() | | | - | |
| | cli.read() | | | | |
| | cli.recv() | | | | |

## 2.10.2. cli.open : 서버 연결

| | |
|---|---|
| ■ 기 능 | **서버(Server)에 TCP/IP 통신 연결합니다.** |
| ■ 형 태 | cli.open() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **서버 연결 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 항목 | 설명 |
|---|---|---|
| 0 | S_OK | 서버 연결 성공 |
| 0x80010001 | E_CONNECTIONFAILED | 서버 연결 실패 |
| 0x80006003 | E_GETFCNTLFAILED | 클라이언트 소켓 설정 실패 |
| 0x80006006 | E_NONBLOCKFAILED | Non-block 설정 실패 |
| 0x80004006 | E_TIMEDOUT | 연결 시간 초과 |
| 0x8000FFFF | E_UNEXPECTED | 알 수 없는 에러 발생 |
| 0x80005004 | E_CONNREFUSED | 서버 연결 거절 |

■ 예 제

**[예제1] cli.open 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client(20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | state=cli.state() | -- 서버와 접속 유/무 판단 |
| 4 | cli.close() | -- 서버 연결 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.3. cli.close : 서버 연결 종료

| | |
|---|---|
| ■ 기 능 | **서버(Server)와 클라이언트(Client)간 통신을 종료합니다.** |
| ■ 형 태 | cli.close() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **서버 연결 종료 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | int | 서버 연결 종료 성공 |

■ 예 제

**[예제1] cli.close 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client(20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | state=cli.state() | -- 서버와 접속 유/무 판단 |
| 4 | cli.close() | -- 서버 연결 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.4. cli.state : 서버와 연결 상태 확인

| | |
|---|---|
| ■ 기 능 | **서버(Server)와 연결 상태를 반환합니다.** |
| ■ 형 태 | cli.state() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **서버와 연결 상태 확인 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 1 | int | 서버 연결 상태 |
| 0 | int | 서버 미연결 상태 |

■ 예 제

**[예제1] cli.state 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | state=cli.state() | -- 서버와 연결 상태 판단 |
| 5 | if state == 0 then | -- 클라이언트 미연결 상태 |
| 6 | cli.open() | -- 서버 연결 |
| 7 | end | |
| 8 | | |
| 9 | if state == 1 then | -- 서버와 연결 상태 |
| 10 | result, data=cli.recv() | -- 서버 데이터 수신 |
| 11 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.5. cli.flush : 송/수신 버퍼 초기화

| | |
|---|---|
| ■ 기 능 | **클라이언트(Client)의 송/수신 버퍼 데이터를 제거합니다.** |
| ■ 형 태 | cli.flush() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **버퍼를 초기화하는 명령어로 인자가 필요하지 않습니다.**

- 통신 연결 시, 이전에 사용된 데이터를 송/수신 받지 않도록 사용됩니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | S_OK | 초기화 완료 |
| 0x80010003 | E_CLIENTSOCKCLOSE | 클라이언트가 이미 종료된 상태 |
| 0x80010014 | E_IOCTLERROR | 초기화 실패 |

■ 예 제

**[예제1] cli.flush 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli.flush() | -- 송/수신 버퍼의 데이터 초기화 |
| 5 | result, data=cli.recv() | -- 데이터 수신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.6. cli.write : 문자열 데이터 송신

■ 기 능: **서버(Server)로 문자열 데이터를 송신합니다.**

■ 형 태: cli.write(Data, [Timeout])

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | string | - | 1 | 1024 | byte |
| [Timeout] | int | 0 | 0 | 600 | sec |

* 단위 : 1byte = 알파벳 1글자 (최대 1024 글자)

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 문자열 데이터를 작성합니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

➢ **[Timeout] (송신 데이터 시간)**

- 해당 인자는 옵션 인자입니다.
- 데이터를 송신받는 시간 설정합니다.
- Timeout이 0으로 설정된 경우, Length 만큼의 데이터가 송신 될 때가지 무기한 대기합니다.
- 설정한 시간 내 데이터가 송신되지 않은 경우, 다음 스텝의 명령을 수행합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | S_OK | 송신 완료 |
| 0x80010005 | E_INVALIDSNDPACKET | 버퍼에 데이터가 없음 |
| 0x80010009 | E_INVALIDDATASIZE | 잘못된 data 크기 |
| 0x80006003 | E_GETFCNTLFAILED | 클라이언트 소켓 설정 실패 |
| 0x80006006 | E_NONBLOCKFAILED | Non-block 실패 |
| 0x80004004 | E_ABORT | Stop버튼 클릭으로인한 전송 중단 |
| 0x8000FFFF | E_UNEXPECTED | 알 수 없는 에러 발생 |

■ 예 제

**[예제1] cli.write 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | if cli.open()==0 then | -- 서버 연결 |
| 4 | data_msg="send test" | -- 전송할 문자 데이터 선언 및 초기화 |
| 5 | cli.write(data_msg) | -- 데이터 송신 |
| 6 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[1025] is range over(1~1024) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.7. cli.set_label : 송신 문자열 데이터 접미사 설정

| | |
|---|---|
| ■ 기 능 | **송신 문자열 데이터 끝에 접미사를 설정합니다.** |
| ■ 형 태 | cli.set_label(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | string | - | 0 | 255 | byte |

* 단위 : 1byte = 알파벳 1글자 (최대 255 글자)

■ **세부사항**

➢ **Data (추가 송신 데이터)**

- 송신할 문자열 데이터의 접미사를 설정합니다.
- 데이터 줄 바꿈 또는 단위 설정과 같이 송신 데이터의 접미사로 활용됩니다.
- 표 2-70과 같이 설정된 추가 송신 데이터는 해제 전까지 송신 데이터 끝에 적용됩니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

| 항목 | set_label | |
|---|---|---|
| | 사용 | 미사용 |
| 송신 명령어 (write) | "send" | "send" |
| 추가 송신 명령어 (set_lable) | - | "add" |
| 실제 송신 데이터 | "send" | "sendadd" |

**표 2-70 cli.set_label 적용 예시**

※ 접미사 해제 시 인자에 (""")를 입력하시기 바랍니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | int | 접미사 설정 성공 |

■ 예 제

**[예제1] cli.set_label 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state == 1 then | -- 서버 연결 |
| 6 | cli.set_label("₩r₩n") | -- 추가 송신 데이터 설정(₩r₩n) |
| 7 | data_msg1 = "send_test" | -- 송신 문자 데이터 선언 및 초기화 |
| 8 | cli_write=cli.write(data_msg1) | -- 데이터 송신 |
| 9 | | ※ 실제 송신 데이터 : "send_test₩r₩n" |
| 10 | cli.set_label("") | -- 추가 송신 데이터 해제 |
| 11 | data_msg2 = "end" | -- 송신 문자 데이터 선언 및 초기화 |
| 12 | cli_write=cli.write(data_msg2) | -- 데이터 송신 |
| 13 | | ※ 실제 송신 데이터 : "end" |
| 14 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[256] is range over(0~255) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.8. cli.send : 버퍼에 쌓인 데이터 송신

■ 기 능

**send 전용 버퍼에 쌓인 데이터를 송신합니다.**

■ 형 태

cli.send([Timeout])

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| [Timeout] | int | 0 | 0 | 600 | sec |

■ **세부사항**

➢ **[Timeout] (송신 데이터 시간)**

- 해당 인자는 옵션 인자입니다.
- 데이터를 송신받는 시간 설정합니다.
- Timeout이 0으로 설정된 경우, Length 만큼의 데이터가 송신 될 때가지 무기한 대기합니다.
- 설정한 시간 내 데이터가 송신되지 않은 경우, 다음 스텝의 명령을 수행합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | S_OK | 송신 완료 |
| 0x80010005 | E_INVALIDSNDPACKET | 버퍼에 데이터가 없음 |
| 0x80010009 | E_INVALIDDATASIZE | 잘못된 data 크기 |
| 0x80006003 | E_GETFCNTLFAILED | 클라이언트 소켓 설정 실패 |
| 0x80006006 | E_NONBLOCKFAILED | Non-block 실패 |
| 0x80004004 | E_ABORT | Stop버튼 클릭으로인한 전송 중단 |
| 0x8000FFFF | E_UNEXPECTED | 알 수 없는 에러 발생 |

■ 예 제

**[예제1] cli.send 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_byte(0xFF) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.9. cli.add_byte : send 버퍼에 부호없는 8비트 정수형(byte) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 부호없는 8비트 정수형(byte) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_byte(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | unsgiend char | - | 0 | 255 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-71 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_byte 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_byte(0xFF) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[256] is range over(0~255) |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.10. cli.add_short : send 버퍼에 16비트 정수형(short) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 16비트 정수형(short) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_short(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | short | - | -32,768 | 32,767 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-72 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_short 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_short(31200) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[32768] is range over(-32768~32767) |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.11. cli.add_ushort : send 버퍼에 부호없는 16비트 정수형(unsigned short) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 부호없는16비트 정수형(unsigned short) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_ushort(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | unsgiend short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-73 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_ushort 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_ushort(61200) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[-10] is range over(0~65535) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.12. cli.add_int : send 버퍼에 32비트 정수형(integer) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 32비트 정수형(integer) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_int(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | int | - | -2,147,483,648 | 2,147,483,647 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-74 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_int 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_int(68000) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2147483648] is range over(-2147483648~2147483647) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.13. cli.add_uint : send 버퍼에 부호없는 32비트 정수형(unsigned integer) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 부호없는 32비트 정수형(unsigned integer) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_uint(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | unsgiend int | - | 0 | 4,294,967,295 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-75 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_uint 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_uint(1240) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[-12] is range over(0~4294967295) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.14. cli.add_float : send 버퍼에 32비트 실수형(float) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 32비트 실수형(float) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_float(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | float | - | 3.4E-38 | 3.4E38 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-76 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_float 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_float(3.14) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.15. cli.add_double : send 버퍼에 64비트 실수형(double) 데이터를 추가

| | |
|---|---|
| ■ 기 능 | **send 버퍼에 64비트 실수형(double) 데이터를 추가합니다.** |
| ■ 형 태 | cli.add_double(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | double | - | 1.79E-308 | 1.79E308 | - |

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 데이터를 작성합니다.

| 자료형 | 비트 | 범위 |
|---|---|---|
| byte | 8 | 0 ~ 255 |
| short | 16 | -32,768 ~ 32,767 |
| unsigned short | 16 | 0 ~ 65,535 |
| int | 32 | -2,147,483,648 ~ 2,147,483,647 |
| unsigned int | 32 | 0 ~ 4,294,967,295 |
| float | 32 | 3.4E-38 ~ 34E38 |
| double | 64 | 1.79E-308 ~ 1.79E308 |

**표 2-77 데이터 자료형 및 범위**

■ 예 제

**[예제1] cli.add_double 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | cli.add_double(3.141592) | -- send 버퍼에 데이터 추가 |
| 7 | ret=cli.send() | -- send버퍼 송신 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.16. cli.read : 문자열 데이터 수신

| | |
|---|---|
| ■ 기 능 | **서버(Server)로부터 문자열 데이터를 수신합니다.** |
| ■ 형 태 | result, data = cli.read(Length, [Timeout]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Length | int | - | 0 | 1024 | byte |
| [Timeout] | int | 0 | 0 | 600 | sec |

* 단위 : 1byte = 알파벳 1글자 (최대 1024 글자), sec (second)

■ **세부사항**

➢ **Length (수신 데이터 갯수)**

- 수신할 데이터의 바이트 수를 지정합니다.

➢ **[Timeout] (수신 데이터 시간)**

- 해당 인자는 옵션 인자입니다.
- 데이터를 수신받는 시간 설정합니다.
- Timeout이 0으로 설정된 경우, Length 만큼의 데이터가 수신 될 때가지 무기한 대기합니다.
- 설정한 시간 내 데이터가 수신되지 않은 경우, 다음 스텝의 명령을 수행합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | string | 수신 데이터 문자(열) |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 데이터 수신 결과 |
| 0 | S_OK | 수신 완료 |
| 0x80010005 | E_INVALIDSNDPACKET | 버퍼에 데이터가 없음 |
| 0x80010009 | E_INVALIDDATASIZE | 잘못된 data 크기 |
| 0x80006003 | E_GETFCNTLFAILED | 클라이언트 소켓 설정 실패 |
| 0x80006006 | E_NONBLOCKFAILED | Non-block 실패 |
| 0x80004004 | E_ABORT | Stop버튼 클릭으로인한 전송 중단 |
| 0x8000FFFF | E_UNEXPECTED | 알 수 없는 에러 발생 |

■ 예 제

**[예제1] cli.read 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 4 | if cli_state == 1 then | -- 서버에 연결된 상태 |
| 5 | result, data=cli.read(10,3) | -- 대기시간 3초동안 문자열 10개 수신 |
| 6 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[601] is range over(0~600) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.17. cli.recv : 서버로부터 수신버퍼에 데이터를 수신

| | |
|---|---|
| ■ 기 능 | **서버로부터 recv 버퍼에 데이터를 수신합니다.** |
| ■ 형 태 | cli.recv(Length, [Timeout]) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Length | int | - | 0 | 1024 | byte |
| [Timeout] | int | 0 | 0 | 600 | sec |

* 단위 : 1byte = 알파벳 1글자 (최대 1024 글자)

■ **세부사항**

➢ **Length (수신 데이터 갯수)**

- 수신할 데이터 갯수를 지정합니다.

➢ **[Timeout] (송신 데이터 시간)**

- 해당 인자는 옵션 인자입니다.
- 데이터를 송신받는 시간 설정합니다.
- Timeout이 0으로 설정된 경우, Length 만큼의 데이터가 송신 될 때가지 무기한 대기합니다.
- 설정한 시간 내 데이터가 송신되지 않은 경우, 다음 스텝의 명령을 수행합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | S_OK | 수신 완료 |
| 0x80010005 | E_INVALIDSNDPACKET | 버퍼에 데이터가 없음 |
| 0x80010009 | E_INVALIDDATASIZE | 잘못된 data 크기 |
| 0x80006003 | E_GETFCNTLFAILED | 클라이언트 소켓 설정 실패 |
| 0x80006006 | E_NONBLOCKFAILED | Non-block 실패 |
| 0x80004004 | E_ABORT | Stop버튼 클릭으로인한 전송 중단 |
| 0x8000FFFF | E_UNEXPECTED | 알 수 없는 에러 발생 |

■ 예 제

**[예제1] cli.recv 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(1) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_byte() | -- 버퍼로부터 1byte 읽어서 data에 할당 |
| 9 | end | |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.18. cli.get_byte : recv 버퍼로부터 부호없는 8비트 정수형(byte) 데이터 읽기

| | |
|---|---|
| ■ 기 능 | **recv 버퍼로부터 부호없는 8비트 정수형(byte) 데이터를 가지고 옵니다.** |
| ■ 형 태 | result, data = cli.get_byte() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | unsigned char | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_byte 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_byte() | -- 버퍼로부터 byte 데이터를 읽어서 data에 할당 |
| 9 | end | |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.19. cli.get_short : recv 버퍼로부터 16비트 정수형(short) 데이터 읽기

| | |
|---|---|
| ■ 기 능 | **recv 버퍼로부터 16비트 정수형(short) 데이터를 가지고 옵니다.** |
| ■ 형 태 | result, data = cli.get_short() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | short | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_short 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_short() | -- 버퍼로부터 short 데이터를 읽어서 |
| 9 | end | data에 할당 |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.20. cli.get_ushort : recv 버퍼로부터 부호없는 16비트 정수형(unsigned short) 데이터 읽기

■ 기 능

**recv 버퍼로부터 부호없는 16비트 정수형(unsigned short) 데이터를 가지고 옵니다.**

■ 형 태

result, data = cli.get_ushort()

