

# MODBUS Organization

## MODBUS Master Series

### MODBUS Serial Master Driver

지원 버전

TOP Design Studio

V1.4.5 이상



## CONTENTS

Touch Operation Panel을 사용해주시는 고객님께 감사 드립니다.

- 1. 시스템 구성** [2 페이지](#)

연결 가능한 기기 및 네트워크 구성에 대해 설명합니다.
- 2. 외부 장치 선택** [3 페이지](#)

TOP 기종과 외부 장치를 선택합니다.
- 3. TOP 통신 설정** [4 페이지](#)

TOP 통신 설정 방법에 대해서 설명합니다.
- 4. 외부 장치 설정** [12 페이지](#)

외부 장치의 통신 설정 방법에 대해서 설명합니다.
- 5. 케이블 표** [13 페이지](#)

연결에 필요한 케이블 사양에 대해 설명합니다.
- 6. 지원 어드레스** [15 페이지](#)

본 절을 참고하여 외부 장치와 통신 가능한 데이터 주소를 확인하십시오.

# 1. 시스템 구성

본 드라이버는 "MODBUS Organization"의 "MODBUS Protocol" 중 "Serial Master Driver" 입니다.  
 외부 장치(MODBUS Slave Protocol 지원)에 따라서 드라이버의 "명령어 코드", "프로토콜 프레임 형식" 등을 별도 설정 해야 할 수 있습니다. 이 경우 통신 방식에 따른 세부 설정 사항을 외부 장치 측에 맞추어 설정 해주십시오.  
 본 드라이버가 지원하는 외부 장치와의 시스템 구성은 아래와 같습니다.

시리즈	CPU	Link I/F	통신 방식	시스템 설정	케이블
MODBUS Slave Device			RS-232C	<a href="#">3. TOP 통신 설정</a> <a href="#">4. 외부 장치 설정</a>	<a href="#">5. 케이블 표</a>
			RS-422 (4 wire)		
			RS-485 (2 wire)		

■ 연결 가능 구성

- 1 : 1 연결

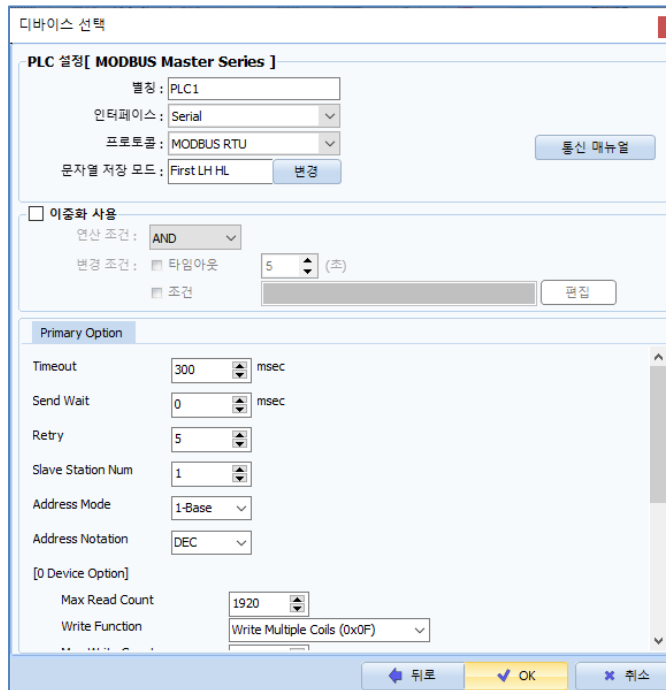
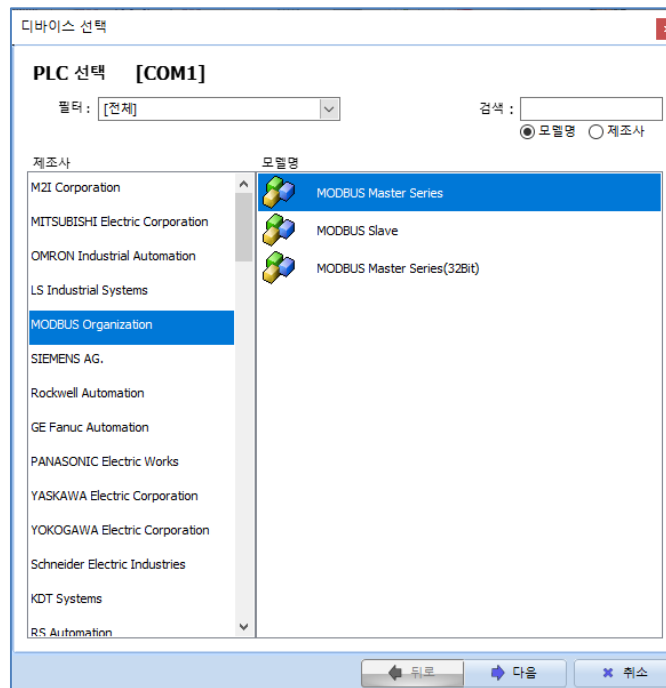


- 1 : N 연결



## 2. 외부 장치 선택

■ TOP 모델 및 포트 선택 후 외부 장치를 선택합니다.



설정 사항		내용									
TOP	모델	TOP 모델을 선택합니다.									
외부 장치	제조사	TOP와 연결 할 외부 장치의 제조사를 선택합니다. "MODBUS Organization"를 선택 하십시오.									
	PLC	TOP와 연결할 외부 장치를 선택합니다. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>모델</th> <th>인터페이스</th> <th>프로토콜</th> </tr> </thead> <tbody> <tr> <td>MODBUS Master Series</td> <td>Serial</td> <td>사용자 설정</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">지원하는 프로토콜</th> </tr> </thead> <tbody> <tr> <td>MODBUS RTU</td> <td>MODBUS ASCII</td> </tr> </tbody> </table> <p>연결을 원하는 외부 장치가 시스템 구성 가능한 기종인지 1장의 시스템 구성에서 확인 하시기 바랍니다.</p>	모델	인터페이스	프로토콜	MODBUS Master Series	Serial	사용자 설정	지원하는 프로토콜		MODBUS RTU
모델	인터페이스	프로토콜									
MODBUS Master Series	Serial	사용자 설정									
지원하는 프로토콜											
MODBUS RTU	MODBUS ASCII										

