Modbus 소개

작성일: Apr 25, 2013 | 0 Ratings | **0.00** out of 5

개요

Modbus는 어플리케이션 계층 프로토콜로서 마스터/슬레이브 또는 요청/응답 아키텍처를 기반으로 합니다. Modbus는 1979년에 Modicon이 발표하였고 주로 산업용 어플리케이션에 사용되고 있습니다. 이 튜토리얼에서는 Modbus 어플리케이션 계층 중 하이레벨 기능에 대한 개요를 알아보며, 시리얼 구현과 TCP/IP 스펙을 위한 스펙에 대해 집중적으로 살펴봅니다.

보다 자세한 정보는 www. modbus. org에서 확인할 수 있습니다.

목차

Modbus 프로토콜

소개

Modbus 프로토콜은 마스터/슬레이브 아키텍처를 따르며, 이 아키텍처에서 마스터는 슬레이브로부터 데이터를 요청합니다. 마스터는 슬레이브가 몇 가지 동작을 수행하도록 요청할 수도 있습니다. 마스터는 수행해야 할 트랜잭션의 유형을 나타내는 함수 코드를 송신하여 프로세스를 개시합니다. Modbus 프로토콜이 수행하는 트래잭션은 다른 하드웨어에 접근 요청하기 위해 컨트롤러가 사용하는 프로세스, 다른 디바이스의 요청에 응답하는 방법, 에러가 감지되고 보고되는 방법을 정의합니다. Modbus 프로토콜은 계층에 대한 공통 형식과 메시지 필드의 컨텐츠를 정합니다.

Modbus 네트워크에서 통신을 진행하는 동안, 프로토콜은 각 컨트롤러가 디바이스 주소를 알게 되고, 메시지를 인지하며, 수행해야 할 동작의 유형을 결정하고, 메시지에 포함된 데이터나 기타 정보를 추출하는 방법을 결정하게 됩니다.

컨트롤러는 한 디바이스 즉, 마스터만이 트랜잭션이나 쿼리를 개시하는 마스터/슬레이브 기법을 이용하여 통신합니다. 다른 디바이스 즉, 슬레이브들은 요청 받은 데이터를 마스터에 공급하거나 쿼리에서 요청한 동작을 수행하여 응답합니다. 일반적인 마스터 디바이스에는 호스트 프로세서와 프로그래밍 패널이 포함되어 있습니다. 일반적인 슬레이브에는 프로그래밍 가능한 컨트롤러들이 포함되어 있습니다.

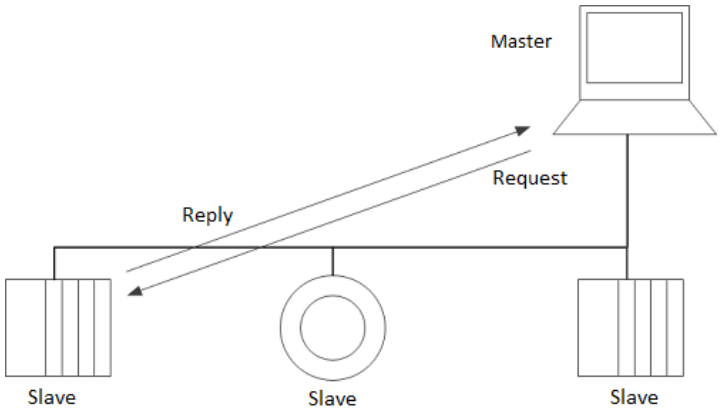

그림 1: 기본 Modbus 네트워크[1]

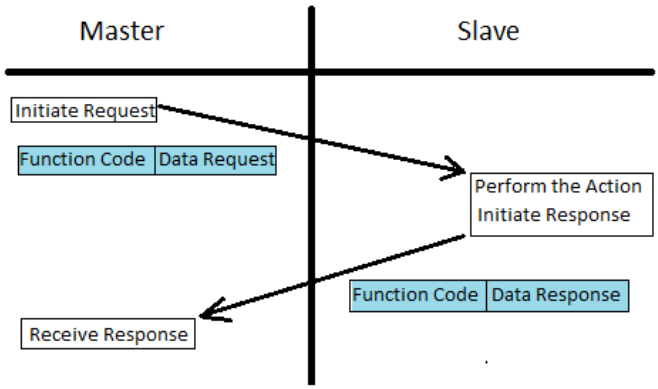

그림 2: 기본 Modbus 트랜잭션[1]

마스터와 슬레이브간에 교환되는 메시지를 프레임이라고 합니다. Modbus 프레임의 유형은 두 가지인 Protocol Data Unit (PDU)과 Application Data Unit (ADU)이 있습니다. PDU 프레임에는 데이터 뒤에 함수 코드가 포함되어 있습니다. 이 함수 코드는 수행해야 할 동작을 나타내고, 데이터는 이 동작을 위해 사용되는 정보를 나타냅니다. ADU 프레임은 추가적인 주소 부분가 있어 약간 더 복잡합니다. 또한 ADU 프레임은 일부 에러 체킹을 제공합니다. ADU 및 PDU 프레임은 빅 엔디언 인코딩을 따릅니다.