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | unsgiend short | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_ushort 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_ushort() | -- 버퍼로부터 unsigned short 데이터를 |
| 9 | end | 읽어서 data에 할당 |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.21. cli.get_int : recv 버퍼로부터 32비트 정수형(integer) 데이터 읽기

| | |
|---|---|
| ■ 기 능 | **recv 버퍼로부터 32비트 정수형(integer) 데이터를 가지고 옵니다.** |
| ■ 형 태 | result, data = cli.get_int() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | int | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_int 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_int() | -- 버퍼로부터 integer 데이터를 읽어서 |
| 9 | end | data에 할당 |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.22. cli.get_uint : recv 버퍼로부터 부호없는 32비트 정수형(unsigned integer) 데이터 읽기

■ 기 능

**recv 버퍼로부터 부호없는 32비트 정수형(unsigned integer) 데이터를 가지고 옵니다.**

■ 형 태

result, data = cli.get_uint()

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | unsigned int | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_uint 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_uint() | -- 버퍼로부터 unsigned integer 데이터를 읽어서 data에 할당 |
| 9 | end | |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.23. cli.get_float : recv 버퍼로부터 32비트 실수형(float) 데이터 읽기

| | |
|---|---|
| ■ 기 능 | **recv 버퍼로부터 32비트 실수형(float) 데이터를 가지고 옵니다.** |
| ■ 형 태 | result, data = cli.get_float() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | float | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_float 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_float() | -- 버퍼로부터 flaot 데이터를 읽어서 |
| 9 | end | data에 할당 |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.10.24. cli.get_doube : recv 버퍼로부터 64비트 실수형(double) 데이터 읽기

| | |
|---|---|
| ■ 기 능 | **recv 버퍼로부터 64비트 실수형(double) 데이터를 가지고 옵니다.** |
| ■ 형 태 | result, data = cli.get_double() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **recv 버퍼로부터 데이터를 가지고 오는 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | double | 수신 데이터 |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 명령어 결과값 |
| 0 | int | 정상 리턴 |
| -1 | int | 버퍼에 크기가 충분하지 않음 |

■ 예 제

**[예제1] cli.get_double 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | cli=client("192.168.1.101", 20000) | -- 클라이언트 소켓 선언 |
| 2 | cli.open() | -- 서버 연결 |
| 3 | | |
| 4 | cli_state=cli.state() | -- 서버 연결 상태 확인 |
| 5 | if cli_state ==1 then | |
| 6 | ret=cli.recv(10) | -- recv 버퍼에 데이터 수신 |
| 7 | if ret == 0 then | -- 수신 성공이면 |
| 8 | ret , data = cli.get_double() | -- 버퍼로부터 double 데이터를 읽어서 |
| 9 | end | data에 할당 |
| 10 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| cli=client() | cli.add_byte() | cli.get_byte() | | |
| cli.open() | cli.add_short() | cli.get_short() | | |
| cli.close() | cli.add_ushort() | cli.get_ushort() | | |
| cli.state() | cli.add_int() | cli.get_int() | | |
| cli.flush() | cli.add_uint() | cli.get_uint() | | |
| cli.write() | cli.add_float() | cli.get_float() | | |
| cli.set_label() | cli.add_double() | cli.get_double() | | |
| cli.send() | | | - | |
| cli.read() | | | | |
| cli.recv() | | | | |

## 2.11. Modbus 통신 서버 스크립트

제어기가 Modbus 서버(Server)로 외부기기와 통신을 수행할 수 있도록 지원하는 명령어입니다.

**표 2-78 Modbus 통신 서버 스크립트 항목**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 연결 / 해제 | modserv=modbus_server() | Modbus 서버 통신 선언 | 2.11.1 |
| 2 | 데이터 수신 | modserv.read_bit(Name) | 해당 이름의 Coil 값 읽기 | 2.11.2 |
| 3 | | modserv.read_bits({Name1, Name2, ...} | 해당 이름들의 Coil 값 읽기 | 2.11.3 |
| 4 | | modserv.read_multi(Name) | 해당하는 이름의 모든 Coil 읽기 | 2.11.4 |
| 5 | | modserv.read_register(Name) | 해당 이름의 Register 값 읽기 | 2.11.5 |
| 6 | | modserv.read_registers({Name1, Name2, ...}) | 해당 이름들의 Register 값 읽기 | 2.11.6 |
| 7 | | modserv.read_register_multi(Name) | 해당하는 이름의 모든 Resgister 읽기 | 2.11.7 |
| 8 | 데이터 송신 | modserv.write_bit(Name, Value) | 해당 이름의 Coil에 값 쓰기 | 2.11.8 |
| 9 | | modserv.write_bits({Name1, Name2, ...}, {Value1, Value2, ...}) | 해당 이름들의 Coil에 값 쓰기 | 2.11.9 |
| 10 | | modserv.write_bit_multi(Name, {Value1, Value2, ...}) | 해당하는 이름의 Coil에 순차로 값 쓰기 | 2.11.10 |
| 11 | | modserv.write_register(Name, Value) | 해당 이름의 Register에 값 쓰기 | 2.11.11 |
| 12 | | modserv.write_registers({Name1, Name2, ...}, {Value1, Value2, ...}) | 해당하는 이름의 Register에 값 쓰기 | 2.11.12 |
| 13 | | modserv.wrtire_register_multi(Name, {Value1, Value2, ...}) | 해당하는 이름의 Register에 순차로 값 쓰기 | 2.11.13 |

표 2-78 은 Modbus 통신 서버 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.11.1. modserv = modbus_server : Modbus 서버 선언

| | |
|---|---|
| ■ 기 능 | **Modbus 서버(Server)를 선언합니다.** |
| ■ 형 태 | modserv = modbus_servr() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **Modbus를 선언하는 명령어로 인자가 필요하지 않습니다.**

■ 예 제

**[예제1] modserv = modbus_server 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | modserv.write_bit(*"aa[0]"*, 1) | -- aa[0]의 Coil에 1 값을 쓰기 |
| 3 | delay(1000) | -- 1초간 대기 |
| 4 | modserv.write_bit(*"aa[0]"*, 0) | -- aa[0]의 Coil에 0 값을 쓰기 |
| 5 | delay(1000) | -- 1초간 대기 |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.2. modserv.read_bit : Coil 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 Coil의 값을 읽습니다.** |
| ■ 형 태 | Value = modserv.read_bit(Name) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |

■ **세부사항**

➢ **Name (Coil 이름)**

- 미리 선언된 Coil의 이름입니다.

➢ **Coil설정**

- 아래의 그림 2-75와 같이 [설정] – [모드버스] 탭에서 Coil를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Coil를 추가, 수정이 가능합니다.
- Coil의 주소는 0~511까지 할당 할 수 있습니다.

**그림 2-75 Coil 설정 화면**

**그림 2-76 Coil 추가, 편집**

ⓘ **Coil를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | int | 읽어 온 비트 값 |
| 1 | int | 읽어 온 비트 값 |

■ **예 제**

**[예제1] modserv.read_bit 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | modserv.write_bit(*"aa[0]"*, 1) | -- aa[0]의 Coil에 1 값 쓰기 |
| 3 | delay(1000) | -- 1초간 대기 |
| 4 | val = modserv.read_bit(*"aa[0]"*) | -- aa[0]의 Coil에 값 읽기 val=1 |
| 5 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 읽기 실패 | Failed to read a coil value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.3. modserv.read_bits : 여러 개의 Coil 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 이름들의 Coil의 값을 읽습니다.** |
| ■ 형 태 | Array = modserv.read_bits({Name1, Name2, ...}) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name1 | string | - | - | - | - |
| ... | | | | | |
| NameN | string | - | - | - | - |

■ **세부사항**

➢ **Name1, ..., N (Coil 이름)**

- 미리 선언된 Coil의 이름입니다.

➢ **Coil 설정**

- 아래의 그림 2-77과 같이 [설정] – [모드버스] 탭에서 Coil를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Coil를 추가, 수정이 가능합니다.
- Coil의 주소는 0~511까지 할당 할 수 있습니다.

**그림 2-77 Coil 설정 화면**

그림 2-78 Coil 추가, 편집

ⓘ **Coil를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| Table | 테이블 | 해당 이름의 Coil 값을 담고 있는 테이블 |

■ **예 제**

**[예제1] modserv.read_bits 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | |
| 3 | modserv.write_bit("aa[0]", 1) | -- aa[0]의 Coil에 1 값 쓰기 |
| 4 | modserv.write_bit("aa[2]", 0) | -- aa[2]의 Coil에 0 값 쓰기 |
| 5 | modserv.write_bit("bb[1]", 1) | -- aa[0]의 Coil에 1 값 쓰기 |
| 6 | | |
| 7 | val = modserv.read_bits({"aa[0]", | -- aa[0], aa[2], bb[1]의 Coil에 값 읽기 |
| 8 | "aa[2]", "bb[1]"}) | val[0]=1, val[1]=0, val[2]=1 |
| 9 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 읽기 실패 | Failed to read a coil value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.4. modserv.read_bit_multi : 해당 이름의 모든 Coil 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 모든 Coil의 값을 읽습니다.** |
| ■ 형 태 | Array = modserv.read_bit_multi(Name) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |

■ **세부사항**

➢ **Name (Coil 이름)**

- 미리 선언된 Coil의 이름입니다.

➢ **Coil 설정**

- 아래의 그림 2-79와 같이 [설정] – [모드버스] 탭에서 Coil를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Coil를 추가, 수정이 가능합니다.
- Coil의 주소는 0~511까지 할당 할 수 있습니다.

**그림 2-79 Coil 설정 화면**

**그림 2-80 Coil 추가, 편집**

ⓘ **Coil를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| Table | 테이블 | 해당 이름의 모든 Coil 값을 담고 있는 테이블 |

■ **예 제**

**[예제1] modserv.read_bit_multi 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | ※ aa[0] ~ [2] Coil 추가 해놓은 상태 |
| 3 | modserv.write_bit("aa[0]", 1) | -- aa[0]의 Coil에 1 값 쓰기 |
| 4 | modserv.write_bit("aa[1]", 0) | -- aa[1]의 Coil에 0 값 쓰기 |
| 5 | modserv.write_bit("aa[2]", 1) | -- aa[2]의 Coil에 1 값 쓰기 |
| 6 | | |
| 7 | val = modserv.read_bit_multi("aa") | -- aa의 모든 Coil에 값 읽기(aa[0] ~ |
| 8 | | [2]) varl[0]=1, val[1]=0, val[2]=1 |
| 9 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 읽기 실패 | Failed to read a coil value |
| | 존재하지 않는 인자 | cc[0] |
| | 명령어 괄호 | aa[0](remove '[', ']') |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.5. modserv.read_register : Holding Register 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 Holding Register의 값을 읽습니다.** |
| ■ 형 태 | Value = modserv.read_register(Name) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |

■ **세부사항**

➢ **Name (Holding Register 이름)**

- 미리 선언된 Holding Register의 이름입니다.

➢ **Register 설정**

- 아래의 그림 2-81과 같이 [설정] – [모드버스] 탭에서 Holding Register를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Holding Register를 추가, 수정이 가능합니다.
- Holding Register의 주소는 301~555까지 할당 할 수 있습니다.

**그림 2-81 Holding Register 설정 화면**

**그림 2-82 Holding Register 추가, 편집**

ⓘ **Holding Register를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| -32,768~32,767 | int | Holding Register의 값 |

■ **예 제**

**[예제1] modserv.read_register 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | modserv.write_register(*"dd0]"*, 0x45) | -- dd [0]의 Holding Register에 0x45 |
| 3 | | 값 쓰기 |
| 4 | delay(1000) | -- 1초간 대기 |
| 5 | val = modserv.read_register(*"dd[0]"*) | -- dd[0]의 Register에 값 읽기 |
| 6 | | val=0x45 |
| 7 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 읽기 실패 | Failed to read a register value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.6. modserv.read_registers : 여러 개의 Holding Register 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 이름들의 Holding Register의 값을 읽습니다.** |
| ■ 형 태 | Array = modserv.read_registers({Name1, Name2, ...}) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name1 | string | - | - | - | - |
| ... | | | | | |
| NameN | string | - | - | - | - |

■ **세부사항**

➢ **Name1, ..., N (Holding Register 이름)**

- 미리 선언된 Holding Register의 이름입니다.

➢ **Register 설정**

- 아래의 그림 2-83과 같이 [설정] – [모드버스] 탭에서 Holding Register를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Holding Register를 추가, 수정이 가능합니다.
- Holding Register의 주소는 301~555까지 할당 할 수 있습니다.

**그림 2-83 Holding Register 설정 화면**

그림 2-84 Holding Register 추가, 편집

ⓘ **Holding Register를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| Table | 테이블 | 해당 이름의 Holding Register 값을 담고 있는 테이블 |

■ **예 제**

**[예제1] modserv.read_registers 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | |
| 3 | modserv.write_register(*"dd[0]"*, 69) | -- dd[0]의 Holding Register에 69 값 쓰기 |
| 4 | | |
| 5 | modserv.write_register(*"ee[3]"*, 40) | -- ee[2]의 Holding Register에 40 값 쓰기 |
| 6 | | |
| 7 | val = modserv.read_registers({*"dd[0]"*, | -- dd[0], ee[3]의 Holding Register에 값 읽기 val[0]=69, val[1]=40 |
| 8 | *"ee[3]"*}) | |
| 9 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 읽기 실패 | Failed to read a register value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.7. modserv.read_register_multi : 해당 이름의 모든 Holding Register 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 모든 Holding Register의 값을 읽습니다.** |
| ■ 형 태 | Array = modserv.read_register_multi(Name) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |

➢ **Name (Holding Register 이름)**

- 미리 선언된 Holding Register의 이름입니다.

➢ **Register 설정**

- 아래의 그림 2-85와 같이 [설정] – [모드버스] 탭에서 Holding Register를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Holding Register를 추가, 수정이 가능합니다.
- Holding Register의 주소는 301~555까지 할당 할 수 있습니다.

**그림 2-85 Holding Register 설정 화면**

**그림 2-86 Holding Register 추가, 편집**

ⓘ **Holding Register를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| Table | 테이블 | 해당 이름의 모든 Holding Register 값을 담고 있는 테이블 |

■ **예 제**

**[예제1] modserv.read_register_multi 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | ※ dd[0] ~ [2] Register 추가 해놓은 상태 |
| 3 | | |
| 4 | modserv.write_register("dd[0]", 30) | -- dd[0]의 Holding Register에 30 값 쓰기 |
| 5 | | |
| 6 | modserv.write_register("dd[1]", 61) | -- dd[1]의 Holding Register에 61 값 쓰기 |
| 7 | | |
| 8 | modserv.write_register("dd[2]", 8) | -- dd[2]의 Holding Register에 8 값 쓰기 |
| 9 | | |
| 10 | val = | -- dd의 모든 Holding Register에 값 읽기(dd[0] ~ [2]) |
| 11 | modserv.read_register_multi("dd") | |
| 12 | | val[0]=30, val[1]=61, val[2]=8 |
| 13 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 읽기 실패 | Failed to read a register value |
| | 존재하지 않는 인자 | cc[0] |
| | 명령어 괄호 | aa[0](remove '[', ']') |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.8. modserv.write_bit : Coil 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 Coil에 값을 씁니다.** |
| ■ 형 태 | modserv.write_bit(Name, Value) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |
| Value | unsigned char | - | 0 | 1 | - |

■ **세부사항**

➢ **Name (Coil 이름)**

- 미리 선언된 Coil의 이름입니다.

➢ **Value (Coil 값)**

- Coil에 입력할 수 있는 값입니다.
- 0, 1을 넣어 해당 Coil를 설정할 수 있습니다.

➢ **Coil 설정**

- 아래의 그림 2-87과 같이 [설정] – [모드버스] 탭에서 Coil를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Coil를 추가, 수정이 가능합니다.
- Coil의 주소는 0~511까지 할당 할 수 있습니다.