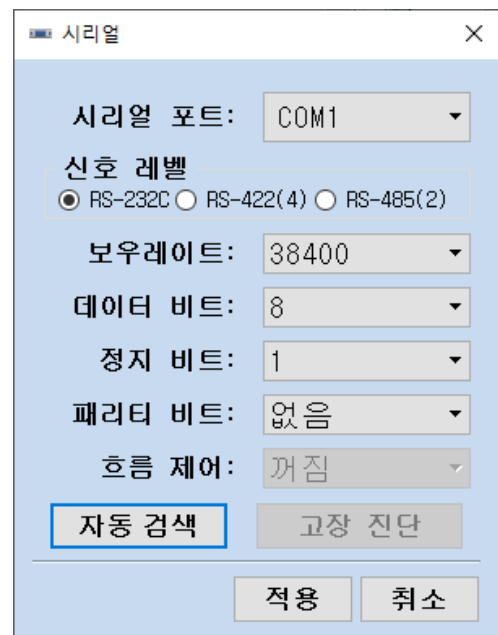
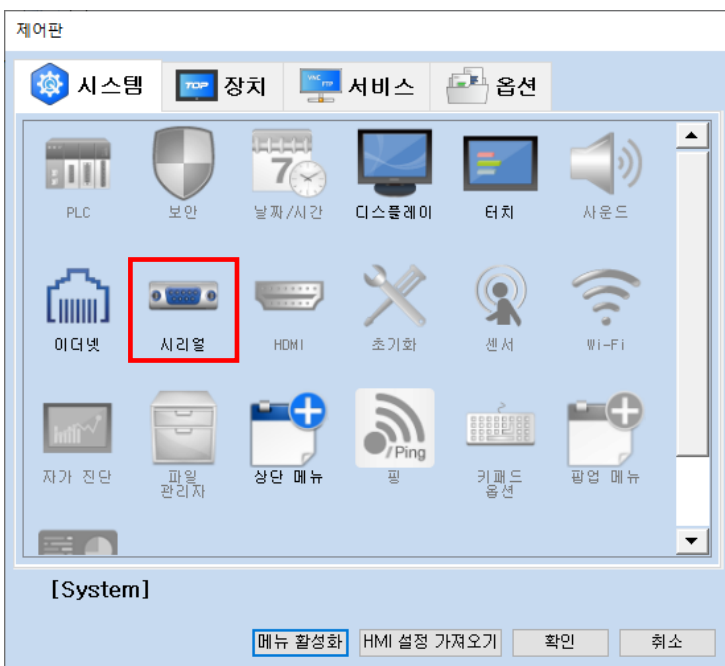
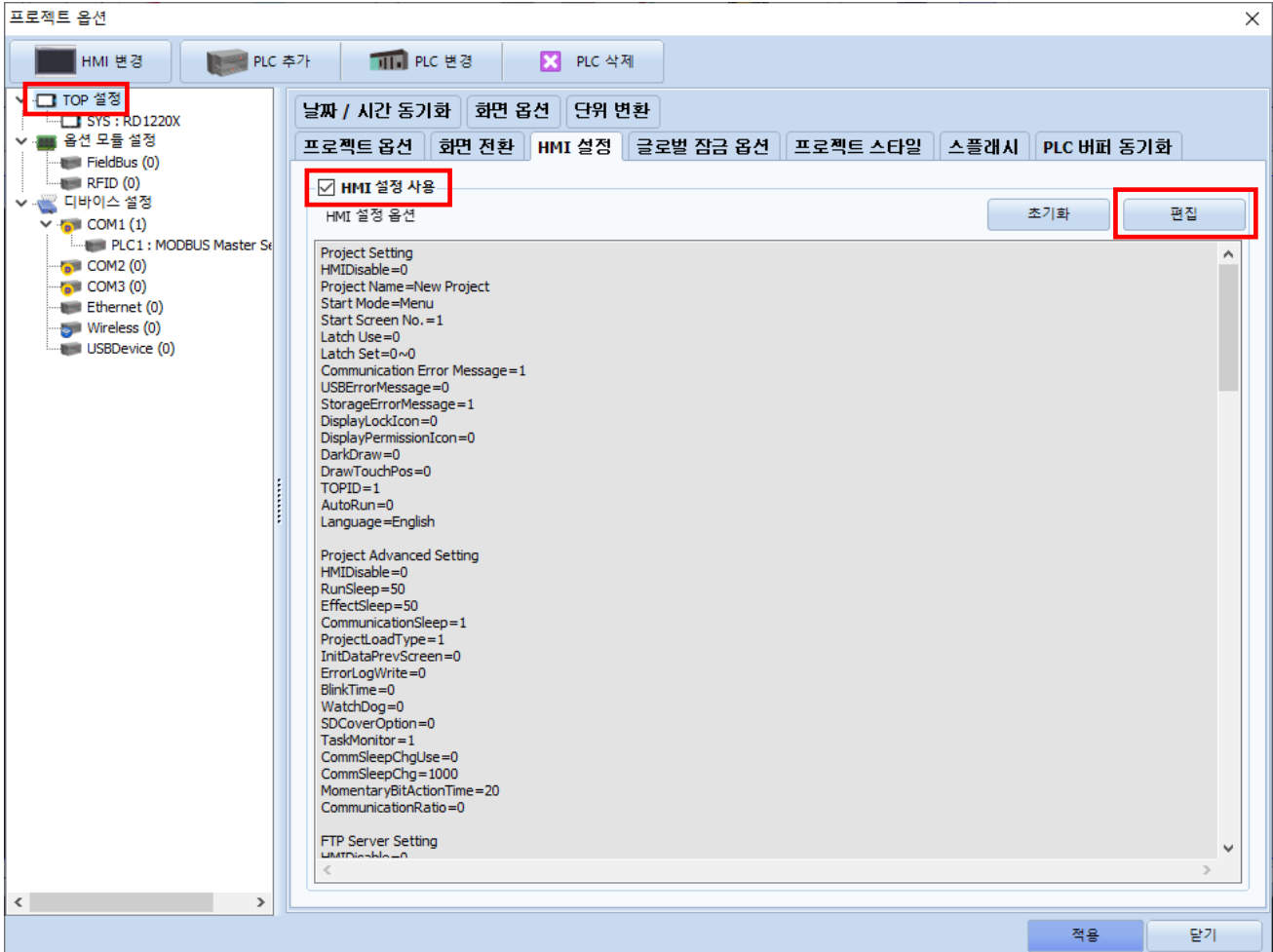
### 3. TOP 통신 설정

통신 설정은 TOP Design Studio 혹은 TOP 메인 메뉴에서 설정 가능 합니다. 통신 설정은 외부 장치와 동일하게 설정해야 합니다.

#### 3.1 TOP Design Studio 에서 통신 설정

##### (1) 통신 인터페이스 설정

- [프로젝트] → [속성] → [TOP 설정] → [HMI 설정] → [HMI 설정 사용 체크] → [편집] → [시리얼]
- TOP 통신 인터페이스를 TOP Design Studio에서 설정합니다.



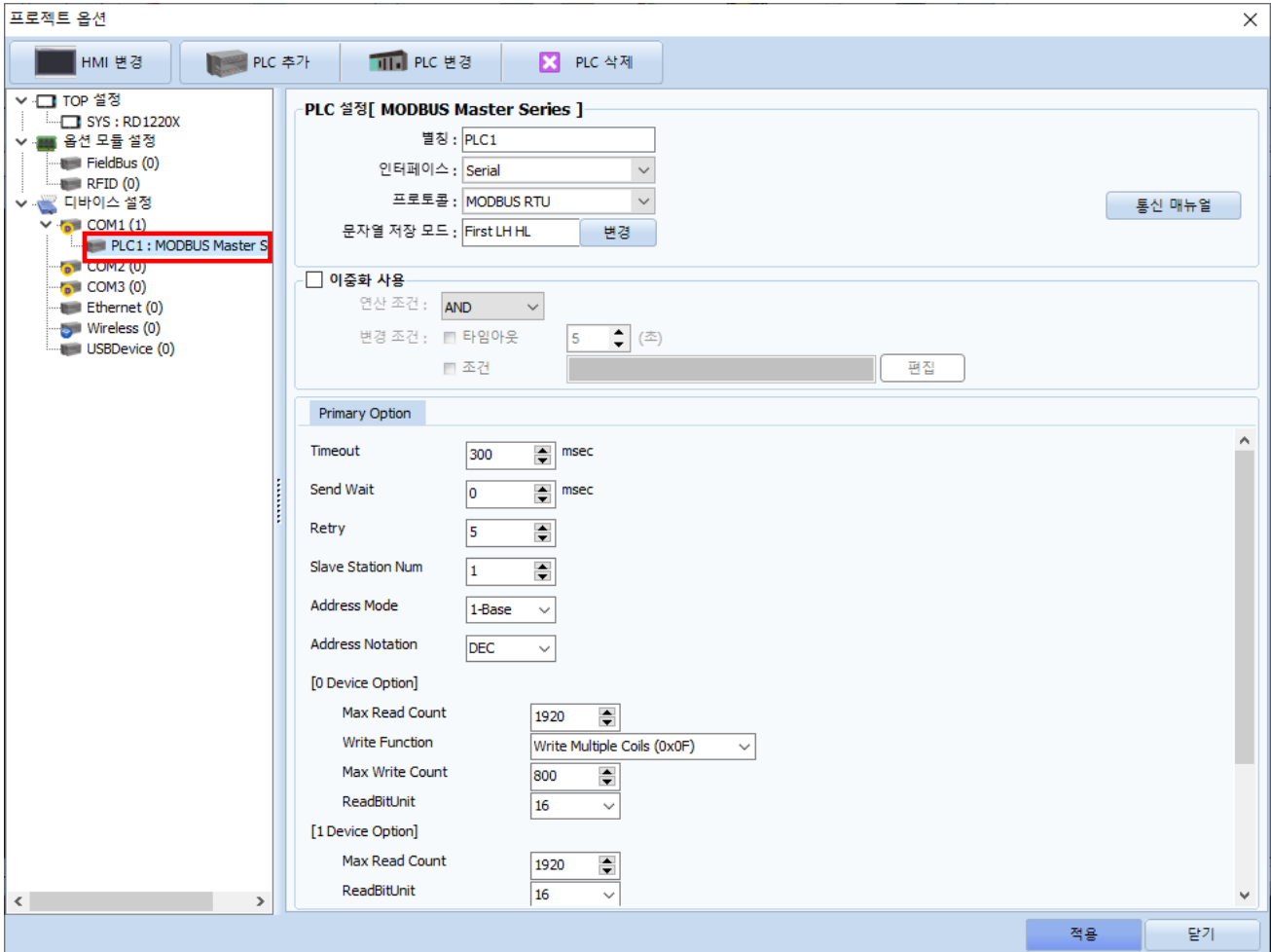
항 목	TOP	외부 장치	비 고
신호 레벨	RS-232C RS-422/485	RS-232C RS-422/485	
보우레이트	38400		
데이터 비트	8		
정지 비트	1		
패리티 비트	없음		

※ 위의 설정 내용은 본 사에서 권장하는 예제입니다.

항 목	설 명
신호 레벨	TOP - 외부 장치 간 시리얼 통신 방식을 선택합니다. (COM3는 RS-485 만 지원합니다.)
보우레이트	TOP - 외부 장치 간 시리얼 통신 속도를 선택합니다.
데이터 비트	TOP - 외부 장치 간 시리얼 통신 데이터 비트를 선택합니다.
정지 비트	TOP - 외부 장치 간 시리얼 통신 정지 비트를 선택합니다.
패리티 비트	TOP - 외부 장치 간 시리얼 통신 패리티 비트 확인 방식을 선택합니다.

(2) 통신 옵션 설정

- [프로젝트] → [프로젝트 속성] → [PLC 설정 > COM1 > PLC1 : MODBUS Master Series]
- MODBUS Serial Master 통신 드라이버의 옵션을 TOP Design Studio에서 설정합니다.



항 목	설 정	비 고
인터페이스	"Serial"을 선택합니다.	<a href="#">"2. 외부 장치 선택" 참고</a>
프로토콜	TOP - 외부 장치 간 통신 프로토콜을 선택합니다.	<a href="#">"2. 외부 장치 선택" 참고</a>
문자열 저장 모드	문자열 데이터 입력 시 데이터의 바이트 순서를 설정합니다.	
이중화	이중화 사용	이중화 설정 사용 유무를 체크합니다.
	연산 조건	변경 조건에 대한 연산 조건을 설정합니다. AND : 체크된 변경 조건이 모두 만족하면 Primary ↔ Secondary 변경 OR : 체크된 변경 조건 중 하나라도 만족하면 Primary ↔ Secondary 변경
	변경 조건	Primary ↔ Secondary 변경 조건을 설정합니다.
TimeOut (ms)	TOP가 외부 장치로부터 응답을 기다리는 시간을 설정합니다.	
SendWait (ms)	TOP가 외부 장치로부터 응답 수신 후 다음 명령어 요청 전송 간에 대기 시간을 설정합니다.	
Retry	데이터 요청 결과가 무응답/부정 응답일 때 요청 재시도 횟수를 설정합니다.	
Slave Station Num	외부 장치의 국번을 입력합니다.	
Address Mode	주소 입력 방식을 선택합니다. 1-base : 장비의 메모리 주소가 1부터 시작. 등록된 주소-1로 데이터 요청. 0-base : 장비의 메모리 주소가 0부터 시작. 등록된 주소로 데이터 요청	
Address Notation	주소 표기 방식을 선택합니다.	
<b>[0 Device Option]</b>	<b>Coil</b>	
Max Read Count	Coil 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1) *주2)
Write Function	Coil 쓰기 요청 명령어를 설정합니다. 0x05 : Force Single Coil (1 비트 단위로 쓰기. 비트 단위 동작만 사용 가능) 0x0F : Force Multiple Coils (16 비트 단위로 쓰기) Auto : 데이터 수에 따라 0x05 또는 0x0F로 요청	*주3)
Max Write Count	Coil 쓰기 요청 시 1회에 요청 할 수 있는 최대 개수를 설정합니다.	*주2)

Read Bit Unit	Coil 읽기 요청 시 요청하는 비트 수를 설정합니다. 설정 값이 16이고 화면에 이어지는 주소가 등록된 경우 1회에 최대 "Max Read Count" 수만큼 데이터를 요청합니다.	
<b>[1 Device Option]</b>	<b>Discrete Input</b>	
Max Read Count	Discrete Input 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1) *주2)
Read Bit Unit	Discrete Input 읽기 요청 시 요청하는 비트 수를 설정합니다. 설정 값이 16이고 화면에 이어지는 주소가 등록된 경우 1회에 최대 "Max Read Count" 수만큼 데이터를 요청합니다.	
<b>[3 Device Option]</b>	<b>Input Register</b>	
Max Read Count	Input Register 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1) *주2)
<b>[4 Device Option]</b>	<b>Holding Register</b>	
Max Read Count	Holding Register 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1)
Write Function	Holding Register 쓰기 요청 명령어를 설정합니다. 0x06 : Preset Single Register (1개 쓰기) 0x10 : Preset Multiple Registers (n개 쓰기) Auto : 데이터 수에 따라 0x06 또는 0x10으로 요청	*주3)
Max Write Count	명령어 0x10으로 Holding Register 데이터 쓰기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주2)

**\*주 1)**

- 각 디바이스의 Max Read Count 는 화면에 등록된 주소들이 연속되지 않을 때 통신을 여러 번 하지 않고 1회에 요청 할 주소 범위로도 사용됩니다.

예 1) 화면에 숫자 오브젝트로 400001, 400002, 400003, 400004, 400005, 400120 을 등록하고 4 디바이스의 Max Read Count 를 120 으로 설정 할 경우 400001 부터 400120 까지 연속된 주소로 가정하여 400001 부터 120 워드로 하여 한 번의 요청으로 데이터를 읽어옵니다.

예 2) 화면에 숫자 오브젝트로 400001, 400002, 400003, 400004, 400005, 400120 을 등록하고 4 디바이스의 Max Read Count 를 3 으로 설정 할 경우 400001 부터 400003 까지 3 워드, 400004 부터 400005 까지 2 워드, 400120 1 워드로 하여 세 번의 요청으로 데이터를 읽어옵니다.

예 3) 화면에 숫자 오브젝트로 400001, 400010, 400011, 400021, 400031, 400041 을 등록하고 4 디바이스의 Max Read Count 를 10 으로 설정 할 경우 400001 부터 400010 까지 10 워드, 400011 1 워드, 400021 1 워드 400031 1 워드, 400041 1 워드로 하여 다섯 번의 요청으로 데이터를 읽어옵니다.

- Max Read Count 를 0 으로 설정 할 경우 연속된 주소에 한해서 최대 120 워드만큼 요청합니다.

**\*주 2)**

- 외부 장치의 매뉴얼을 참고하여 등록된 주소부터 한번에 몇 개의 데이터가 읽기/쓰기가 가능한지 확인 하십시오.

외부 장치가 지원하는 범위보다 크게 설정 할 경우 통신이 정상적으로 이루어지지 않습니다.

예) 외부 장치의 Holding Register(4 디바이스)가 1 회 통신에 최대 10 개 워드만 응답 가능한 경우 TOP 의 통신 설정 중 4 디바이스의 Max Read Count 를 외부 장치의 사양에 맞게 10 으로 설정하십시오.

**\*주 3)**

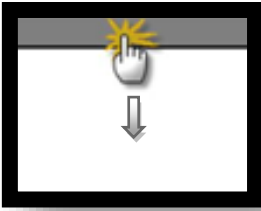
- 외부 장치의 매뉴얼을 참고하여 지원하는 쓰기 명령어에 맞게 설정하십시오.

지원하지 않는 쓰기 명령어 설정 시 데이터 쓰기 동작이 이루어지지 않습니다.

### 3.2 TOP에서 통신 설정

※ “3.1 TOP Design Studio 에서 통신 설정” 항목의 “HMI 설정 사용”을 체크 하지 않은 경우의 설정 방법입니다.

■ TOP 화면 상단을 터치하여 아래로 드래그 합니다. 팝업 창의 “EXIT”를 터치하여 시스템 화면으로 이동합니다.



#### (1) 통신 인터페이스 설정

■ [제어판] → [시리얼]



항 목	TOP	외부 장치	비 고
신호 레벨	RS-232C RS-422/485	RS-232C RS-422/485	
보우레이트	38400		
데이터 비트	8		
정지 비트	1		
패리티 비트	없음		

※ 위의 설정 내용은 본 사에서 권장하는 예제입니다.