**그림 2-87 Coil 설정 화면**

그림 2-88 Coil 추가, 편집

ⓘ **Coil를 설정은 수동모드에서 가능합니다.**

■ **예 제**

**[예제1] modserv.write_bit 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | modserv.write_bit(*"aa[0]"*, 1) | -- aa[0]의 Coil에 1 값 쓰기 |
| 3 | delay(1000) | -- 1초간 대기 |
| 4 | val = modserv.read_bit(*"aa[0]"*) | -- aa[0]의 Coil에 값 읽기 val=1 |
| 5 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 쓰기 실패 | Failed to write a coil value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.9. modserv.write_bits : 여러 개의 Coil 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 이름들의 Coil에 값을 씁니다.** |
| ■ 형 태 | modserv.read_bits({Name1, Name2, ...}, {Value1, Value2, ...}) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name1 | string | - | - | - | - |
| ... | | | | | |
| NameN | string | - | - | - | - |
| Value1 | unsigned char | - | 0 | 1 | - |
| ... | | | | | |
| ValueN | unsigned char | - | 0 | 1 | - |

■ **세부사항**

➢ **Name1, ..., N (Coil 이름)**

- 미리 선언된 Coil의 이름입니다.

➢ **Value1, ..., N (Coil 값)**

- Coil에 입력할 수 있는 값입니다.
- 0, 1을 넣어 해당 Coil를 설정할 수 있습니다.

➢ **Coil 설정**

- 아래의 그림 2-89과 같이 [설정] – [모드버스] 탭에서 Coil를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Coil를 추가, 수정이 가능합니다.
- Coil의 주소는 0~511까지 할당 할 수 있습니다.

**그림 2-89 Coil 설정 화면**

그림 2-90 Coil 추가, 편집

ⓘ **Coil를 설정은 수동모드에서 가능합니다.**

■ **예 제**

**[예제1] modserv.write_bits 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | |
| 3 | modserv.write_bits({*"aa[0]", aa[2]"*}, {1, | -- aa[0], aa[2]의 Coil에 각각 1, 0 값 |
| 4 | 0}) | 쓰기 |
| 5 | | |
| 6 | val = modserv.read_bits({*"aa[0]",* | -- aa[0], aa[2] 의 Coil에 값 읽기 |
| 7 | *"aa[2]"*}) | val[0]=1, val[1]=0 |
| 8 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 쓰기 실패 | Failed to write a coil value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.10. modserv.write_bit_multi : 해당 이름의 모든 Coil에 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 모든 Coil에 값을 씁니다.** |
| ■ 형 태 | modserv.read_bit_multi(Name, {Value1, Value2, ...}) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |
| Value1 | unsigned char | - | 0 | 1 | - |
| ... | | | | | |
| ValueN | unsigned char | - | 0 | 1 | - |

■ **세부사항**

➢ **Name (Coil 이름)**

- 미리 선언된 Coil의 이름입니다.

➢ **Value1, ..., N (Coil 값)**

- Coil에 입력할 수 있는 값입니다.
- 0, 1을 넣어 해당 Coil를 설정할 수 있습니다.

➢ **Coil 설정**

- 아래의 그림 2-91과 같이 [설정] – [모드버스] 탭에서 Coil를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Coil를 추가, 수정이 가능합니다.
- Coil의 주소는 0~511까지 할당 할 수 있습니다.

**그림 2-91 Coil 설정 화면**

그림 2-92 Coil 추가, 편집

ⓘ **Coil를 설정은 수동모드에서 가능합니다.**

■ **예 제**

**[예제1] modserv.write_bit_multi 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | ※ aa[0] ~ [2] Coil 추가 해놓은 상태 |
| 3 | modserv.write_bit_multi("*aa*", {1, 1, 0}) | -- aa의 모든 Coil에 값 쓰기(aa[0] ~ |
| 4 | | [2]) |
| 5 | val = modserv.read_bit_multi(*"aa"*) | -- aa의 모든 Coil에 값 읽기(aa[0] ~ |
| 6 | | [2]) val[0]=1, val[1]=1, val[2]=0 |
| 7 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 쓰기 실패 | Failed to write a coil value |
| | 존재하지 않는 인자 | cc[0] |
| | 명령어 괄호 | aa[0](remove '[', ']') |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.11. modserv.write_register : Holding Register 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 이름의 Holding Register에 값을 씁니다.** |
| ■ 형 태 | modserv.write_register(Name, Value) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |
| Value | unsgiend short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Name (Holding Register 이름)**

- 미리 선언된 Holding Register의 이름입니다.

➢ **Value (Holding Register 값)**

- Holding Register에 입력할 수 있는 값입니다.

➢ **Register 설정**

- 아래의 그림 2-93과 같이 [설정] – [모드버스] 탭에서 Holding Register를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Holding Register를 추가, 수정이 가능합니다.
- Holding Register의 주소는 301~555까지 할당 할 수 있습니다.

**그림 2-93 Holding Register 설정 화면**

그림 2-94 Holding Register 추가, 편집

ⓘ **Holding Register를 설정은 수동모드에서 가능합니다.**

■ **예 제**

**[예제1] modserv.write_register 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | modserv.write_register(*"dd0]"*, 0x45) | -- dd [0]의 Holding Register에 0x45 값 쓰기 |
| 3 | | |
| 4 | delay(1000) | -- 1초간 대기 |
| 5 | val = modserv.read_register(*"dd[0]"*) | -- dd[0]의 Holding Register에 값 읽기 val=0x45 |
| 6 | | |
| 7 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 쓰기 실패 | Failed to write a register value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.12. modserv.write_registers : 여러 개의 Holding Register 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 이름들의 Holding Register에 값을 씁니다.** |
| ■ 형 태 | modserv.read_registers({Name1, Name2, ...}, {Value1, Value2, ...{) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name1 | string | - | - | - | - |
| ... | | | | | |
| NameN | string | - | - | - | - |
| Value1 | unsigned short | - | 0 | 65,535 | - |
| ... | | | | | |
| ValueN | unsigned short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Name1, ..., N (Holding Register 이름)**

- 미리 선언된 Holding Register의 이름입니다.

➢ **Value (Holding Register 값)**

- Holding Register에 입력할 수 있는 값입니다.

➢ **Register 설정**

- 아래의 그림 2-95와 같이 [설정] – [모드버스] 탭에서 Holding Register를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Register를 추가, 수정이 가능합니다.
- Holding Register의 주소는 301~555까지 할당 할 수 있습니다.

**그림 2-95 Holding Register 설정 화면**

그림 2-96 Holding Register 추가, 편집

ⓘ **Holding Register를 설정은 수동모드에서 가능합니다.**

■ **반환 값**

| 값 | 자료형 | 설명 |
|---|---|---|
| 0x00~0xFF | int | Holding Register의 값 |

■ **예 제**

**[예제1] modserv.write_registers 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | |
| 3 | modserv.write_registers({"dd[0]", | -- dd[0]에 69, ee[3]에 40 값 쓰기 |
| 4 | "ee[3]"}, {69, 40}) | |
| 5 | | |
| 6 | val = modserv.read_registers({"dd[0]", | -- dd[0], ee[3]의 Holding Register에 값 읽기 |
| 7 | "ee[3]"}) | val[0]-69, val[1]=40 |
| 8 | delay(1000) | -- 1초간 대기 |
| 9 | | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 쓰기 실패 | Failed to write a register value |
| | 존재하지 않는 인자 | cc[0] |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.11.13. modserv.write_register_multi : 해당 이름의 모든 Holding Register에 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 이름들의 모든 Holding Register에 값을 씁니다.** |
| ■ 형 태 | modserv.write_register_multi(Name, {Value1, Value2, ...}) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Name | string | - | - | - | - |
| Value1 | unsigned short | - | 0 | 65,535 | - |
| ... | | | | | |
| ValueN | unsigned short | - | 0 | 65,535 | - |

➢ **Name (Holding Register 이름)**

- 미리 선언된 Holding Register의 이름입니다.

➢ **Value (Holding Register 값)**

- Holding Register에 입력할 수 있는 값입니다.

➢ **Register 설정**

- 아래의 그림 2-97과 같이 [설정] – [모드버스] 탭에서 Holding Register를 설정 할수 있습니다.
- 추가, 편집 버튼을 이용하여 Holding Register를 추가, 수정이 가능합니다.
- Holding Register의 주소는 301~555까지 할당 할 수 있습니다.

**그림 2-97 Holding Register 설정 화면**

그림 2-98 Holding Register 추가, 편집

ⓘ **Holding Register를 설정은 수동모드에서 가능합니다.**

■ **예 제**

**[예제1] modserv.write_register_multi 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modserv = modbus_server() | -- 모드버스 서버 선언 |
| 2 | | ※ dd[0] ~ [2] Register 추가 해놓은 상태 |
| 3 | | |
| 4 | modserv.write_register_multi("dd", {30, | -- dd의 Holding Register에 30, 61, 8 값 쓰기 |
| 5 | 61, 8}) | |
| 6 | | |
| 7 | val = | -- dd의 모든 Holding Register에 값 읽기(dd[0] ~ [2]) val[0]=30, val[1]=61, val[8] |
| 8 | modserv.read_register_multi("dd") | |
| 9 | | |
| 10 | delay(1000) | -- 1초간 대기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 인자 없음 | empty |
| | 서버 생성되지 않음 | Cannot use Modbus server |
| | 데이터 쓰기 실패 | Failed to write a register value |
| | 존재하지 않는 인자 | cc[0] |
| | 명령어 괄호 | aa[0](remove '[', ']') |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modserv=modbus_server() | | | | |
| | | | | |
| modserv.read_bit() | | | | |
| modserv.read_bits() | | | | |
| modserv.read_bit_multi() | | | | |
| modserv.read_register() | | | | |
| modserv.read_registers() | | | | |
| modserv.read_register_multi() | | | | |
| | | | | |
| modserv.write_bit() | | | | |
| modserv.write_bits() | | | | |
| modserv.write_bit_multi() | | | | |
| modserv.write_register() | | | | |
| modserv.write_registers() | | | | |
| modserv.write_register_multi() | | | | |

## 2.12. Modbus 통신 클라이언트 스크립트

제어기가 Modbus 클라이언트(Client)로 외부기기와 통신을 수행할 수 있도록 지원하는 명령어입니다.

**표 2-79 Modbus 통신 스크립트 항목**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 연결 / 해제 | modcli=modbus_client() | Modbus 클라이언트 통신 선언 | 2.12.1 |
| 2 | | modcli.open(Type, Address, Port, Slave) | Modbus 서버와 연결 | 2.12.2 |
| 3 | | modcli.close() | Modbus 서버와 연결 해제 | 2.12.3 |
| 4 | 데이터 수신 | modcli.read_bit(Address) | 해당 주소의 Coil 값 1개 읽기 | 2.12.4 |
| 5 | | modcli.read_bits(Address, Number) | 해당 주소의 Number 만큼의 Coil 값 읽기 | 2.12.5 |
| 6 | | modcli.read_register(Address) | 해당 주소의 Holding Register 값 1개 읽기 | 2.12.6 |
| 7 | | modcli.read_registers(Address, Number) | 해당 주소의 Number 만큼의 Holding Register 값 읽기 | 2.12.7 |
| 8 | 데이터 송신 | modcli.write_bit(Address, Value) | 해당 주소의 Coil에 값 쓰기 | 2.12.12 |
| 9 | | modcli.write_bits(Address, Number, Array) | 해당 주소의 Coil에 여러 개의 값 쓰기 | 2.12.13 |
| 10 | | modcli.write_register(Address, Value) | 해당 주소의 Holding Register에 값 스기 | 2.12.14 |
| 11 | | modcli.wrtie_registers(Address, Number, Array) | 해당 주소의 Holding Register에 여러 개의 값 쓰기 | 2.12.15 |

표 2-79 는 Modbus 통신 클라이언트 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.12.1. modcli = modbus_client() : Modbus 클라이언트 선언

| | |
|---|---|
| ■ 기 능 | **Modbus 클라이언트(Client)를 선언합니다.** |
| ■ 형 태 | modcli = modbus_client() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **Modbus를 선언하는 명령어로 인자가 필요하지 않습니다.**

■ 예 제

**[예제1] modcli = modbus_client 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modcli=modbus_client() | | | | |
| modcli.open() | | | | |
| modcli.close() | | | | |
| | | | | |
| modcli.read_bit() | | | | |
| modcli.read_bits() | | | | |
| modcli.read_register() | | | | |
| modcli.read_registers() | | | | |
| | | | | |
| modcli.write_bit() | | | | |
| modcli.write_bits() | | | | |
| modcli.write_register() | | | | |
| modcli.write_registers() | | | | |

## 2.12.2. modcli.open : Modbus 서버와 연결

| | |
|---|---|
| ■ 기 능 | **Modbus 서버와 연결을 시도합니다.** |
| ■ 형 태 | modcli.open(Type, Address, Port, Slave) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Type | string | - | "TCP", "RTU | | - |
| Address | string | - | - | - | - |
| Port | unsgiend int | - | 0 | 256,000 | - |
| Slave | unsgiend short | - | 0 | 65,535 | |

■ **세부사항**

➢ **Type (Modbus 통신 타입)**

- Modbus 통신 타입을 설정합니다. TCP와 RTU 타입 중에 선택합니다.

➢ **Address (Modbus 서버 주소)**

- 연결할 Modbus 서버의 주소를 입력합니다.

➢ **Port (Modbus 통신 포트)**

- Modbus 통신을 주고 받을 포트 번호를 입력합니다.

➢ **Slave (Modbus 슬레이브 ID)**

- Modbus 슬레이브의 ID를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

■ 예 제

**[예제1] modcli.open 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[327688] is range over(0~65535) |
| | Modbus 타입 입력 | Invalid Modbus Type |
| | 서버 주소 입력 | Failed to open the Modbus port |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modcli=modbus_client() | | | | |
| modcli.open() | | | | |
| modcli.close() | | | | |
| | | | | |
| modcli.read_bit() | | | | |
| modcli.read_bits() | | | | |
| modcli.read_register() | | | | |
| modcli.read_registers() | | | | |
| | | | | |
| modcli.write_bit() | | | | |
| modcli.write_bits() | | | | |
| modcli.write_register() | | | | |
| modcli.write_registers() | | | | |

## 2.12.3. modcli.close : Modbus 서버와 연결 해제

| | |
|---|---|
| ■ 기 능 | **Modbus 서버와의 연결을 해제합니다.** |
| ■ 형 태 | modcli.close() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **Modbus를 해제하는 명령어로 인자가 필요하지 않습니다.**

■ 예 제

**[예제1] modcli.close 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| modcli=modbus_client() | | | | |
| modcli.open() | | | | |
| modcli.close() | | | | |
| | | | | |
| modcli.read_bit() | | | | |
| modcli.read_bits() | | | | |
| modcli.read_register() | | | | |
| modcli.read_registers() | | | | |
| | | | | |
| modcli.write_bit() | | | | |
| modcli.write_bits() | | | | |
| modcli.write_register() | | | | |
| modcli.write_registers() | | | | |

## 2.12.4. modcli.read_bit : Coil 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Coil 값 1개를 읽습니다.** |
| ■ 형 태 | Value = modcli.read_bit(Address) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsgiend short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Address (Coil 주소)**

- 값을 읽어 올 Coil의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | unsgiend char | 읽어 온 비트 값 |
| 1 | unsigned char | 읽어 온 비트 값 |

■ 예 제

**[예제1] modcli.read_bit 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | | |
| 5 | val = modcli.read_bit(0x42) | -- 0x42 주소에 있는 Coil 값 읽기 |
| 6 | delay(1000) | -- 1초간 대기 |
| 7 | | |
| 8 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 읽기 실패 | Failed to read a coil value |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | modcli=modbus_client() | | | | |
| | modcli.open() | | | | |
| | modcli.close() | | | | |
| | | | | | |
| | modcli.read_bit() | | | | |
| | modcli.read_bits() | | | | |
| | modcli.read_register() | | | | |
| | modcli.read_registers() | | | | |
| | | | | | |
| | modcli.write_bit() | | | | |
| | modcli.write_bits() | | | | |
| | modcli.write_register() | | | | |
| | modcli.write_registers() | | | | |

## 2.12.5. modcli.read_bits : 여러 개의 Coil 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 지정 개수 만큼의 Coil 값을 읽습니다.** |
| ■ 형 태 | Array = modcli.read_bits(Address, Number) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsgiend short | - | 0 | 65,535 | - |
| Number | unsigned short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Address (Coil 주소)**

- 값을 읽어 올 Coil의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

➢ **Number (읽어 올 Coil 개수)**

- 해당 주소의 메모리에서 몇 개의 Coil을 가져올 것인지 입력합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| Table | 테이블 | 지정한 개수만큼의 Coil 값을 담고 있는 테이블 |

■ 예 제

**[예제1] modcli.read_bits 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | | |
| 5 | val = modcli.read_bits(0x42, 4) | -- 0x42 주소에 있는 Coil 값 4개 읽기 |
| 6 | delay(1000) | -- 1초간 대기 |
| 7 | | |
| 8 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 읽기 실패 | Failed to read a coil value |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| modcli=modbus_client() | | | | |
| modcli.open() | | | | |
| modcli.close() | | | | |
| | | | | |
| modcli.read_bit() | | | | |
| modcli.read_bits() | | | | |
| modcli.read_register() | | | | |
| modcli.read_registers() | | | | |
| | | | | |
| modcli.write_bit() | | | | |
| modcli.write_bits() | | | | |
| modcli.write_register() | | | | |
| modcli.write_registers() | | | | |

## 2.12.6. modcli.read_register : Holding Register 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Holding Register 값 1개를 읽습니다.** |
| ■ 형 태 | Value = modcli.read_register(Address) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsigned short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Address (Holding Register 주소)**

- 값을 읽어 올 Holding Register의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0~65,535 | unsgiend short | Holding Register의 값 |

■ 예 제

**[예제1] modcli.read_register 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | | |
| 5 | val = modcli.read_register(0x53) | -- 0x53 주소에 있는 Holding Register |
| 6 | | 값 읽기 |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | | |
| 9 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 읽기 실패 | Failed to read a register value |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | modcli=modbus_client() | | | | |
| | modcli.open() | | | | |
| | modcli.close() | | | | |
| | | | | | |
| | modcli.read_bit() | | | | |
| | modcli.read_bits() | | | | |
| | modcli.read_register() | | | | |
| | modcli.read_registers() | | | | |
| | | | | | |
| | modcli.write_bit() | | | | |
| | modcli.write_bits() | | | | |
| | modcli.write_register() | | | | |
| | modcli.write_registers() | | | | |

## 2.12.7. modcli.read_registers : 여러 개의 Holding Register 값 읽기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 지정 개수 만큼의 Holding Register 값을 읽습니다.** |
| ■ 형 태 | Array = modcli.read_registers(Address, Number) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsgiend short | - | 0 | 65,535 | - |
| Number | unsigned short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Address (Holding Register 주소)**

- 값을 읽어 올 Holding Register의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

➢ **Number (읽어 올 Holding Register 개수)**

- 해당 주소의 메모리에서 몇 개의 Holding Register을 가져올 것인지 입력합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| Table | 테이블 | 지정한 개수만큼의 Holding Register 값을 담고 있는 테이블 |

■ 예 제

**[예제1] modcli.read_registers 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | | |
| 5 | val = modcli.read_registers(0x53, 3) | -- 0x53 주소에 있는 Holding 값 3개 읽기 |
| 6 | | |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | | |
| 9 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 읽기 실패 | Failed to read a register value |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| modcli=modbus_client() | | | | |
| modcli.open() | | | | |
| modcli.close() | | | | |
| | | | | |
| modcli.read_bit() | | | | |
| modcli.read_bits() | | | | |
| modcli.read_register() | | | | |
| modcli.read_registers() | | | | |
| | | | | |
| modcli.write_bit() | | | | |
| modcli.write_bits() | | | | |
| modcli.write_register() | | | | |
| modcli.write_registers() | | | | |

## 2.12.8. modcli.write_bit : Coil에 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Coil에 값을 씁니다.** |
| ■ 형 태 | modcli.write_bit(Address, Value) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsgiend short | - | 0 | 65,535 | - |
| Value | unsigned char | - | 0 | 1 | - |

■ **세부사항**

➢ **Address (Coil 주소)**

- 값을 쓸 Coil의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

➢ **Value (입력 값)**

- 지정한 주소의 Coil에 쓸 값을 입력합니다.

■ 예 제

**[예제1] modcli.write_bit 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | | |
| 5 | modcli.write_bit(0x42, 0) | -- 0x42 주소에 있는 Coil에 0 값 쓰기 |
| 6 | delay(1000) | -- 1초간 대기 |
| 7 | | |
| 8 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 쓰기 실패 | Failed to write a coil value |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | modcli=modbus_client() | | | | |
| | modcli.open() | | | | |
| | modcli.close() | | | | |
| | | | | | |
| | modcli.read_bit() | | | | |
| | modcli.read_bits() | | | | |
| | modcli.read_register() | | | | |
| | modcli.read_registers() | | | | |
| | | | | | |
| | modcli.write_bit() | | | | |
| | modcli.write_bits() | | | | |
| | modcli.write_register() | | | | |
| | modcli.write_registers() | | | | |

## 2.12.9. modcli.write_bits : 여러 개의 Coil에 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Coil에 지정한 만큼의 값을 씁니다.** |
| ■ 형 태 | modcli.write_bits(Address, Array) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsigned short | - | 0 | 65,535 | - |
| Array | 테이블 | - | 0 | 1 | - |

■ **세부사항**

➢ **Address (Coil 주소)**

- 값을 쓸 Coil의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

➢ **Array (입력 값들의 배열)**

- Coil들에 입력될 값들입니다.
- 중괄호 '{ }'를 이용하여 배열 형식으로 작성합니다.
  예) val = {1, 0, 1, 1}

■ 예 제

**[예제1] modcli.write_bits 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open("TCP", "192.168.0.10", 1502, 0x01) | -- 모드버스 TCP로 서버와 연결 |
| 3 | | |
| 4 | arr={0, 1, 0, 0, 1} | -- 0x42 주소에 있는 Coil에 arr 테이블 중 5개 값 쓰기 |
| 5 | modcli.write_bits(0x42, arr) | |
| 6 | delay(1000) | -- 1초간 대기 |
| 7 | | |
| 8 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 쓰기 실패 | Failed to write a coil value |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | modcli=modbus_client() | | | | |
| | modcli.open() | | | | |
| | modcli.close() | | | | |
| | | | | | |
| | modcli.read_bit() | | | | |
| | modcli.read_bits() | | | | |
| | modcli.read_register() | | | | |
| | modcli.read_registers() | | | | |
| | | | | | |
| | modcli.write_bit() | | | | |
| | modcli.write_bits() | | | | |
| | modcli.write_register() | | | | |
| | modcli.write_registers() | | | | |

## 2.12.10. modcli.write_register : Holding Register에 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Holding Register에 값을 씁니다.** |
| ■ 형 태 | modcli.write_register(Address, Value) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsigned short | - | 0 | 65,535 | - |
| Value | unsigned short | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Address (Holding Register 주소)**

- 값을 쓸 Holding Register의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

➢ **Value (입력 값)**

- 지정한 주소의 Holding Register에 쓸 값을 입력합니다.

■ 예 제

**[예제1] modcli.write_register 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open(*"TCP"*, *"192.168.0.10"*, | -- 모드버스 TCP로 서버와 연결 |
| 3 | 1502, 0x01) | |
| 4 | | |
| 5 | modcli.write_register(0x55, 65) | -- 0x55 주소에 있는 Holding |
| 6 | | Register에 65 값 쓰기 |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | | |
| 9 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 쓰기 실패 | Failed to write a register value |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | modcli=modbus_client() | | | | |
| | modcli.open() | | | | |
| | modcli.close() | | | | |
| | | | | | |
| | modcli.read_bit() | | | | |
| | modcli.read_bits() | | | | |
| | modcli.read_register() | | | | |
| | modcli.read_registers() | | | | |
| | | | | | |
| | modcli.write_bit() | | | | |
| | modcli.write_bits() | | | | |
| | modcli.write_register() | | | | |
| | modcli.write_registers() | | | | |

## 2.12.11. modcli.write_registers : 여러 개의 Holding Register에 값 쓰기

| | |
|---|---|
| ■ 기 능 | **해당 주소의 Holding Register에 지정한 만큼의 값을 씁니다.** |
| ■ 형 태 | modcli.write_registers(Address, Array) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Address | unsigned short | - | 0 | 65,535 | - |
| Array | 테이블 | - | 0 | 65,535 | - |

■ **세부사항**

➢ **Address (Holding Register 주소)**

- 값을 쓸 Holding Register의 주소를 입력합니다.
- 해당 값은 10진수, 16진수 모두 작성 가능합니다.

➢ **Array (입력 값들의 배열)**

- Holding Register들에 입력될 값들입니다.
- 중괄호 '{ }'를 이용하여 배열 형식으로 작성합니다.
  예) val = {65, 100, 85}

■ 예 제

**[예제1] modcli.write_registers 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | modcli = modbus_client() | -- 모드버스 클라이언트 선언 |
| 2 | modcli.open("TCP", "192.168.0.10", 1502, 0x01) | -- 모드버스 TCP로 서버와 연결 |
| 3 | | |
| 4 | arr={65, 100, 85} | -- 0x55 주소에 있는 Holding Register에 arr 테이블 중 5개 값 쓰기 |
| 5 | modcli.write_registers(0x55, arr) | |
| 6 | delay(1000) | -- 1초간 대기 |
| 7 | | |
| 8 | modcli.close() | -- 모드버스 서버와 연결 해제 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 쓰기 실패 | Failed to write a register value |

| | | | | | |
|---|---|---|---|---|---|
| ■ 관련 함수 | modcli=modbus_client() | | | | |
| | modcli.open() | | | | |
| | modcli.close() | | | | |
| | | | | | |
| | modcli.read_bit() | | | | |
| | modcli.read_bits() | | | | |
| | modcli.read_register() | | | | |
| | modcli.read_registers() | | | | |
| | | | | | |
| | modcli.write_bit() | | | | |
| | modcli.write_bits() | | | | |
| | modcli.write_register() | | | | |
| | modcli.write_registers() | | | | |

## 2.13. 시리얼 통신 스크립트

시리얼(Serial) 통신 기반의 외부기기와 통신을 수행할 수 있도록 지원하는 명령어 입니다.

**표 2-80 시리얼 통신 스크립트 항목**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 연결 / 해제 | seri=serial(Port, Baudrate, Data bit, Paritybit, Stopbit) | 통신 포트 선언 | 2.13.1 |
| 2 | | seri.open() | 통신 포트 열기 | 2.13.2 |
| 3 | | seri.close() | 통신 포트 닫기 | 2.13.3 |
| 4 | 통신 상태 | seri.state() | 통신 포트 상태 확인 | 2.13.4 |
| 5 | 데이터 수신 | seri.read(Length, Timeout) | 데이터 읽기 | 2.13.5 |
| 6 | 데이터 송신 | seri.write(Data, Length) | 데이터 쓰기 | 2.13.6 |
| 7 | 데이터 설정 | seri.set_byte_timeout(Time) | 수신 데이터 문자간의 타임아웃(Timeout) 설정 | 2.13.7 |
| 8 | | seri.set_label(Data) | 송신 데이터의 접미사 지정 | 2.13.8 |
| 9 | 데이터 개수 확인 | seri.count_rx() | 수신 데이터 개수 확인 | 2.13.9 |
| 10 | | seri.count_tx() | 송신 데이터 개수 확인 | 2.13.10 |
| 11 | 버퍼 초기화 | seri.flush_rx() | 수신 데이터 버퍼 초기화 | 2.13.11 |
| 12 | | seri.flush_tx() | 송신 데이터 버퍼 초기화 | 2.13.12 |
| 13 | | seri.flush() | 송/수신 데이터 버퍼 초기화 | 2.13.13 |

표 2-80 은 시리얼 통신 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

**표 2-81 시리얼 통신 오류 반환값**

| 반환 값 | 항 목 | 설 명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 시리얼 통신 연결이 성공한 경우 |
| -101 | ERROR_SERIAL_FAIL | 시리얼 통신 디스크립터가 유효하지 않은 경우 |
| -102 | ERROR_SERIAL_BAD_ARGUMENT | 시리얼 데이터 인자 타입 |
| -103 | ERROR_SERIAL_OPENED | 시리얼 통신이 연결된 상태 |
| -104 | ERROR_SERIAL_OPEN_FAIL | 시리얼 통신 연결 실패 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신이 연결되지 않은 상태 |
| -106 | ERROR_SERIAL_WRITE_FAIL | 데이터 송신 실패 |
| -107 | ERROR_SERIAL_MEM_ALLOC | 수신 데이터 메모리 할당 실패 |
| -108 | ERROR_SERIAL_TIMEOUT | 수신 데이터 타임 아웃(Timeout) |

표 2-81 는 시리얼 통신 오류 반환값에 대해 간략히 설명한 표이며, 오류 발생에 대한 조치사항은 알람 매뉴얼을 참조하시기 바랍니다.

### 2.13.1. seri = serial : 시리얼 포트 선언

| | |
|---|---|
| ■ 기 능 | **시리얼 통신의 Port와 Baudrate, Data bit, Parity bit, Stop bit를 선언합니다.** |
| ■ 형 태 | seri=serial(Port, Baudrate, Data bit, Parity bit, Stop bit) |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Port | string | ttyS1 | - | - | - |
| Baudrate | int | 115200 | 2400 | 115200 | bps |
| Data bit | int | 8 | 5 | 8 | bit |
| Parity bit | string | NONE | - | - | - |
| Stop bit | int | 1 | 1 | 2 | bit |

* 단위 : bps (bits per second)

■ **세부사항**

➢ **Port (COM 포트 번호)**

- 시리얼 통신 Port 번호를 입력합니다.
- 사용자가 설정 가능한 Port는 ttyS1, ttyS2, ttyS3 입니다.

ⓘ **ttyS0는 Safety board와 연결되어 있어, 설정이 불가합니다.**

➢ **Baudrate (통신 속도)**

- Baudrate는 2400, 4800, 9600, 19200, 38400, 57600, 115200가 제공됩니다.
- 지정된 Baudrate 이외의 값 입력 시 115200으로 설정됩니다.

➢ **Data bit (전송할 비트 수)**

- Data bit는 5~8비트로 설정 가능합니다.

➢ **Parity bit (오류 검사 비트)**

- Parity bit는 Even Parity 또는 Odd Parity로 설정 가능합니다.
- 사용 가능한 인자는 "NONE", "ODD", "EVEN"이 제공됩니다.

➢ **Stop bit (종료 신호 비트)**

- Stop bit는 1 또는 2로 설정 가능합니다.

■ 예 제

**[예제1] seri=serial 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_OPEN=0 | -- 통신 포트 연결 상태 |
| 2 | | |
| 3 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 통신 포트 선언 |
| 4 | 1) | |
| 5 | | |
| 6 | repeat | -- 조건에 따라 반복 |
| 7 | seri_open = seri.open() | -- 통신 포트 연결 |
| 8 | until seri_open==SERIAL_OPEN | -- 시리얼 통신이 연결될 때까지 조건 |
| 9 | | |
| 10 | result, data=seri.read(5,3) | -- 데이터 수신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[5] of argument |
| | 인자 타입 | Invalid argument[5] type of number data |
| | 명령어 괄호 | ')' expected near ']' |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.2. seri.open : 시리얼 통신 연결

| | |
|---|---|
| ■ 기 능 | **시리얼 통신 포트를 연결합니다.** |
| ■ 형 태 | result = seri.open() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ 세부사항

➢ 시리얼 포트 연결 명령어로 인자가 필요하지 않습니다.