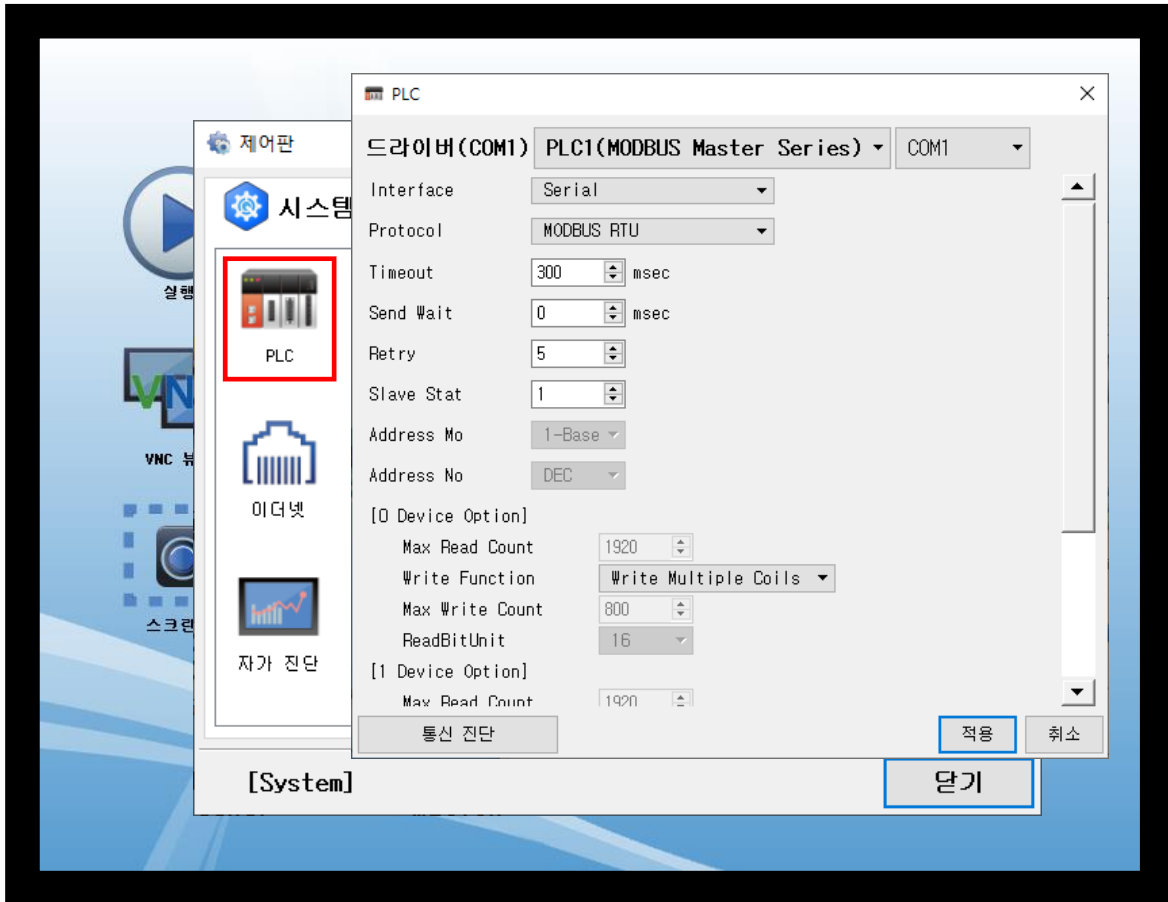
항 목	설 명
신호 레벨	TOP - 외부 장치 간 시리얼 통신 방식을 선택합니다. (COM3는 RS-485 만 지원합니다.)
보우레이트	TOP - 외부 장치 간 시리얼 통신 속도를 선택합니다.
데이터 비트	TOP - 외부 장치 간 시리얼 통신 데이터 비트를 선택합니다.
정지 비트	TOP - 외부 장치 간 시리얼 통신 정지 비트를 선택합니다.
패리티 비트	TOP - 외부 장치 간 시리얼 통신 패리티 비트 확인 방식을 선택합니다.





(2) 통신 옵션 설정

■ [제어판] → [PLC]



항 목	설 정	비 고
인터페이스	"Serial"을 선택합니다.	<a href="#">"2. 외부 장치 선택" 참고</a>
프로토콜	TOP - 외부 장치 간 통신 프로토콜을 선택합니다.	
문자열 저장 모드	문자열 데이터 입력 시 데이터의 바이트 순서를 설정합니다.	
이중화	이중화 설정 사용 유무를 체크합니다.	
연산 조건	변경 조건에 대한 연산 조건을 설정합니다. AND : 체크된 변경 조건이 모두 만족하면 Primary ↔ Secondary 변경 OR : 체크된 변경 조건 중 하나라도 만족하면 Primary ↔ Secondary 변경	
변경 조건	Primary ↔ Secondary 변경 조건을 설정합니다.	
TimeOut (ms)	TOP가 외부 장치로부터 응답을 기다리는 시간을 설정합니다.	
SendWait (ms)	TOP가 외부 장치로부터 응답 수신 후 다음 명령어 요청 전송 간에 대기 시간을 설정합니다.	
Retry	데이터 요청 결과가 무응답/부정 응답일 때 요청 재시도 횟수를 설정합니다.	
Slave Station Num	외부 장치의 국번을 입력합니다.	
Address Mode	주소 입력 방식을 선택합니다. 1-base : 장비의 메모리 주소가 1부터 시작. 등록된 주소-1로 데이터 요청. 0-base : 장비의 메모리 주소가 0부터 시작. 등록된 주소로 데이터 요청	
Address Notation	주소 표기 방식을 선택합니다.	
<b>[0 Device Option]</b>	<b>Coil</b>	
Max Read Count	Coil 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1) *주2)
Write Function	Coil 쓰기 요청 명령어를 설정합니다. 0x05 : Force Single Coil (1 비트 단위로 쓰기. 비트 단위 동작만 사용 가능) 0x0F : Force Multiple Coils (16 비트 단위로 쓰기) Auto : 데이터 수에 따라 0x05 또는 0x0F로 요청	*주3)
Max Write Count	Coil 쓰기 요청 시 1회에 요청 할 수 있는 최대 개수를 설정합니다.	*주2)
Read Bit Unit	Coil 읽기 요청 시 요청하는 비트 수를 설정합니다. 설정 값이 16이고 화면에 이어지는 주소가 등록된 경우 1회에 최대 "Max Read	

	Count" 수만큼 데이터를 요청합니다.	
<b>[1 Device Option]</b>	<b>Discrete Input</b>	
Max Read Count	Discrete Input 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1) *주2)
Read Bit Unit	Discrete Input 읽기 요청 시 요청하는 비트 수를 설정합니다. 설정 값이 16이고 화면에 이어지는 주소가 등록된 경우 1회에 최대 "Max Read Count" 수만큼 데이터를 요청합니다.	
<b>[3 Device Option]</b>	<b>Input Register</b>	
Max Read Count	Input Register 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1) *주2)
<b>[4 Device Option]</b>	<b>Holding Register</b>	
Max Read Count	Holding Register 읽기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주1)
Write Function	Holding Register 쓰기 요청 명령어를 설정합니다. 0x06 : Preset Single Register (1개 쓰기) 0x10 : Preset Multiple Registers (n개 쓰기) Auto : 데이터 수에 따라 0x06 또는 0x10으로 요청	*주3)
Max Write Count	명령어 0x10으로 Holding Register 데이터 쓰기 요청 시 1회에 요청할 수 있는 최대 개수를 설정합니다.	*주2)

**\*주 1)**

- 각 디바이스의 Max Read Count 는 화면에 등록된 주소들이 연속되지 않을 때 통신을 여러 번 하지 않고 1 회에 요청 할 주소 범위로도 사용됩니다.

예 1) 화면에 숫자 오브젝트로 400001, 400002, 400003, 400004, 400005, 400120 을 등록하고 4 디바이스의 Max Read Count 를 120 으로 설정 할 경우 400001 부터 400120 까지 연속된 주소로 가정하여 400001 부터 120 워드로 하여 한 번의 요청으로 데이터를 읽어옵니다.

예 2) 화면에 숫자 오브젝트로 400001, 400002, 400003, 400004, 400005, 400120 을 등록하고 4 디바이스의 Max Read Count 를 3 으로 설정 할 경우 400001 부터 400003 까지 3 워드, 400004 부터 400005 까지 2 워드, 400120 1 워드로 하여 세 번의 요청으로 데이터를 읽어옵니다.