■ 반환 값

| 값 | 항목 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 시리얼 통신 연결 성공 |
| -103 | ERROR_SERIAL_PORT_OPENED | 시리얼 통신이 연결된 상태 |
| -104 | ERROR_SERIAL_OPEN_FAIL | 시리얼 통신 연결 실패 |

■ 예 제

**[예제1] seri.open 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_OPEN=0 | -- 통신 포트 연결 상태 |
| 2 | | |
| 3 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 통신 포트 선언 |
| 4 | 1) | |
| 5 | | |
| 6 | repeat | -- 조건에 따라 반복 |
| 7 | seri_open = seri.open() | -- 통신 포트 연결 |
| 8 | until seri_open==SERIAL_OPEN | -- 시리얼 통신이 연결될 때까지 조건 |
| 9 | | |
| 10 | result, data=seri.read(5,3) | -- 데이터 수신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | seri=serial() |
| | seri.open() |
| | seri.close() |
| | seri.state() |
| | seri.read() |
| | seri.write() |
| | seri.set_byte_timeout() |
| | seri.set_label() |
| | seri.count_rx() |
| | seri.count_tx() |
| | seri.flush_rx() |
| | seri.flush_tx() |
| | seri.flush() |

## 2.13.3. seri.close : 시리얼 통신 종료

| | |
|---|---|
| ■ 기 능 | **시리얼 통신 포트를 종료합니다.** |
| ■ 형 태 | result = seri.close() |

■ 인자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ 세부사항

➢ 시리얼 통신 연결 종료 명령어로 인자가 필요하지 않습니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 시리얼 통신 종료 성공 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신이 연결되지 않은 상태 |

■ 예 제

**[예제1] seri.close 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_OPEN=0 | -- 통신 포트 연결 상태 |
| 2 | | |
| 3 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 통신 포트 선언 |
| 4 | 1) | |
| 5 | | |
| 6 | repeat | -- 조건에 따라 반복 |
| 7 | seri_open = seri.open() | -- 통신 포트 연결 |
| 8 | until seri_open==SERIAL_OPEN | -- 시리얼 통신이 연결될 때까지 조건 |
| 9 | | |
| 10 | result, data=seri.read(5,3) | -- 데이터 수신 |
| 11 | | |
| 12 | seri.close() | -- 통신 포트 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | seri=serial() |
| | seri.open() |
| | seri.close() |
| | seri.state() |
| | seri.read() |
| | seri.write() |
| | seri.set_byte_timeout() |
| | seri.set_label() |
| | seri.count_rx() |
| | seri.count_tx() |
| | seri.flush_rx() |
| | seri.flush_tx() |
| | seri.flush() |

## 2.13.4. seri.state : 시리얼 통신 포트 상태 확인

| | |
|---|---|
| ■ 기 능 | **시리얼 통신 포트의 상태를 반환합니다.** |
| ■ 형 태 | result = seri.state() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

➢ **시리얼 통신 포트의 상태 확인 명령어로 인자가 필요하지 않습니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 1 | SERIAL_PORT_OPENED | 시리얼 통신 연결 상태 |
| 0 | SERIAL_PORT_CLOSED | 시리얼 통신 미연결 상태 |

■ 예 제

**[예제1] seri.state 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_OPEN=0 | -- 통신 포트 연결 |
| 2 | SERIAL_CONNECT=-103 | -- 통신 포트 연결된 상태 |
| 3 | | |
| 4 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 통신 포트 선언 |
| 5 | 1) | |
| 6 | seri.open() | -- 통신 포트 연결 |
| 7 | | |
| 8 | state = seri.state() | -- 통신 상태 확인 |
| 9 | while state == SERIAL_CONNECT do | -- 통신 연결 상태인 경우 반복 |
| 10 | state = seri.state() | -- 통신 상태 확인 |
| 11 | result, data=seri.read(5,3) | -- 데이터 수신 |
| 12 | end | |
| 13 | | |
| 14 | seri.close() | -- 통신 포트 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.5. seri.read : 데이터 수신

| | |
|---|---|
| ■ 기 능 | **시리얼 포트에서 데이터를 수신합니다.** |
| ■ 형 태 | result, data = seri.read(Length, Timeout) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Length | int | - | 1 | 256 | byte |
| Timeout | int | - | 0 | 600 | sec |

* 단위 : 1byte = 알파벳 1글자 (최대 256 글자), sec (second)

■ **세부사항**

➢ **Length (수신 데이터 개수)**

- 수신할 데이터의 데이터 개수를 지정합니다.

➢ **Timeout (수신 데이터 시간)**

- 데이터를 수신받는 시간 설정합니다.
- 설정한 시간 내 데이터가 수신되지 않은 경우, 다음 스텝의 명령을 수행합니다.
- 타임아웃(Timeout)을 0으로 설정 시, 통신 주기가 빠른 경우 데이터 일부만 수신될 수 있습니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| data | string | 수신 데이터 문자(열) |

| 값 | 자료형 | 설명 |
|---|---|---|
| result | int | 데이터 수신 결과 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신 연결 실패 |
| -107 | ERROR_SERIAL_MEM_ALLOC | 수신할 데이터 메모리 동적할당 실패 |
| -108 | ERROR_SERIAL_TIMEOUT | 데이터 수신 시간 초과 |

■ 예 제

**[예제1] seri.read 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_OPEN=0 | -- 통신 포트 연결 |
| 2 | SERIAL_CONNECT=-103 | -- 통신 포트 연결된 상태 |
| 3 | SERIAL_TIMEOUT=-108 | -- 데이터 수신 시간 초과 |
| 4 | | |
| 5 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 통신 포트 선언 |
| 6 | 1) | |
| 7 | seri.open() | -- 통신 포트 연결 |
| 8 | state = seri.state() | -- 통신 상태 확인 |
| 9 | while state == SERIAL_CONNECT do | -- 시리얼 통신이 연결된 경우 반복 |
| 10 | state = seri.state() | -- 통신 상태 확인 |
| 11 | result, data=seri.read(5,3) | -- 데이터 수신 |
| 12 | if result == SERIAL_TIMEOUT then | -- 데이터 수신 시간이 초과한 경우 |
| 13 | msg.error("SERIAL_TIMEOUT") | -- 에러메시지 출력 |
| 14 | end | |
| 15 | end | |
| 16 | | |
| 17 | seri.close() | -- 통신 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[1000] is range over(1~256) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.6. seri.write : 데이터 송신

| | |
|---|---|
| ■ 기 능 | **시리얼 통신으로 데이터를 송신합니다.** |
| ■ 형 태 | result = seri.write(Data, Length) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | string | - | 1 | 256 | byte |
| Length | int | - | 1 | 256 | - |

* 단위 : 1byte = 알파벳 1글자 (최대 1024 글자)

■ **세부사항**

➢ **Data (송신 데이터)**

- 송신할 문자열 데이터를 작성합니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

➢ **Length (송신 데이터 개수)**

- 송신할 문자열 데이터의 길이를 작성합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 데이터 송신 성공 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신 연결 실패 |

■ 예 제

**[예제1] seri.write 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_CONNECT=1 | -- 통신 연결 상태 |
| 2 | SERIAL_COMM_FAIL=-105 | -- 통신 연결 실패 |
| 3 | | |
| 4 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 시리얼 통신 포트 선언 |
| 5 | 1) | |
| 6 | state = seri.open() | -- 시리얼 통신 연결 |
| 7 | if state ==SERIAL_CONNECT then | -- 시리얼 통신 연결된 경우, |
| 8 | data_msg="send test" | -- 전송할 문자 데이터 선언 및 초기화 |
| 9 | seri.write(data_msg, 9) | -- 데이터 송신 |
| 10 | | |
| 11 | if seri.write==SERIAL_COMM_FAIL | -- 반환 값이 -105인 경우 |
| 12 | then | |
| 13 | seri.close() | -- 시리얼 통신 연결 종료 |
| 14 | seri.open() | -- 시리얼 통신 연결 |
| 15 | end | |
| 16 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of number data |
| | 명령어 괄호 | unexpected symbol near ')' |
| 실행오류 | 데이터 유효범위 | Value[1000] is range over(1~256) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.7. seri.set_byte_timeout : 수신 데이터 문자간 타임아웃 설정

| | |
|---|---|
| ■ 기 능 | **수신 데이터 문자간 타임아웃(Timeout)을 설정합니다.** |
| ■ 형 태 | result = seri.set_byte_timeout(Time) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Time | int | - | 0 | 600 | sec |

■ **세부사항**

➢ **Time (수신 시간)**

- 데이터 문자간 수신 받는 시간을 설정합니다.
- 타임아웃 발생 이전까지 데이터를 연속적인 데이터로 처리합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 타임아웃 설정 성공 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신 연결 실패 |

■ 예 제

**[예제1] seri.set_byte_timeout 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SUCCESS_SERIAL=0 | -- 통신 연결 상태 |
| 2 | SERIAL_PORT_CLOSED =-105 | -- 통신 연결 실패 |
| 3 | | |
| 4 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 시리얼 통신 포트 선언 |
| 5 | 1) | |
| 6 | seri.open() | -- 시리얼 통신 연결 |
| 7 | | |
| 8 | state = seri.set_byte_timeout(2) | -- 바이트 간의 타임아웃 설정 |
| 9 | if state == SERIAL_PORT_CLOSED | -- 시리얼 통신 연결이 실패 한 경우 |
| 10 | then | |
| 12 | msg.error("ERROR PORT CLOSED") | -- 에러 메시지 출력 |
| 13 | elseif state == SUCCESS_SERIAL then | -- 타임아웃 설정 성공인 경우 |
| 14 | result, data = seri.read(5,3) | -- 데이터 수신 |
| 15 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[1000] is range over(1~600) |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.8. seri.set_label : 송신 데이터 접미사 설정

| | |
|---|---|
| ■ 기 능 | **송신 데이터 끝에 접미사를 설정합니다.** |
| ■ 형 태 | result = seri.set_label(Data) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Data | string | - | 0 | 255 | byte |

* 단위 : 1byte = 알파벳 1글자 (최대 255 글자)

■ **세부사항**

➢ **Data (추가 송신 데이터)**

- 송신할 문자열 데이터의 접미사를 설정합니다.
- 데이터 줄 바꿈 또는 단위 설정과 같이 송신 데이터의 접미사로 활용됩니다.
- 표 2-82와 같이 설정된 추가 송신 데이터는 해제 전까지 송신 데이터 끝에 적용됩니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

| 항목 | set_label | |
|---|---|---|
| | 미사용 | 사용 |
| 송신 명령어 (write) | "send" | "send" |
| 접미사 설정 명령어 (set_lable) | - | "add" |
| 실제 송신 데이터 | "send" | "sendadd" |

**표 2-82 seri.set_label 적용 예시**

※ 접미사 해제 시 인자에 ("")를 입력하시기 바랍니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 데이터 송신 성공 |

■ 예 제

**[예제1] seri.set_label 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_OPEN=0 | -- 통신 포트 연결 |
| 2 | SERIAL_CONNECT=-103 | -- 통신 포트 연결된 상태 |
| 3 | | |
| 4 | seri = serial("ttyS1", 115200, 8, | -- 통신 포트 선언 |
| 5 | "NONE", 1) | |
| 6 | | -- 조건에 따라 반복 |
| 7 | repeat | -- 통신 포트 연결 |
| 8 | seri_open = seri.open() | -- 시리얼 통신이 연결될 때까지 조건 |
| 9 | until seri_open == SERIAL_OPEN | |
| 10 | | -- 통신 상태 확인 |
| 11 | state = seri.state() | -- 통신이 연결된 상태인 경우 |
| 12 | while state == SERIAL_CONNECT do | -- 송신 데이터 접미사 설정(₩r₩n) |
| 13 | seri.set_label("₩r₩n") | -- 송신 데이터 할당 |
| 14 | send_msg = "send_test" | -- 데이터 송신 |
| 15 | seri.write(send_msg,9) | ※ 실제 송신 데이터 : |
| 16 | | "send_test₩r₩n" |
| 17 | seri.set_label("") | -- 송신 데이터 접미사 초기화 |
| 18 | end_msg = "test_end" | -- 송신 데이터 할당 |
| 19 | seri.write(end_msg,8) | -- 데이터 송신 |
| 20 | end | ※ 실제 송신 데이터 : "end" |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[1000] is range over(0~255) |

| | |
|---|---|
| ■ 관련 함수 | seri=serial() |
| | seri.open() |
| | seri.close() |
| | seri.state() |
| | seri.read() |
| | seri.write() |
| | seri.set_byte_timeout() |
| | seri.set_label() |
| | seri.count_rx() |
| | seri.count_tx() |
| | seri.flush_rx() |
| | seri.flush_tx() |
| | seri.flush() |

## 2.13.9. seri.count_rx : 수신 버퍼 데이터 개수 반환

| | |
|---|---|
| ■ 기 능 | **시리얼 포트의 수신 버퍼에 남아있는 데이터 개수를 반환합니다.** |
| ■ 형 태 | recv_count = seri.count_rx() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

- ➢ **인자가 필요하지 않습니다.**
- ➢ **수신 버퍼에 남은 데이터 개수를 확인하는 용도로 사용됩니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| recv_count | int | 수신 버퍼 데이터 개수 |

■ 예 제

**[예제1] seri.count_rx 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_CONNECT=1 | -- 통신 연결 상태 |
| 2 | | |
| 3 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 시리얼 통신 포트 선언 |
| 4 | 1) | -- 시리얼 통신 연결 |
| 5 | open_status=seri.open() | |
| 6 | | -- 시리얼 통신 연결된 경우, |
| 7 | if open_status==SERIAL_CONNECT | |
| 8 | then | |
| 9 | while 1 do | -- 데이터 수신 반복문 |
| 10 | rx_count = seri.count_rx() | -- 수신 버퍼 데이터 바이트 수 리턴 |
| 11 | if rx_count >= 5 then | -- 수신 데이터 개수가 5개인 경우 |
| 12 | result, data = seri.read(5,2) | -- 데이터 수신 |
| 13 | break | -- 데이터 수신 반복문 탈출 |
| 14 | end | |
| 15 | end | |
| 16 | end | |
| 17 | seri.close() | -- 시리얼 통신 연결 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.10. seri.count_tx : 송신 버퍼 데이터 개수 반환

| | |
|---|---|
| ■ 기 능 | **시리얼 포트의 송신 버퍼에 남아있는 데이터 개수를 반환합니다.** |
| ■ 형 태 | send_count = seri.count_tx() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ 세부사항

- ➢ 인자가 필요하지 않습니다.
- ➢ 송신 버퍼에 남은 데이터 개수를 확인하는 용도로 사용됩니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| send_count | int | 송신 버퍼 데이터 개수 |

■ 예 제

**[예제1] seri.count_tx 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | SERIAL_CONNECT=1 | -- 통신 연결 상태 |
| 2 | | |
| 3 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 시리얼 통신 포트 선언 |
| 4 | 1) | |
| 5 | open_status = seri.open() | -- 시리얼 통신 연결 |
| 6 | | |
| 7 | if open_status ==SERIAL_CONNECT | -- 시리얼 통신 연결된 경우, |
| 8 | then | |
| 9 | while 1 do | -- 데이터 송신 반복문 |
| 10 | tx_count = seri.count_tx() | -- 송신 버퍼 데이터 바이트 수 리턴 |
| 11 | if tx_count == 0 then | -- 송신 데이터 개수가 없는 경우 |
| 12 | seri.write("send_msg",8) | -- 데이터 송신 |
| 13 | break | -- 데이터 송신 반복문 탈출 |
| 14 | end | |
| 15 | end | |
| 16 | end | |
| 17 | seri.close() | -- 시리얼 통신 연결 종료 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.11. seri.flush_rx : 수신 버퍼 초기화

| | |
|---|---|
| ■ 기 능 | **시리얼 포트의 수신 버퍼를 초기화합니다.** |
| ■ 형 태 | result = seri.flush_rx() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ 세부사항

- 수신 버퍼 초기화 명령어로 인자가 필요하지 않습니다.
- 통신 개시 또는 통신 비정상 종료 시 수신 버퍼의 남아있는 데이터를 초기화합니다.