예 3) 화면에 숫자 오브젝트로 400001, 400010, 400011, 400021, 400031, 400041 을 등록하고 4 디바이스의 Max Read Count 를 10 으로 설정 할 경우 400001 부터 400010 까지 10 워드, 400011 1 워드, 400021 1 워드 400031 1 워드, 400041 1 워드로 하여 다섯 번의 요청으로 데이터를 읽어옵니다.

- Max Read Count 를 0 으로 설정 할 경우 연속된 주소에 한해서 최대 120 워드만큼 요청합니다.

**\*주 2)**

- 외부 장치의 매뉴얼을 참고하여 등록된 주소부터 한번에 몇 개의 데이터가 읽기/쓰기가 가능한지 확인 하십시오.

외부 장치가 지원하는 범위보다 크게 설정 할 경우 통신이 정상적으로 이루어지지 않습니다.

예) 외부 장치의 Holding Register(4 디바이스)가 1 회 통신에 최대 10 개 워드만 응답 가능한 경우 TOP 의 통신 설정 중 4 디바이스의 Max Read Count 를 외부 장치의 사양에 맞게 10 으로 설정하십시오.

**\*주 3)**

- 외부 장치의 매뉴얼을 참고하여 지원하는 쓰기 명령어에 맞게 설정하십시오.

지원하지 않는 쓰기 명령어 설정 시 데이터 쓰기 동작이 이루어지지 않습니다.

### 3.3 통신 진단

■ TOP - 외부 장치 간 인터페이스 설정 상태를 확인

- TOP 화면 상단을 터치하여 아래로 드래그. 팝업 창의 "EXIT"를 터치하여 메인 화면으로 이동한다
- [제어판] → [시리얼]에서 연결된 포트의 설정이 외부 장치의 설정과 같은지 확인한다

■ 포트 통신 이상 유무 진단

- [제어판] → [PLC] 에서 "통신 진단"을 터치한다.
- 통신 연결 여부를 확인한다.

통신 진단, 성공	통신 설정 정상
에러 메시지	통신 설정 비정상
	- 케이블 및 TOP, 외부 장치의 설정 상태 확인한다. (통신 진단 시트 참고)

■ 통신 진단 시트

- 외부 단말기와 통신 연결에 문제가 있을 경우 아래 시트의 설정 내용을 확인 바랍니다.

항목	내용	확인		참 고	
시스템 구성	시스템 연결 방법	OK	NG	<a href="#">1. 시스템 구성</a>	
	케이블	OK	NG		
TOP	버전 정보	OK	NG	<a href="#">2. 외부 장치 선택</a> <a href="#">3. TOP 통신 설정</a>	
	통신 포트	OK	NG		
	통신 드라이버 및 프로토콜	OK	NG		
	기타 세부 설정 사항	OK	NG		
	상대 국번	프로젝트 설정	OK		NG
		통신 진단	OK		NG
	시리얼 파라미터	전송 속도	OK		NG
		데이터 비트	OK		NG
정지 비트		OK	NG		
패리티 비트		OK	NG		
외부 장치	CPU 명칭	OK	NG	<a href="#">4. 외부 장치 설정</a>	
	통신 포트	OK	NG		
	프로토콜	OK	NG		
	국번	OK	NG		
	기타 세부 설정 사항	OK	NG		
	시리얼 파라미터	전송 속도	OK		NG
		데이터 비트	OK		NG
		정지 비트	OK		NG
패리티 비트		OK	NG		
어드레스 범위 확인		OK	NG	<a href="#">6. 지원 어드레스</a> (자세한 내용은 PLC 제조사의 매뉴얼을 참고 하시기 바랍니다.)	

## 4. 외부 장치 설정

---

제조사 사용자 매뉴얼을 참고하여 모드버스 슬레이브(서버)로 설정 하십시오.

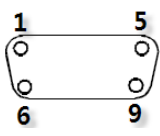
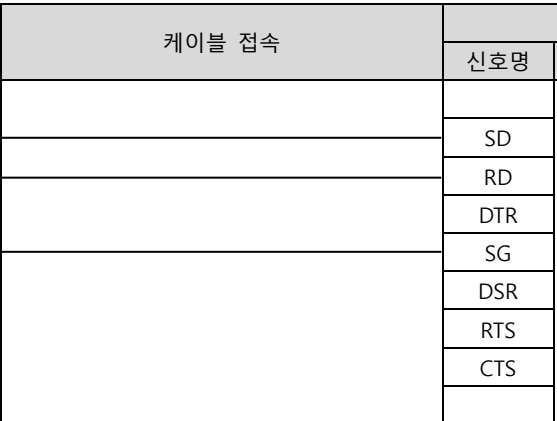


- 프로토콜 설정 중 RTU / ASCII 선정에 주의 하십시오.
  - 외부 장치의 메모리 주소를 확인하십시오.
-

## 5. 케이블 표

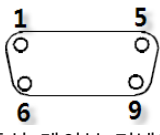
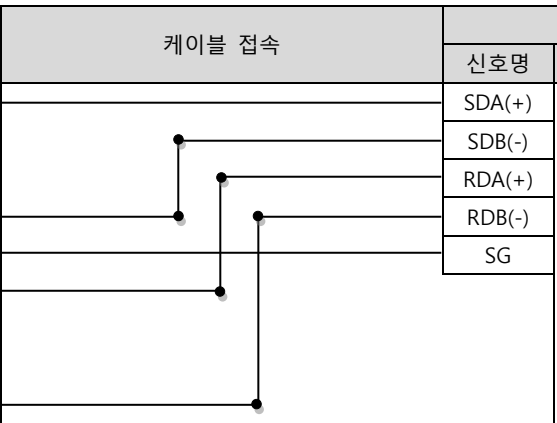
본 Chapter는 TOP과 해당 기기 간 정상 통신을 위한 케이블 다이어그램을 소개 합니다.  
 (본 절에서 설명되는 케이블 다이어그램은 외부 장치 제조사의 권장 사항과 다를 수 있습니다)

### ■ RS-232C (1 : 1 연결)

COM1 / COM2			케이블 접속	외부 장치	
핀 배열*주1)	신호명	핀번호		신호명	
 <p>통신 케이블 커넥터 전면 기준, D-SUB 9 Pin male(수, 블록)</p>	CD	1			
	RD	2		SD	
	SD	3		RD	
	DTR	4		DTR	
	SG	5		SG	
	DSR	6		DSR	
	RTS	7		RTS	
	CTS	8		CTS	
		9			

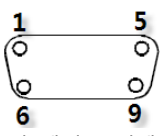
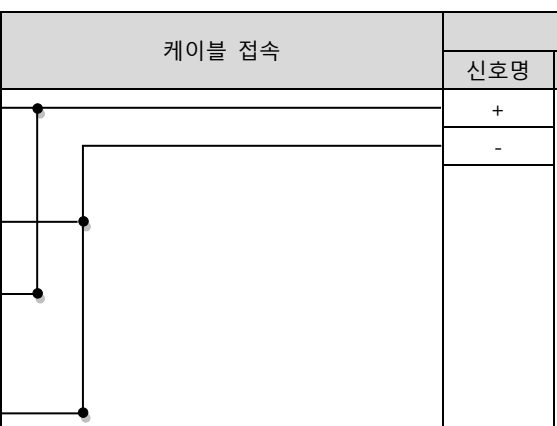
\*주1) 핀 배열은 케이블 접속 커넥터의 접속면에서 본 것 입니다.

### ■ RS-422 (1 : 1 연결)

COM1 / COM2			케이블 접속	외부 장치	
핀 배열*주1)	신호명	핀번호		신호명	
 <p>통신 케이블 커넥터 전면 기준, D-SUB 9 Pin male(수, 블록)</p>	RDA(+)	1		SDA(+)	
		2		SDB(-)	
		3		RDA(+)	
	RDB(-)	4		RDB(-)	
	SG	5		SG	
	SDA(+)	6			
		7			
		8			
	SDB(-)	9			

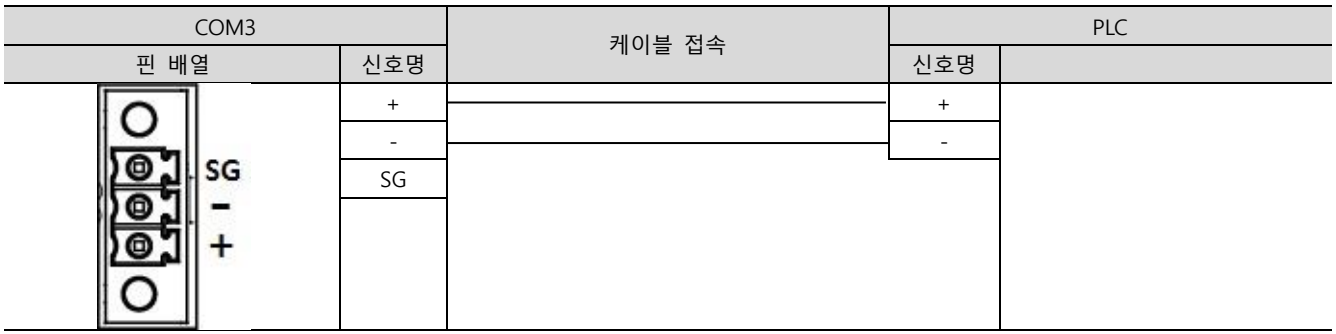
\*주1) 핀 배열은 케이블 접속 커넥터의 접속면에서 본 것 입니다.