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 데이터 초기화 성공 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신 닫혀 있음 |

■ 예 제

**[예제1] seri.flush_rx 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 시리얼 통신 포트 선언 |
| 2 | 1) | |
| 3 | seri.open() | -- 통신 연결 |
| 4 | | |
| 5 | seri.flush_rx() | -- 수신 버퍼의 데이터 초기화 |
| 6 | result, data=seri.read(1,3) | -- 데이터 수신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.13.12. seri.flush_tx : 송신 버퍼 초기화

| | |
|---|---|
| ■ 기 능 | **시리얼 포트의 송신 버퍼를 초기화합니다.** |
| ■ 형 태 | result = seri.flush_tx() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

- **송신 버퍼 초기화 명령어로 인자가 필요하지 않습니다.**
- **통신 개시 또는 통신 비정상 종료 시 송신 버퍼의 남아있는 데이터를 초기화합니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 데이터 초기화 성공 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신 닫혀 있음 |

■ 예 제

**[예제1] seri.flush_tx 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | seri=serial(*"ttyS1"*, 115200, 8, *"NONE"*, | -- 시리얼 통신 포트 선언 |
| 2 | 1) | |
| 3 | seri.open() | -- 통신 연결 |
| 4 | | |
| 5 | seri.flush_tx() | -- 송신 버퍼의 데이터 초기화 |
| 6 | seri.write(*"send_msg"*, 8) | -- 데이터 송신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | seri=serial() |
| | seri.open() |
| | seri.close() |
| | seri.state() |
| | seri.read() |
| | seri.write() |
| | seri.set_byte_timeout() |
| | seri.set_label() |
| | seri.count_rx() |
| | seri.count_tx() |
| | seri.flush_rx() |
| | seri.flush_tx() |
| | seri.flush() |

## 2.13.13. seri.flush : 송/수신 버퍼 초기화

| | |
|---|---|
| ■ 기 능 | **시리얼 포트의 송/수신 버퍼를 초기화합니다.** |
| ■ 형 태 | result = seri.flush() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| - | - | - | - | - | - |

■ **세부사항**

- **송/수신 버퍼 초기화 명령어로 인자가 필요하지 않습니다.**
- **통신 개시 또는 통신 비정상 종료 시 송/수신 버퍼의 남아있는 데이터를 초기화합니다.**

■ 반환 값

| 값 | 자료형 | 설명 |
|---|---|---|
| 0 | SUCCESS_SERIAL | 데이터 초기화 성공 |
| -105 | ERROR_SERIAL_PORT_CLOSED | 시리얼 통신 닫혀 있음 |

■ 예 제

**[예제1] seri.flush 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | seri=serial("ttyS1", 115200, 8, "NONE", | -- 시리얼 통신 포트 선언 |
| 2 | 1) | |
| 3 | seri.open() | -- 통신 연결 |
| 4 | | |
| 5 | seri.flush() | -- 송/수신 버퍼의 데이터 초기화 |
| 6 | result, data=seri.read(1,3) | -- 데이터 수신 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[0] of argument |
| | 명령어 괄호 | ')' expected near ']' |

| | |
|---|---|
| ■ 관련 함수 | |

| | | | | |
|---|---|---|---|---|
| seri=serial() | | | | |
| seri.open() | | | | |
| seri.close() | | | | |
| seri.state() | | | | |
| seri.read() | | | | |
| seri.write() | | | | |
| seri.set_byte_timeout() | | | | |
| seri.set_label() | | | | |
| seri.count_rx() | | | | |
| seri.count_tx() | | | | |
| seri.flush_rx() | | | | |
| seri.flush_tx() | | | | |
| seri.flush() | | | | |

## 2.14. 전역변수 스크립트

전역변수는 위치형(Pose), 정수형(Integer), 실수형(Float), 논리형(Boolean), 문자열(String)와 같이 총 5가지 데이터 타입을 제공하고 있습니다.

모든 프로그램에서는 전역변수 스크립트 사용으로 공통의 데이터 영역에 읽고 쓰기를 수행할 수 있습니다.

**표 2-83 전역변수 스크립트 항목**

| No | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 위치형 | var.p(Index, Mode) | 위치형 데이터 읽기 | 2.14.1 |
| 2 | | var.p(Index, Mode, Value) | 위치형 데이터 쓰기 | 2.14.2 |
| 3 | 정수형 | var.i(Index) | 정수형 데이터 읽기 | 2.14.3 |
| 4 | | var.i(Index, Value) | 정수형 데이터 쓰기 | 2.14.4 |
| 5 | 실수형 | var.f(Index) | 실수형 데이터 읽기 | 2.14.5 |
| 6 | | var.f(Index, Value) | 실수형 데이터 쓰기 | 2.14.6 |
| 7 | 논리형 | var.b(Index) | 논리형 데이터 읽기 | 2.14.7 |
| 8 | | var.b(Index, Value) | 논리형 데이터 쓰기 | 2.14.8 |
| 9 | 문자열 | var.s(Index) | 문자열 데이터 읽기 | 2.14.9 |
| 10 | | var.s(Index, Value) | 실수형 데이터 쓰기 | 2.14.10 |

표 2-83은 전역변수 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.14.1. var.p : 위치 값 읽기

| | |
|---|---|
| ■ 기 능 | **위치형 전역변수 읽기를 수행합니다.** |
| ■ 형 태 | var.p(Index, [Mode]) |

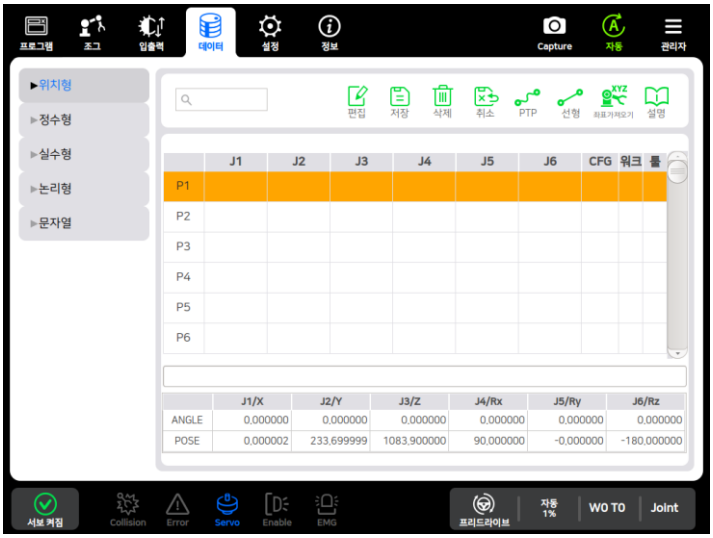
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |
| [Mode] | string | "joint" | "joint", "xyz", "tool" | | - |

■ **세부사항**

➢ **Index (데이터 번호)**

- [데이터]-[위치형] 탭의 데이터 번호입니다.




➢ **[Mode] (좌표 모드)**

- 해당 인자는 옵션 인자입니다.
- 좌표 모드 설정이 없는 경우, "Joint" 좌표값으로 읽어 옵니다.
- 좌표 모드 설정은 "joint", "xyz", "tool" 좌표 모드를 지원합니다.

| 축 | 각 축의 데이터 단위 | | |
|---|---|---|---|
| | "joint" | "xyz" | "tool" |
| 1 | [deg] | [mm] | [mm] |
| 2 | [deg] | [mm] | [mm] |
| 3 | [deg] | [mm] | [mm] |
| 4 | [deg] | [deg] | [deg] |
| 5 | [deg] | [deg] | [deg] |
| 6 | [deg] | [deg] | [deg] |

**표 2-84 좌표 모드 변경에 따른 데이터 단위**

**표 2-84와 같이 좌표 모드 변경에 따라 데이터 단위가 다르니 유의하시기 바랍니다.**

### ➢ 관절 데이터 확인

- 그림 2-99과 같이 좌표 모드 설정이 Joint 인 경우, 표 2-85 의 명령어를 사용하여 개별 또는 전체 **관절 데이터** 위치 값을 확인할 수 있습니다.

**그림 2-99 관절 데이터 화면**

| 명령어 | 설명 |
|---|---|
| var.p(Index, "joint") | 전체 관절 위치 값 |
| var.p(Index).j1 | 관절 1 번축 위치 값 |
| var.p(Index).j2 | 관절 2 번축 위치 값 |
| var.p(Index).j3 | 관절 3 번축 위치 값 |
| var.p(Index).j4 | 관절 4 번축 위치 값 |
| var.p(Index).j5 | 관절 5 번축 위치 값 |
| var.p(Index).j6 | 관절 6 번축 위치 값 |

**표 2-85 관절 데이터 읽기**

### ➢ 직교 데이터 확인

- 그림 2-100과 같이 좌표 모드 설정이 X-Y-Z 또는 Tool 인 경우, 표 2-86의 명령어를 사용하여 개별 또는 전체 **직교 데이터** 위치 값을 확인할 수 있습니다.

**그림 2-100 직교 데이터 화면**

| 명령어 | 설명 |
|---|---|
| var.p(Index, "xyz") | 직교 좌표계 위치 값 |
| var.p(Index, "xyz").x | 직교 좌표계 x 위치 값 |
| var.p(Index, "xyz").y | 직교 좌표계 y 위치 값 |
| var.p(Index, "xyz").z | 직교 좌표계 z 위치 값 |
| var.p(Index, "xyz").rx | 직교 좌표계 rx 위치 값 |
| var.p(Index, "xyz").ry | 직교 좌표계 ry 위치 값 |
| var.p(Index, "xyz").rz | 직교 좌표계 rz 위치 값 |

**표 2-86 직교 데이터 읽기**

■ 반환 값

| 인자 | | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|---|
| 좌표계 | 축 | | | 최소 | 최대 | |
| Joint | J1 | double | - | -360.0 | 360.0 | deg |
| | J2 | double | - | -360.0 | 360.0 | deg |
| | J3 | double | - | -150.0 | 150.0 | deg |
| | J4 | double | - | -360.0 | 360.0 | deg |
| | J5 | double | - | -360.0 | 360.0 | deg |
| | J6 | double | - | -360.0 | 360.0 | deg |
| XYZ | X | double | - | -1700.0 | 1700.0 | mm |
| | Y | double | - | -1700.0 | 1700.0 | mm |
| | Z | double | - | -1700.0 | 1700.0 | mm |
| | Rx | double | - | -180.0 | 180.0 | deg |
| | Ry | double | - | -180.0 | 180.0 | deg |
| | Rz | double | - | -180.0 | 180.0 | deg |
| Tool | X | double | - | -1700.0 | 1700.0 | mm |
| | Y | double | - | -1700.0 | 1700.0 | mm |
| | Z | double | - | -1700.0 | 1700.0 | mm |
| | Rx | double | - | -180.0 | 180.0 | deg |
| | Ry | double | - | -180.0 | 180.0 | deg |
| | Rz | double | - | -180.0 | 180.0 | deg |

■ 예 제

**[예제1] 위치형 데이터 읽어오기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | p1 = var.p(1, *"joint"*) | -- P1번 위치형 데이터 읽기 (관절 좌표계) |
| 2 | movep(p1) | |
| 3 | | |
| 4 | p2 = var.p(2, *"xyz"*) | -- P2번 위치형 데이터 읽기 (직교 좌표계) |
| 5 | movel(p2) | |
| 6 | | |
| 7 | p3 = var.p(3, *"tool"*) | -- P3번 위치형 데이터 읽기 (툴 좌표계) |
| 8 | movel(p3) | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[2] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

| | |
|---|---|
| ■ 관련 함수 | var.p() |
| | var.i() |
| | var.f() |
| | var.b() |
| | var.s() |

## 2.14.2. var.p : 위치 값 쓰기

| | |
|---|---|
| ■ 기 능 | **위치형 전역변수 쓰기를 수행합니다.** |
| ■ 형 태 | var.p(Index, Mode, Value) |

■ 인 자

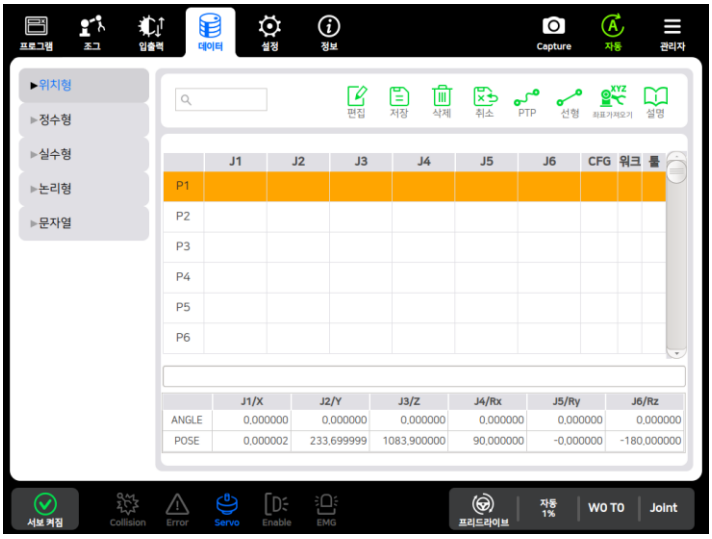
| 인자 | | | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|---|---|
| 좌표계 | | 축 | | | 최소 | 최대 | |
| Index | | | int | - | 1 | 2000 | - |
| Mode | | | string | "joint" | "joint", "xyz", "tool" | | - |
| Value | Joint | J1 | double | - | -360.0 | 360.0 | deg |
| | | J2 | double | - | -360.0 | 360.0 | deg |
| | | J3 | double | - | -150.0 | 150.0 | deg |
| | | J4 | double | - | -360.0 | 360.0 | deg |
| | | J5 | double | - | -360.0 | 360.0 | deg |
| | | J6 | double | - | -360.0 | 360.0 | deg |
| | Pose | X | double | - | -1700.0 | 1700.0 | mm |
| | | Y | double | - | -1700.0 | 1700.0. | mm |
| | | Z | double | - | -1700.0 | 1700.0 | mm |
| | | Rx | double | - | -180.0 | 180.0 | deg |
| | | Ry | double | - | -180.0 | 180.0 | deg |
| | | Rz | double | - | -180.0 | 180.0 | deg |

* 단위 : deg (degree) / mm (millimeter)

■ **세부사항**

➢ **Index (데이터 번호)**

- [데이터]-[위치형] 탭의 데이터 번호 입니다.




➢ **Mode (좌표 모드)**

- 좌표 모드는 **"joint"**, **"xyx"**, **"tool"** 모드를 지원합니다.

➢ **Value (위치형 데이터)**

- 위치형 데이터 입력 방식은 표 2-87의 위치형 함수를 이용한 방식과 사용자 변수를 이용한 방식을 제공합니다.

| 입력방식 | 좌표계 | 예 제 |
|---|---|---|
| 위치형 함수 | joint | var.p(1, *"joint"*, joint(30,30,60,20,0,0)) |
| | pose | var.p(1, *"xyz"*, pose(700,150,800,0,0,30)) |
| 사용자 변수 | joint | local_point = joint(30,30,60,20,0,0)<br>var.p(1, *"joint"*, local_point) |
| | pose | local_point = pose(700,150,800,0,0,30)<br>var.p(1, *"xyz"*, local_point) |

**표 2-87 위치형 함수 입력 방식**

■ **예 제**

**[예제1] 위치형 데이터 쓰기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | p = var.p(1, *"joint"*, joint(0,0,90,0,90,0)) | -- P1번 위치형 데이터에 쓰기 (관절 좌표계) |
| 2 | movep(p) | |
| 3 | | |
| 4 | p1 = pose(530,50,600,180,0,180) | -- 사용자 변수 정의 |
| 5 | p2 = var.p(2, *"xyz"*, p1) | -- P2번 위치형 데이터에 쓰기 (직교 좌표계) |
| 6 | movel(p2) | |
| 7 | | |
| 8 | p3 = var.p(3, *"tool"*, p1) | -- P3번 위치형 데이터에 쓰기 (툴 좌표계) |
| 9 | movel(p3) | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[3] of argument |
| | 인자 타입 | Invalid argument[2] type of string data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.3. var.i : 정수 값 읽기

| | |
|---|---|
| ■ 기 능 | **정수형 전역변수 읽기를 수행합니다.** |
| ■ 형 태 | var.i(Index) |

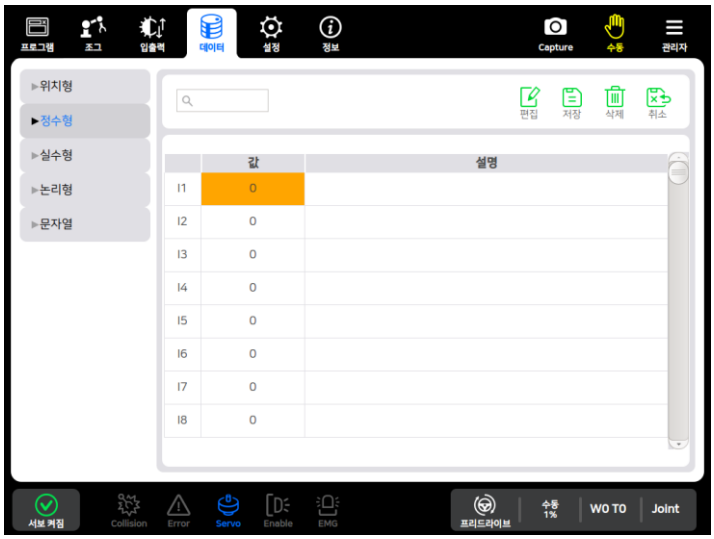
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |

■ **세부사항**

➢ **Index (정수형 테이블 번호)**

- [데이터]-[정수형] 탭의 데이터 번호입니다.




■ 반환 값

| 구분 | 반환값 | 설명 |
|---|---|---|
| 읽기 | int | 해당 번호(Index)의 정수 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] 정수형 데이터 읽기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | if (var.i(1) == 1) then | -- I1번 정수형 데이터가 1인 경우 |
| 2 | dio.set(1) | -- 디지털 출력 1번 신호 ON |
| 3 | else | -- 1이 아닌 경우 |
| 4 | dio.reset(1) | -- 디지털 출력 1번 신호 OFF |
| 5 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.4. var.i : 정수 값 쓰기

| | |
|---|---|
| ■ 기 능 | **정수형 전역변수 쓰기를 수행합니다.** |
| ■ 형 태 | var.i(Index, Value) |

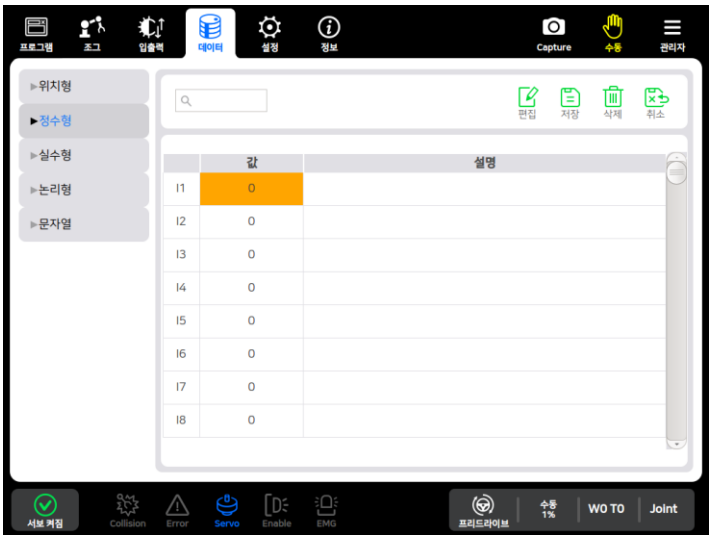
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |
| Value | int | - | -2147483648 | 2147483647 | - |

■ **세부사항**

➢ **Index (정수형 테이블 번호)**

- [데이터]-[정수형] 탭의 데이터 번호입니다.




➢ **Value (정수형 데이터)**

- 정수형 데이터 입력 방식은 표 2-88과 같이 2가지 입력 방식을 제공합니다..

| 입력방식 | 예 제 | 설명 |
|---|---|---|
| 상수 | var.i(1, 100) | 정수형 데이터 I1번에 상수 100 쓰기 |
| 변수 | int_data = 100<br>var.i(1, int_data) | 정수형 데이터 I1번에 변수 100 쓰기 |

**표 2-88 정수형 데이터 입력 방식**

■ 예 제

**[예제1] 정수형 데이터 쓰기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | var.i(100, 123) | -- I100번 정수형 데이터에 123 쓰기 |
| 2 | var.i(200, 456) | -- I200번 정수형 데이터에 456 쓰기 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.5. var.f : 실수 값 읽기

| | |
|---|---|
| ■ 기 능 | **실수형 전역변수 읽기를 수행합니다.** |
| ■ 형 태 | var.f(Index) |

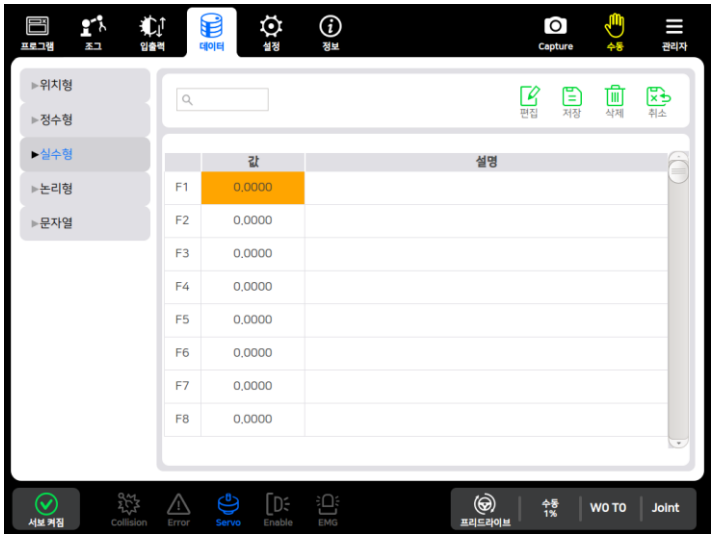
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |

■ **세부사항**

➢ **Index (실수형 테이블 번호)**

- [데이터]-[실수형] 탭의 데이터 번호입니다.




■ **반환 값**

| 구분 | 반환값 | 설명 |
|---|---|---|
| 읽기 | float | 해당 번호(Index)의 실수 값 |
| 실패 | - | 에러 메시지 참고 |

■ **예 제**

**[예제1] 실수형 데이터 읽기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | if (var.f(1) > 1.0) then | -- F1번 실수형 데이터가 1.0보다 큰 경우 |
| 2 | dio.set(1) | -- 디지털 출력 1번 신호 ON |
| 3 | else | -- 1.0보다 작은 경우 |
| 4 | dio.reset(1) | -- 디지털 출력 1번 신호 OFF |
| 5 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.6. var.f : 실수 값 쓰기

| | |
|---|---|
| ■ 기 능 | **실수형 전역변수 쓰기를 수행합니다.** |
| ■ 형 태 | var.f(Index, Value) |

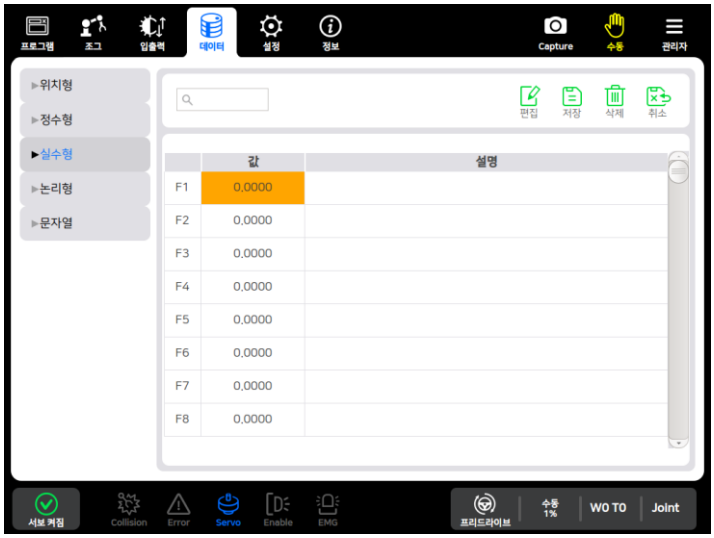
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |
| Value | float | - | -3.402823e+38 | 3.402823e+38 | |

■ **세부사항**

➢ **Index (실수형 테이블 번호)**

- [데이터]-[실수형] 탭의 데이터 번호입니다.




➢ **Value (실수형 데이터)**

- 실수형 데이터 입력 방식은 표 2-89과 같이 2가지 입력 방식을 제공합니다..

| 입력방식 | 예 제 | 설명 |
|---|---|---|
| 상수 | var.f(1, 1.234) | 실수형 데이터 F1번에 상수 1.234 쓰기 |
| 변수 | float_data = 1.234<br>var.f(1, float_data) | 실수형 데이터 F1번에 변수 1.234 쓰기 |

**표 2-89 실수형 데이터 입력 방식**

■ 예 제

**[예제1] 실수형 데이터 쓰기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | var.f(10, 1.23) | -- F10번 실수형 데이터에 1.23 쓰기 |
| 2 | var.f(20, 4.56) | -- F20번 실수형 데이터에 4.56 쓰기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.7. var.b : 논리형 읽기

| | |
|---|---|
| ■ 기 능 | **논리형 전역변수 읽기를 수행합니다.** |
| ■ 형 태 | var.b(Index) |

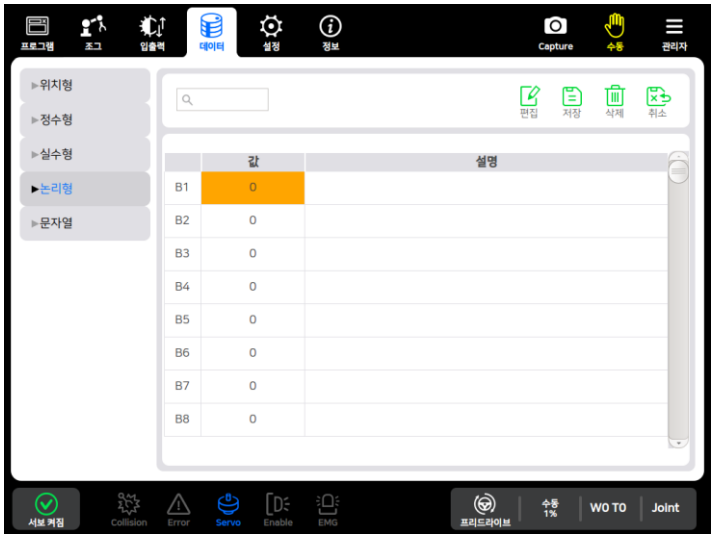
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |

■ **세부사항**

➢ **Index (논리형 테이블 번호)**

- [데이터]-[논리형] 탭의 데이터 번호입니다.




■ 반환 값

| 구분 | 반환값 | 설명 |
|---|---|---|
| 읽기 | boolean | 해당 번호(Index)의 논리 값 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] 논리형 데이터 읽기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | if (var.b(1) == true) then | -- B1번 논리형 데이터가 true인 경우 |
| 2 | dio.set(1) | -- 디지털 출력 1번 신호 ON |
| 3 | else | -- false인 경우 |
| 4 | dio.reset(1) | -- 디지털 출력 1번 신호 OFF |
| 5 | end | |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.8. var.b : 논리형 쓰기

| | |
|---|---|
| ■ 기 능 | **논리형 전역변수 쓰기를 수행합니다.** |
| ■ 형 태 | var.b(Index) |

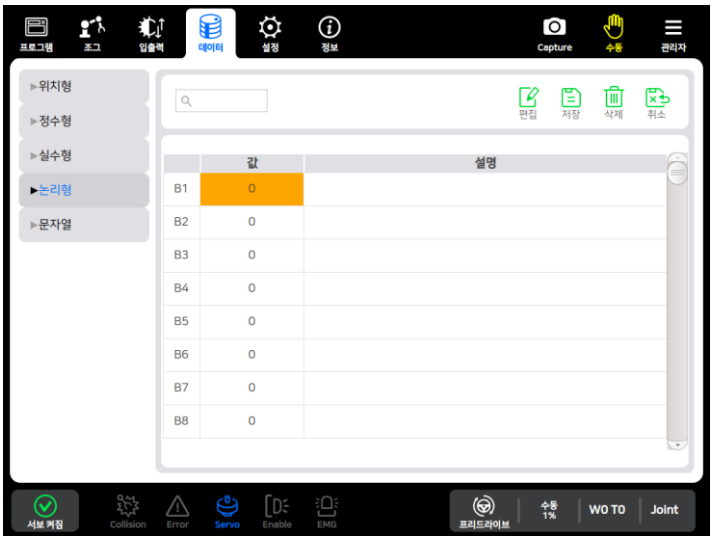
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |
| Value | boolean | false | true, false | | |

■ **세부사항**

➢ **Index (논리형 테이블 번호)**

- [데이터]-[논리형] 탭의 데이터 번호입니다.




➢ **Value (실수형 데이터)**

- 논리형 데이터 입력 방식은 표 2-90와 같이 2가지 입력 방식을 제공합니다...

| 입력방식 | 예 제 | 설명 |
|---|---|---|
| 상수 | var.b(1, true) | 논리형 데이터 B1번에 상수 true 쓰기 |
| 변수 | bool_data = true<br>var.b(1, bool_data) | 논리형 데이터 B1번에 변수 false 쓰기 |

**표 2-90 논리형 데이터 입력 방식**

■ 예 제

**[예제1] 논리형 데이터 쓰기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | var.b(2, false) | -- B100번 논리형 데이터에 false 쓰기 |
| 2 | var.b(2,true) | -- B200번 논리형 데이터에 true 쓰기 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.9. var.s : 문자열 읽기

| | |
|---|---|
| ■ 기 능 | **문자열 전역변수 읽기를 수행합니다.** |
| ■ 형 태 | var.s(Index) |

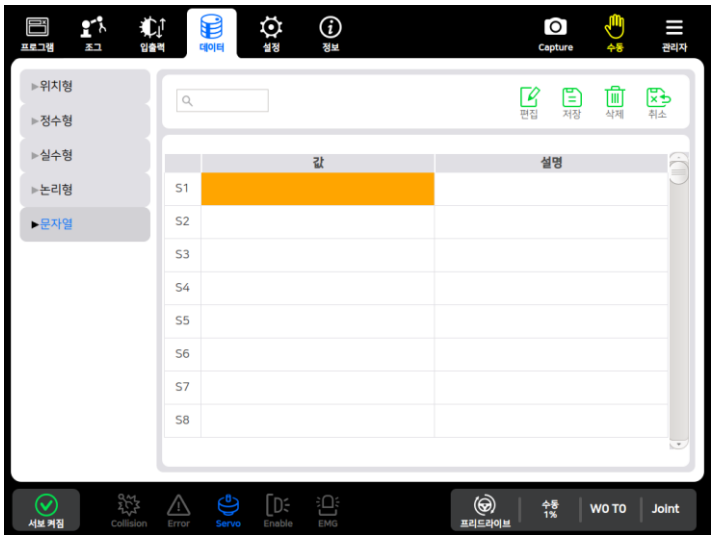
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |

■ **세부사항**

➢ **Index (문자열 테이블 번호)**

- [데이터]-[문자열] 탭의 데이터 번호입니다.