### ■ RS-485 (1 : 1 연결)

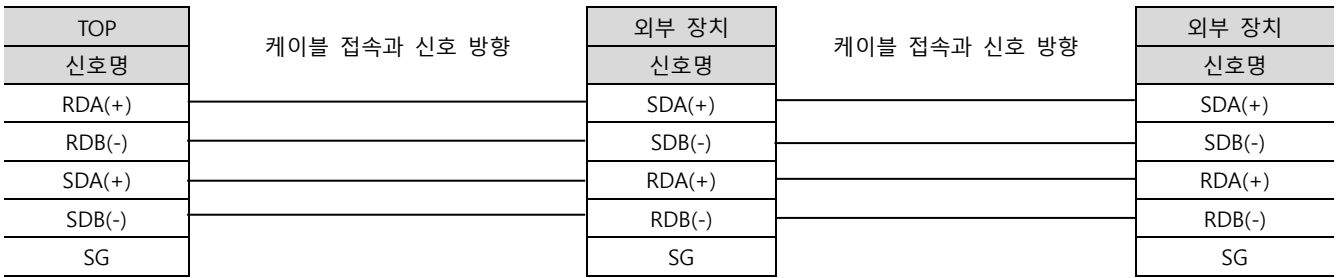
COM1 / COM2			케이블 접속	외부 장치	
핀 배열*주1)	신호명	핀번호		신호명	
 <p>통신 케이블 커넥터 전면 기준, D-SUB 9 Pin male(수, 블록)</p>	RDA(+)	1		+	
		2		-	
		3			
	RDB(-)	4			
	SG	5			
	SDA(+)	6			
		7			
		8			
	SDB(-)	9			

\*주1) 핀 배열은 케이블 접속 커넥터의 접속면에서 본 것 입니다.

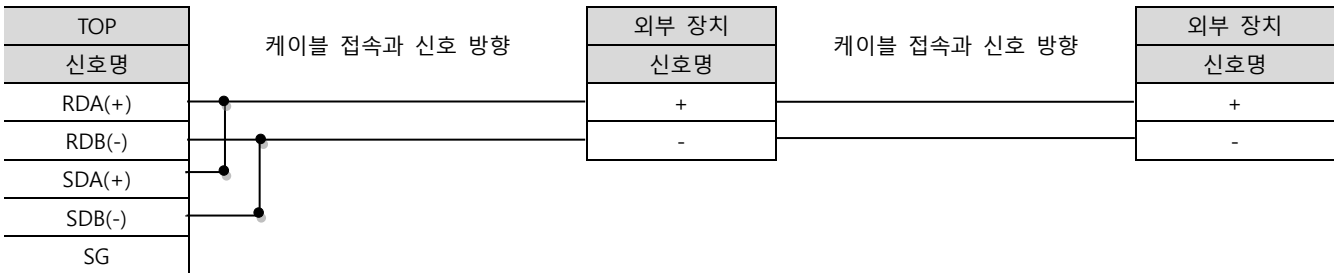
■ RS-485 (1 : 1 연결)



■ RS-422 (1 : N 연결)



■ RS-485 (1 : N 연결)



## 6. 지원 어드레스

TOP에서 사용 가능한 디바이스는 아래와 같습니다.

CPU 모듈 시리즈/타입에 따라 디바이스 범위(어드레스) 차이가 있을 수 있습니다. TOP 시리즈는 외부 장치 시리즈가 사용하는 최대 어드레스 범위를 지원합니다. 사용하고자 하는 장치가 지원하는 어드레스 범위를 벗어 나지 않도록 각 CPU 모듈 사용자 매뉴얼을 참조/주의 하십시오.

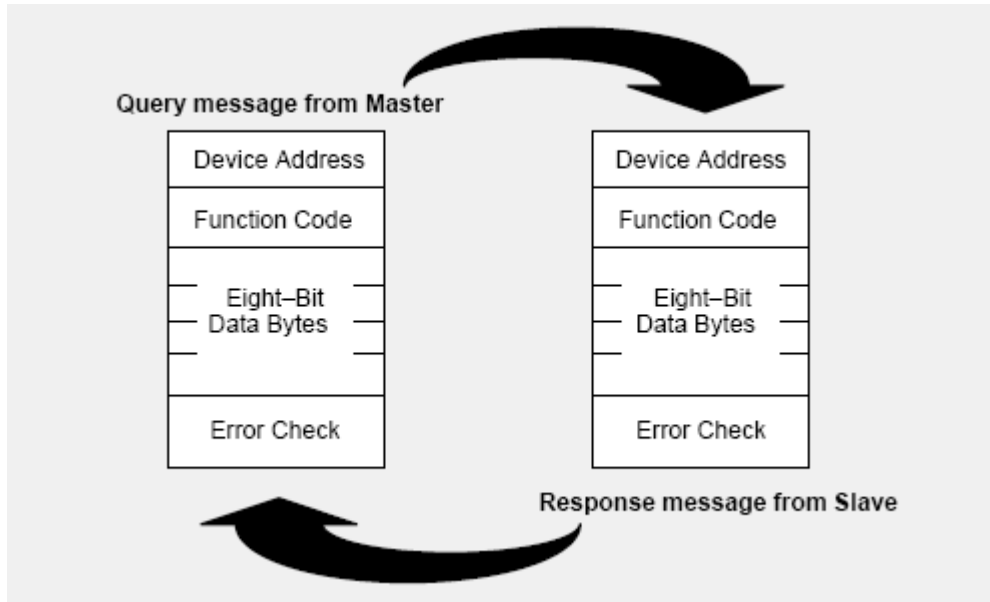
	비트	워드	비고
Coil	000001 – 065536	000001 – 065521	
Discrete Input	100001 – 165536	100001 – 165521	*주1)
Input Register	300001.00 – 365536.15	300001 – 365536	*주1)
Holding Register	400001.00 – 465536.15	400001 – 465536	

\*주1) 워기 전용

## Appendix A. Standard MODBUS Protocol

본 기기의 "MODBUS Serial Master Driver"가 지원하는 MODBUS 프로토콜 명령어 및 디바이스에 대해 설명 합니다.

At the message level, the MODBUS protocol still applies the master-slave principle even though the network communication method is peer-to-peer. If a controller originates a message, it does so as a master device, and expects a response from a slave device. Similarly, when a controller receives a message it constructs a slave response and returns it to the originating controller.



**The Query:** The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

**The Response:** If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.



## A.1 "0" Device (Coil)

### Read Single Coil : 01

MASTER 기기에서 Slave 기기 측(국번:17번)의 "000020-000056 Coil" 데이터를 읽어 오는 예제를 통해 "01"명령어 프레임에 설명 합니다.