■ 반환 값

| 구분 | 반환값 | 설명 |
|---|---|---|
| 읽기 | string | 해당 번호(Index)의 문자열 |
| 실패 | - | 에러 메시지 참고 |

■ 예 제

**[예제1] 문자열 데이터 읽기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | msg.info(var.s(1)) | -- S1번 문자열 데이터 정보 메시지 출력 |
| 2 | | |
| 3 | str1 = var.s(2) | -- 문자열 데이터를 사용자 변수로 정의 |
| 4 | msg.error(str1) | -- 경고 메시지 출력 |

■ **에러 상황**

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ **관련 함수**

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.14.10. var.s : 문자열 쓰기

| | |
|---|---|
| ■ 기 능 | **문자열 전역변수 쓰기를 수행합니다.** |
| ■ 형 태 | var.s(Index, Value) |

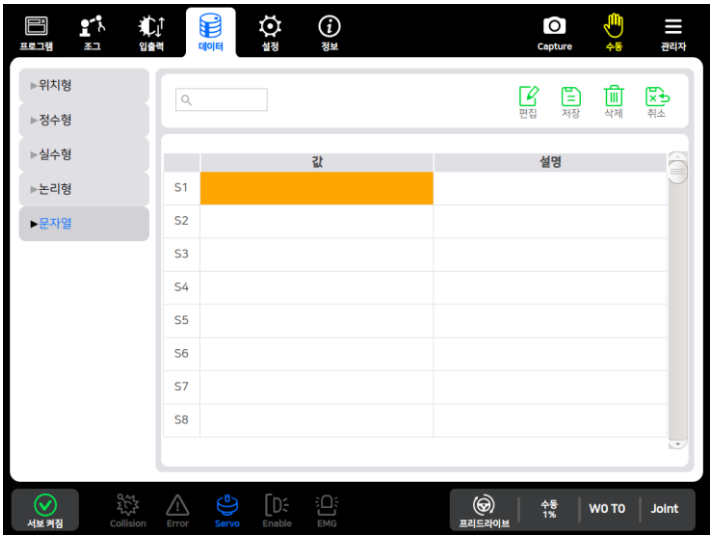
■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Index | int | - | 1 | 2000 | - |
| Value | string | - | 0 | 255 | |

■ **세부사항**

➢ **Index (문자열 테이블 번호)**

- [데이터]-[문자열] 탭의 데이터 번호입니다.
-




➢ **Value (문자열 데이터)**

- 문자열 데이터 입력 방식은 표 2-91과 같이 2가지 입력 방식을 제공합니다.

| 입력방식 | 예 제 | 설명 |
|---|---|---|
| 상수 | var.s(1, *"start"*) | 문자열 데이터 S1번에 상수 "start" 쓰기 |
| 변수 | str_data = *"start"*<br>var.s(1, str_data) | 문자열 데이터 S1번에 변수 "start" 쓰기 |

**표 2-91 문자열 데이터 입력 방식**

■ 예 제

**[예제1] 문자열 데이터 쓰기**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | var.s(1, *"Joint coordinate"*) | -- S1번 문자열 데이터 쓰기 |
| 2 | var.s(2, *"Value is range over"*) | -- S2번 문자열 데이터 쓰기 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[2] of argument |
| | 인자 타입 | Invalid argument[1] type of integer data |
| | 명령어 괄호 | ')' expected near ']' |
| 실행오류 | 데이터 유효범위 | Value[2001] is range over(1~2000) |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| var.p() | | | | |
| var.i() | | | | |
| var.f() | | | | |
| var.b() | | | | |
| var.s() | | | | |

## 2.15. 서브루틴(Subroutine) 스크립트

서브루틴 스크립트는 프로그램 내에서 다른 스크립트를 호출하여 실행할 때 사용됩니다.

메인 프로그램에서의 서브루틴 사용에 개수 제한은 없지만, 서브루틴 내에서 하위 서브루틴을 반복 호출 할 시, 최대 4 개로 제한 됩니다.

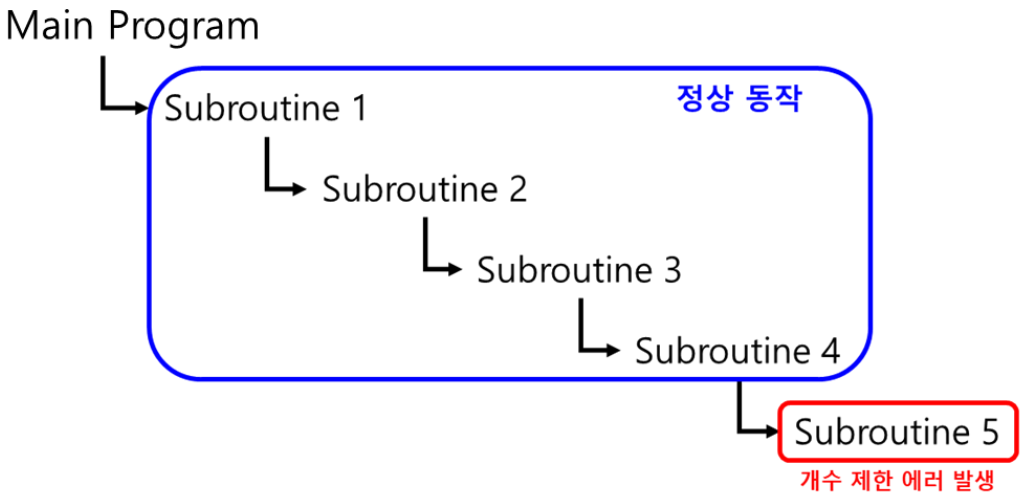



**표 2-92 서브루틴 스크립트 명령어 리스트**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 호출 | call(Program_name) | 해당 이름의 프로그램를 호출 합니다. | 2.15.1 |

표 2-92 는 서브루틴 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.15.1. call : 서브루틴 호출

| | |
|---|---|
| ■ 기 능 | **해당 이름의 프로그램을 호출합니다.** |
| ■ 형 태 | call(Program_name) |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| Script_name | string | - | 1 | 50 | byte |

* 단위 : 1byte = 알파벳 1글자 (최대 50 글자)

■ **세부사항**

➢ **Program_name (스크립트 이름)**

- 호출할 프로그램의 이름을 작성합니다.
- 문자열 데이터를 작성 시, **쌍따옴표 (" ")** 기호 안에 작성하시기 바랍니다.

■ 예 제

**[예제1] call 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | motion.speed=100 | |
| 2 | | |
| 3 | p1=joint(0,0,90,0,90,0) | |
| 4 | movep(p1) | -- 로봇 p1 위치로 이동 |
| 5 | | |
| 6 | io=dio.get(1) | -- 디지털 입력 1번 신호값 변수에 저장 |
| 7 | | |
| 8 | if io==1 then | -- 신호가 입력되었을 경우 |
| 9 | call("Sub1") | -- Sub1 이름의 프로그램 실행 |
| 10 | else | -- 입력도지 않았을 경우 |
| 11 | call("Sub2") | -- Sub2 이름의 프로그램 실행 |
| 12 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 인자 개수 | Invalid number[1] of argument |
| | 인자 타입 | Invalid argument[1] type of number data |
| | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |
| 실행오류 | 프로그램 이름 | File doesn't exist.. |
| | 하위 서브루틴 호출 초과 | too many call.. |
| | 호출 시간 경과 | Failed to call subroutine. |

| | | | | |
|---|---|---|---|---|
| ■ 관련 함수 | | | | |
| call() | | | - | - |
| | | | - | - |
| | - | | - | - |
| | - | - | - | - |
| | - | - | - | - |
| | - | - | - | - |
| | | | | |
| | | | | |

## 2.16. 타이머 스크립트

타이머 스크립트는 프로그램 내에서 타이머 시작지점과 종료지점을 지정하여 명령어의 수행시간을 측정할 때 사용됩니다.

**표 2-93 타이머 스크립트 명령어 리스트**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 생성 | timer=time() | 타이머를 생성합니다. | 2.16.1 |
| 2 | 시작 | timer.start() | 타이머를 시작합니다. | 2.16.2 |
| 3 | 정지 | timer.stop() | 타이머를 정지합니다. | 2.16.3 |
| 4 | 소요시간 계산 | timer.elapsed() | 시작시간과 정지시간의 차이값을 계산하여 반환합니다. | 2.16.4 |

표 2-93 은 타이머 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.16.1. timer = time : 타이머 생성

| | |
|---|---|
| ■ 기 능 | **타이머를 생성합니다.** |
| ■ 형 태 | timer = time() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| | | | | | |

■ **세부사항**

➢ **타이머를 생성하는 명령어로 인자가 필요하지 않습니다.**

■ 예 제

**[예제1] timer = time 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | motion.speed = 100 | -- 이동속도 설정 |
| 2 | | |
| 3 | timer = time() | -- 타이머 생성 |
| 4 | timer.start() | -- 타이머 시작 |
| 5 | | |
| 6 | movep(lp1) | -- lp1 위치로 로봇 이동 |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | movel(lp2) | -- lp2 위치로 로봇 이동 |
| 9 | delay(500) | -- 0.5초간 대기 |
| 10 | | |
| 11 | timer.stop() | -- 타이머 정지 |
| 12 | result = timer.elapsed() | -- 타이머 시작과 정지 사이의 시간 측정 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| timer=time() | | | - | - |
| timer.start() | | | - | - |
| timer.stop() | - | | - | - |
| timer.elapsed() | - | - | - | - |
| | - | - | - | - |

## 2.16.2. timer.start : 타이머 시작

| | |
|---|---|
| ■ 기 능 | 타이머를 시작합니다. |
| ■ 형 태 | timer.start() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| | | | | | |

■ 세부사항

➢ 타이머를 시작하는 명령어로 인자가 필요하지 않습니다.

■ 예 제

**[예제1] timer.start 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | motion.speed = 100 | -- 이동속도 설정 |
| 2 | | |
| 3 | timer = time() | -- 타이머 생성 |
| 4 | timer.start() | -- 타이머 시작 |
| 5 | | |
| 6 | movep(lp1) | -- lp1 위치로 로봇 이동 |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | movel(lp2) | -- lp2 위치로 로봇 이동 |
| 9 | delay(500) | -- 0.5초간 대기 |
| 10 | | |
| 11 | timer.stop() | -- 타이머 정지 |
| 12 | result = timer.elapsed() | -- 타이머 시작과 정지 사이의 시간 측정 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| timer=time() | | | - | - |
| timer.start() | | | - | - |
| timer.stop() | - | | - | - |
| timer.elapsed() | - | - | - | - |
| | - | - | - | - |

## 2.16.3. timer.stop : 타이머 정지

| | |
|---|---|
| ■ 기 능 | 타이머를 정지합니다. |
| ■ 형 태 | timer.stop() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| | | | | | |

■ 세부사항

➢ 타이머를 정지하는 명령어로 인자가 필요하지 않습니다.

■ 예 제

**[예제1] timer.stop 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | motion.speed = 100 | -- 이동속도 설정 |
| 2 | | |
| 3 | timer = time() | -- 타이머 생성 |
| 4 | timer.start() | -- 타이머 시작 |
| 5 | | |
| 6 | movep(lp1) | -- lp1 위치로 로봇 이동 |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | movel(lp2) | -- lp2 위치로 로봇 이동 |
| 9 | delay(500) | -- 0.5초간 대기 |
| 10 | | |
| 11 | timer.stop() | -- 타이머 정지 |
| 12 | result = timer.elapsed() | -- 타이머 시작과 정지 사이의 시간 측정 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| timer=time() | | | - | - |
| timer.start() | | | - | - |
| timer.stop() | - | | - | - |
| timer.elapsed() | - | - | - | - |
| | - | - | - | - |

## 2.16.4. timer.elapsed : 소요시간 계산

| | |
|---|---|
| ■ 기 능 | **시작시간과 정지시간의 차이값을 계산하여 반환합니다.** |
| ■ 형 태 | Result = timer.elapsed() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| | | | | | |

■ **세부사항**

➢ **타이머의 소요시간을 계산하는 명령어로 인자가 필요하지 않습니다.**

■ 반환값

| 값 | 자료형 | 설명 |
|---|---|---|
| Result | double | 타이머 시작, 정지까지의 소요시간 (단위 : 초) |

■ 예 제

**[예제1] timer=time 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | motion.speed = 100 | -- 이동속도 설정 |
| 2 | | |
| 3 | timer = time() | -- 타이머 생성 |
| 4 | timer.start() | -- 타이머 시작 |
| 5 | | |
| 6 | movep(lp1) | -- lp1 위치로 로봇 이동 |
| 7 | delay(1000) | -- 1초간 대기 |
| 8 | movel(lp2) | -- lp2 위치로 로봇 이동 |
| 9 | delay(500) | -- 0.5초간 대기 |
| 10 | | |
| 11 | timer.stop() | -- 타이머 정지 |
| 12 | result = timer.elapsed() | -- 타이머 시작과 정지 사이의 시간 측정 |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| timer=time() | | | - | - |
| timer.start() | | | - | - |
| timer.stop() | - | | - | - |
| timer.elapsed() | - | - | - | - |
| | - | - | - | - |

## 2.17. 프로그램 제어 스크립트

프로그램 제어 스크립트를 사용하여, 현재 실행 중인 프로그램을 일시정지 혹은 정지가 가능합니다.

**표 2-94 프로그램 제어 스크립트 명령어 리스트**

| No. | 분류 | 항목 | 설명 | 참조 |
|---|---|---|---|---|
| 1 | 시작 | robot.pause() | 현재 실행 중인 프로그램을 일시정지합니다. | 2.17.1 |
| 2 | 정지 | robot.stop() | 현재 실행 중인 프로그램을 정지합니다. | 2.17.2 |

표 2-94 는 프로그램 제어 스크립트에 대해 간략히 설명한 표이며, 보다 안전한 사용을 위해 각 항목의 참조 페이지로 이동 후 자세히 숙지 후 사용하시기 바랍니다.

## 2.17.1. robot.pause : 프로그램 일시정지

| | |
|---|---|
| ■ 기 능 | 현재 실행 중인 프로그램을 일시정지합니다. |
| ■ 형 태 | robot.pause() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| | | | | | |

■ 세부사항

➢ 프로그램을 일시정지하는 명령어로 인자가 필요하지 않습니다.

■ 예 제

**[예제1] pause 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | val1 = dio.get(1) | -- 디지털 입력 1번 포트의 값 읽기 |
| 2 | val2 = dio.get(2) | -- 디지털 입력 2번 포트의 값 읽기 |
| 3 | | |
| 4 | if val1 == 1 then | -- 디지털 입력 1번 포트에 신호가 |
| 5 | robot.pause() | 입력되었을 경우, 프로그램 일시 정지 |
| 6 | elseif val1 == 1 and val2 == 1 then | -- 디지털 입력 1, 2번 포트 모두 신호가 |
| 7 | robot.stop() | 입력되었을 경우, 프로그램 정지 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| robotpause() | | | | |
| robot.stop() | | | | |
| | | | | |
| | | | | |

## 2.17.2. robot.stop : 프로그램 정지

| | |
|---|---|
| ■ 기 능 | 현재 실행 중인 프로그램을 정지합니다. |
| ■ 형 태 | robot.stop() |

■ 인 자

| 인자 | 자료형 | 기본값 | 범위 | | 단위 |
|---|---|---|---|---|---|
| | | | 최소 | 최대 | |
| | | | | | |

■ 세부사항

➢ 프로그램를 정지하는 명령어로 인자가 필요하지 않습니다.

■ 예 제

**[예제1] stop 사용**

| 번호 | 스크립트 | 주석 |
|---|---|---|
| 1 | val1 = dio.get(1) | -- 디지털 입력 1번 포트의 값 읽기 |
| 2 | val2 = dio.get(2) | -- 디지털 입력 2번 포트의 값 읽기 |
| 3 | | |
| 4 | if val1 == 1 then | -- 디지털 입력 1번 포트에 신호가 |
| 5 | robot.pause() | 입력되었을 경우, 프로그램 일시 정지 |
| 6 | elseif val1 == 1 and val2 == 1 then | -- 디지털 입력 1, 2번 포트 모두 신호가 |
| 7 | robot.stop() | 입력되었을 경우, 프로그램 정지 |
| 8 | end | |

■ 에러 상황

| 발생 시점 | 검사 항목 | 메시지 |
|---|---|---|
| 컴파일 오류 | 명령어 괄호 | ')' expected (to close '('at line 5) near <eof> |

■ 관련 함수

| | | | | |
|---|---|---|---|---|
| robot.pause() | | | | |
| robot.stop() | | | | |
| | | | | |
| | | | | |