#### ■ RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 기번	패킷번호	전부다바이스		다바이스번호		체크범위 (CRC)	
	H L	H L	H L	H L	L H	H		
Hex	11	01	00	13	00	25	-	-

( Slave → Master : 응답 프레임 )

Comment	Slave 기번	패킷번호	데이터 개수(byte)	데이터				체크범위 (CRC)	
	H L	H L	L	H L	H L	H L	H L	L H	
Hex	11	01	05	CD	6B	B2	0E	1B	-

#### ■ Coils 데이터 상태

Coils	27	26	25	24	23	22	21	20
on/off	1	1	0	0	1	1	0	1
Coils	35	34	33	32	31	30	29	28
on/off	0	1	1	0	1	0	1	1
Coils	43	42	41	40	39	38	37	36
on/off	1	0	1	1	0	0	1	0
Coils	51	50	49	48	47	46	45	44
on/off	0	0	0	0	1	1	1	0
Coils	59	58	57	56	55	54	53	52
on/off	-	-	-	1	1	0	1	1

0: OFF / 1:ON

#### ■ ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 기번		패킷번호		전부다바이스			다바이스번호				체크범위 (CRC)		Tail		
	H L	H L	H L	H L	-	-	L	H	-	-	L	L	H	CR	LF		
ASCII	:	1	1	0	1	0	0	1	3	0	0	2	5	-	-	CR	LF
Hex	3A	31	31	30	31	30	30	31	33	30	30	32	35	-	-	0D	0A

( Slave → Master : 응답 프레임 )

Comment	Header	Slave 기번		패킷번호		데이터 개수(byte)		Coils 27~20		Coils 35~28		Coils 43~36		Coils 51~44		Coils 56~52		체크범위 (CRC)		Tail	
	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	L H	L H	CR	LF					
ASCII	:	1	1	0	1	0	5	C	D	6	B	B	2	0	E	1	B	-	-	CR	LF
Hex	3A	31	31	30	31	30	35	43	44	36	42	42	32	30	45	31	42	-	-	0D	0A



**Force Single Coil : 05**

MASTER 기기에서 Slave 기기 측의 Coil 000173 에 FORCE "ON" 하는 예제를 통해 "05"명령어 프레임에 설명 합니다.

■ RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 기판	명령어		전누디바이스				Force data		체크점비 (CRC)	
		H	L	H	L	H	L	L	H		
Hex	11	05	00	AC	FF	00	—	—			

( Slave → Master : 응답 프레임 )

Comment	Slave 기판	명령어		전누디바이스				Force data		체크점비 (CRC)	
		H	L	H	L	H	L	L	H		
Hex	11	05	00	AC	FF	00	—	—			

■ Force Data

	High	Low
Force ON	FF <sub>H</sub>	00 <sub>H</sub>
Force OFF	00 <sub>H</sub>	00 <sub>H</sub>

■ ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 기판		명령어		전누디바이스				Force data				체크점비 (CRC)		Tail	
		H	L	H	L	H	-	-	L	H	-	-	L	L	H	CR	LF
ASCII	:	1	1	0	5	0	0	1	3	0	0	2	5	—	—	0D	0A
Hex	3A	31	31	30	31	30	30	41	43	45	45	30	30	—	—	0D	0A

( Slave → Master : 응답 프레임 )

comment	Header	Slave 기판		명령어		전누디바이스				Force data				체크점비 (CRC)		Tail	
		H	L	H	L	H	-	-	L	H	-	-	L	L	H	CR	LF
ASCII	:	1	1	0	5	0	0	1	3	0	0	2	5	—	—	0D	0A
Hex	3A	31	31	30	31	30	30	41	43	45	45	30	30	—	—	0D	0A

## A.2 "1" Device (Discrete Input)

### Read Input Status : 02

MASTER 기기에서 Slave 기기 측(국번:17번)의 "100197~100218 Input" 데이터를 읽어 오는 예제를 통해 "02"명령어 프레임을 설명합니다.

#### RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 기구번호	프레임번호	신부디바이스번호		디바이스주소		체크점비 (CRC)	
	H L	H L	H L	H L	L H	L H		
Hex	11	02	00	C4	00	16	—	—

( Slave → Master : 응답 프레임 )

Comment	Slave 기구번호	프레임번호	데이터 개수(byte)	데이터(Inputs)			체크점비 (CRC)	
	H L	H L	10204~10197	10212~10205	10218~10213	L H	L H	
Hex	11	02	03	AC	DB	35	—	—

Coils 데이터 상태

Coils on/off	204	203	202	201	200	199	198	197
Coils on/off	212	211	210	209	208	207	206	205
Coils on/off	220	219	218	217	216	215	214	213
	1	0	1	0	1	1	0	0
	1	1	0	1	1	0	1	1
	-	-	1	1	0	1	0	1

0: OFF / 1:ON

#### ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 기구번호		프레임번호		신부디바이스번호			디바이스주소				체크점비 (CRC)		Tail		
	H L	H L	H L	H L	- -	L	H - -	L	L H	CR	LF						
ASCII	:	1	1	0	2	0	0	C	4	0	0	1	6	—	—	0D	0A
Hex	3A	31	31	30	32	30	30	43	34	30	30	31	36	—	—	0D	0A

( Slave → Master : 응답 프레임 )

Comment	Header	Slave 기구번호		프레임번호		데이터 개수(byte)	데이터(Inputs)						체크점비 (CRC)		Tail		
	H L	H L	H L	0	3	10204~10197	10212~10205	10218~10213	L H	L H	CR	LF					
ASCII	:	1	1	0	2	0	3	A	C	D	B	3	5	—	—	0D	0A
Hex	3A	31	31	30	31	30	35	41	43	44	42	33	35	—	—	0D	0A

### A.3 "3" Device (Input Register)

#### Read Input Registers : 04

MASTER 기기에서 Slave 기기 측(국번:17번)의 "30009 Register" 데이터를 읽어 오는 예제를 통해 "03"명령어 프레임 설명 합니다.

#### ■ RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 기구번호	패킷번호		신부디바이스		디바이스주소수 (Word Count)		체크점비 (CRC)	
		H	L	H	L	L	H	L	H
Hex	11	04	00	08	00	01	—	—	—

( Slave → Master : 응답 프레임 )

Comment	Slave 기구번호	패킷번호	데이터 개수(byte)	데이터 Register 30009	체크점비 (CRC)	
			H	L	L	H
Hex	11	04	02	00	0A	—

#### ■ ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 기구번호		패킷번호		신부디바이스			디바이스주소수 (Word)			체크점비 (CRC)		Tail			
		H	L	H	L	H	-	-	L	H	-	-	L	L	H	CR	LF
ASCII	:	1	1	0	1	0	0	0	8	0	0	0	1	—	—	0D	0A
Hex	3A	31	31	30	31	30	30	30	38	30	30	30	31	—	—	0D	0A

( Slave → Master : 응답 프레임 )

Comment	Header	Slave 기구번호		패킷번호		데이터 개수(byte)		데이터 Register 40108			체크점비 (CRC)		Tail		
				H	L	H	-	-	L	L	H	CR	LF		
ASCII	:	1	1	0	4	0	2	0	0	0	A	—	—	0D	0A
Hex	3A	31	31	30	31	30	35	30	30	30	41	—	—	0D	0A

### A.4 "4" Device (Holding Register)

#### Read Holding Registers : 03

MASTER 기기에서 Slave 기기 측(국번:17)의 "400108 - 400110 Register" 데이터를 읽어 오는 예제를 통해 "03"명령어 프레임에 설명 합니다.

#### ■ RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 개수	주소		디바이스주소		체크섬비 (CRC)	
		H	L	H	L	L	H
Hex	11	03	00	6B	00	03	— —

( Slave → Master : 응답 프레임 )

Comment	Slave 개수	주소	데이터				체크섬비 (CRC)			
		주소 개수(byte)	Register 40108	Register 40109	Register 40110					
			H	L	H	L	H	L		
Hex	11	03	06	02	2B	00	00	00	64	— —

#### ■ ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 개수		주소		디바이스주소			디바이스주소 (Word)			체크섬비 (CRC)		Tail			
		H	L	H	L	H	-	-	L	H	-	-	L	H	CR	LF	
ASCII	:	1	1	0	1	0	0	1	3	0	0	2	5	—	—		
Hex	3A	31	31	30	31	30	30	31	33	30	30	32	35	—	—	0D	0A

( Slave → Master : 응답 프레임 )

Comment	Header	Slave 개수		주소		데이터			데이터			데이터			체크섬비 (CRC)		Tail				
ASCII	:	1	1	0	3	0	6	0	2	2	B	0	0	0	0	0	0	6	4		
Hex	3A	31	31	30	31	30	35	30	32	32	42	30	30	30	30	30	30	36	34	0D	0A

**Preset Single Register : 06**

Slave 기기 측의 40002 Register 에 00 03 (hex) 데이터를 입력 하는 예제를 통해 "06"명령어 프레임을 설명 합니다.

■ RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 기구번호	명령어		선택데이터비트		Preset data		체크섬비 (CRC)	
		H	L	H	L	H	L	L	H
Hex	11	06	00	01	00	03	—	—	

( Slave → Master : 응답 프레임 )

Comment	Slave 기구번호	명령어		선택데이터비트		Preset data		체크섬비 (CRC)	
		H	L	H	L	H	L	L	H
Hex	11	06	00	01	00	03	—	—	

■ ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 기구번호		명령어		선택데이터비트				Preset data			체크섬비 (CRC)		Tail		
		H	L	H	L	H	-	-	L	H	-	-	L	L	H	CR	LF
ASCII	:	1	1	0	6	0	0	0	1	0	0	0	3	—	—	0D	0A
Hex	3A	31	31	30	36	30	30	30	31	30	30	30	33	—	—	0D	0A

( Slave → Master : 응답 프레임 )

comment	Header	Slave 기구번호		명령어		선택데이터비트				Preset data			체크섬비 (CRC)		Tail		
		H	L	H	L	H	-	-	L	H	-	-	L	L	H	CR	LF
ASCII	:	1	1	0	6	0	0	0	1	0	0	0	3	—	—	0D	0A
Hex	3A	31	31	30	36	30	30	30	31	30	30	30	33	—	—	0D	0A

**Preset Multiple Register : 10**

Slave 기기 측의 40002 Register 에 "00 0A (hex)", "01 02 (hex)" 연속한 두 개의 데이터를 입력 하는 예제를 통해 "10"명령어 프레임임을 설명 합니다. (Error Code : 90<sub>H</sub>)

■ RTU Mode

( Master → Slave : 요청 프레임 )

Comment	Slave 기판	패킷번호	전투디바이스		Quantity of Register (Word Count)		데이터 개수(Byte)	데이터				체크섬비 (CRC)	
			H	L	H	L		H	L	H	L	L	H
Hex	11	10	00	01	00	02	04	00	0A	01	02	—	—

( Slave → Master : 응답 프레임 )

Comment	Slave 기판	패킷번호	전투디바이스		Quantity of Register (Word Count)		체크섬비 (CRC)	
			H	L	H	L	L	H
Hex	11	10	00	01	00	02	—	—

■ ASCII Mode

( Master → Slave : 요청 프레임 )

comment	Header	Slave 기판		패킷번호		전투디바이스				Quantity of Register (Word Count)				데이터 개수(Byte)				데이터													
		H	L	H	L	H	-	-	L	H	-	-	L	H	-	-	L	H	-	-	L	H	-	-	L						
ASCII	:	1	1	1	0	0	0	0	1	0	0	0	2	0	4	0	0	0	A	0	1	0	2	0	0	0	0	0	1	0	2
Hex	3A	31	31	31	30	30	30	41	43	30	30	30	32	30	34	30	30	30	41	30	31	30	32	30	30	30	41	30	31	30	32

계속...

Tail	체크섬비 (CRC)	
	L	H
ASCII	CR	LF
Hex	0D	0A

( Slave → Master : 응답 프레임 )

comment	Header	Slave 기판		패킷번호		전투디바이스				Quantity of Register (Word Count)				체크섬비 (CRC)		Tail	
		H	L	H	L	H	-	-	L	H	-	-	L	L	H	CR	LF
ASCII	:	1	1	1	0	0	0	0	1	0	0	0	2	—	—	0D	0A
Hex	3A	31	31	30	31	30	30	30	31	30	30	30	32	—	—	0D	0A

## A.5 LRC/CRC Generation

### (1) LRC Generation

The Longitudinal Redundancy Check (LRC) field is one byte, containing an 8-bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The receiving device recalculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results.

The LRC is calculated by adding together successive 8-bit bytes in the message, discarding any carries, and then two's complementing the result. The LRC is an 8-bit field, therefore each new addition of a character that would result in a value higher than 255 decimal simply 'rolls over' the field's value through zero. Because there is no ninth bit, the carry is discarded automatically.

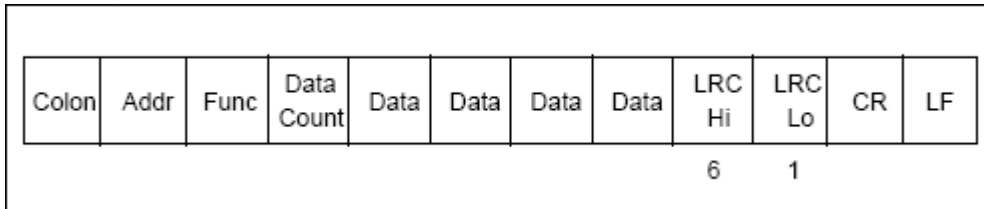
A procedure for generating an LRC is:

1. Add all bytes in the message, excluding the starting 'colon' and ending CRLF. Add them into an 8-bit field, so that carries will be discarded.
2. Subtract the final field value from FF hex (all 1's), to produce the ones-complement.
3. Add 1 to produce the twos-complement.

#### – Placing the LRC into the Message

When the 8-bit LRC (2 ASCII characters) is transmitted in the message, the high-order character will be transmitted first, followed by the low-order character.

For example, if the LRC value is 61 hex (0110 0001):



#### – Example

An example of a C language function performing LRC generation is shown below.

The function takes two arguments:

```
unsigned char *auchMsg ;           // A pointer to the message buffer containing
                                   // binary data to be used for generating the LRC
unsigned short usDataLen ;         // The quantity of bytes in the message buffer.
```

The function returns the LRC as a type unsigned char.

#### – LRC Generation Function

```
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg ;           /* message to calculate LRC upon */
unsigned short usDataLen ;         /* quantity of bytes in message */
{
    unsigned char uchLRC = 0 ;      /* LRC char initialized */
    while (usDataLen--)            /* pass through message buffer */
        uchLRC += *auchMsg++;      /* add buffer byte without carry */
    return ((unsigned char)-((char)uchLRC)); /* return twos complement */
}
```



**(2) CRC Generation**

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a 16-bit register to all 1's. Then a process begins of applying successive 8-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC.

During generation of the CRC, each 8-bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

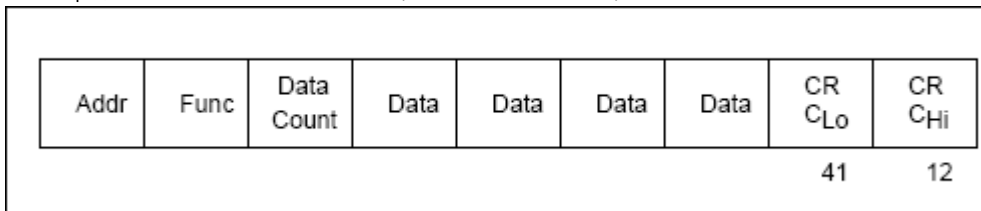
A procedure for generating a CRC is:

1. Load a 16-bit register with FFFF hex (all 1's). Call this the CRC register.
2. Exclusive OR the first 8-bit byte of the message with the low-order byte of the 16-bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), zero-filling the MSB. Extract and examine the LSB.
4. (If the LSB was 0): Repeat Step 3 (another shift). (If the LSB was 1): Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).
5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8-bit byte will have been processed.
6. Repeat Steps 2 through 5 for the next 8-bit byte of the message. Continue doing this until all bytes have been processed.
7. The final contents of the CRC register is the CRC value.
8. When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

**– Placing the CRC into the Message**

When the 16-bit CRC (two 8-bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high-order byte.

For example, if the CRC value is 1241 hex (0001 0010 0100 0001):



**– Example**

An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer.

One array contains all of the 256 possible CRC values for the high byte of the 16-bit CRC field, and the other array contains all of the values for the low byte. Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

**Note** This function performs the swapping of the high/low CRC bytes internally. The bytes are already swapped in the CRC value that is returned from the function. Therefore the CRC value returned from the function can be directly placed into the message for transmission.

The function takes two arguments:

```
unsigned char *puchMsg ;           //A pointer to the message buffer containing
                                   //binary data to be used for generating the CRC
unsigned short usDataLen ;        //The quantity of bytes in the message buffer.
```

The function returns the CRC as a type unsigned short.

**- CRC Generation Function**

```

unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ;          /* message to calculate CRC upon */
unsigned short usDataLen ;       /* quantity of bytes in message */
{
    unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
    unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
    unsigned ulIndex ;           /* will index into CRC lookup table */
    while (usDataLen—)         /* pass through message buffer */
    {
        ulIndex = uchCRCHi ^ *puchMsgg++ ; /* calculate the CRC */
        uchCRCHi = uchCRCLo ^ auchCRCHi[ulIndex] ;
        uchCRCLo = auchCRCLo[ulIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}

```

**- High-Order Byte Table**

```

/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40
} ;

```

**- Low-Order Byte Table**

```

/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;

```