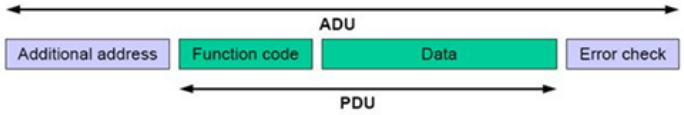

그림 3: Modbus 프레임[1]

Modbus 트랜잭션은 4가지 데이터 유형이 담긴 한 세트를 읽거나 쓰기하여 한 세트의 동작을 항상 수행합니다. 표 1은 Modbus 어플리케이션 계층이 사용하는 네 개의 데이터 형식입니다.

기본 테이블	객체 유형	종류
Discrete Input	단일 비트	읽기만
Coils	단일 비트	읽기-쓰기
Input Registers	16-비트 워드	읽기만
Holding Registers	16-비트 워드	읽기-쓰

표 1: Modbus 데이터 유형[1]

Discrete Input은 읽을 수만 있는 단일 비트 (불리언)를 뜻합니다. 다른 말로, 마스터는 개별 입력에서 읽기 동작만 수행할 수 있습니다. 이런 동작은 **Input Register**에서도 동일합니다. 마스터는 슬레이브의 Input Register만 읽을 수 있습니다. Discrete Input과 Input Register의 차이는 입력 레지스터가 16 비트를 표현하는 반면, Discrete Input은 단일 비트만 될 수 있다는 것입니다. **Coil**은 마스터가 읽고 쓸 수 있는 불리언 데이터 타입을 표현합니다.

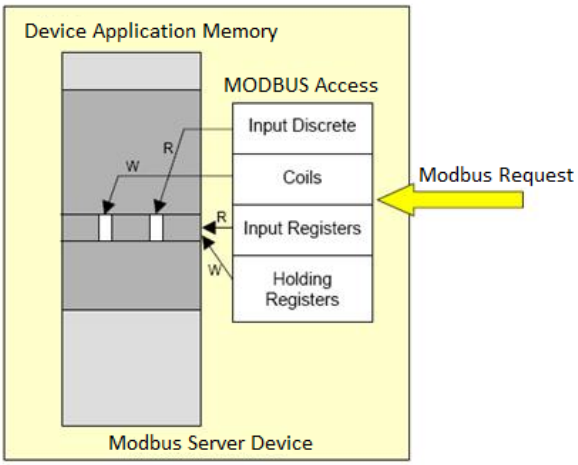

그림 5: 데이터 유형을 이용한 Modus 트랜잭션[1]

완성된 Modbus 트랜잭션

앞에서 언급한 것처럼, 슬레이브가 수행하는 동작의 종류는 함수 코드가 정의합니다. 예를 들어, 마스터가 특정한 Discrete Input을 읽고자 한다면, 바람직한 Discrete Input의 주소 뒤에 0x02의 함수 코드를 전송하게 됩니다. 슬레이브는 0x02를 읽어 마스터가 Discrete Input을 원한다는 것을 알게 됩니다. 슬레이브는 주어진 주소에서 개별 입력을 회수하고 마스터에 다시 답신합니다.

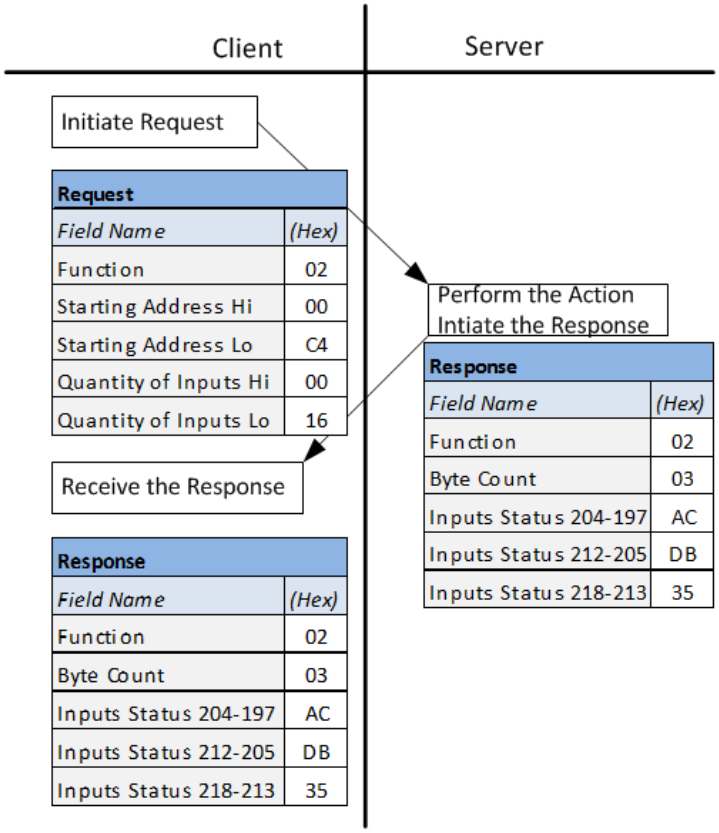

그림 6: 완성된 Modbus 트랜잭션[1]

더 많은 함수 코드

함수 코드의 주 종류로는 세 가지인 Public, User Defined, Reserved가 있습니다. 일반 함수 코드는 modbus- ida.org 커뮤니티에서 검증하고, 공개적으로 문서화했으며, 적합 테스트도 거쳤기 때문에 대부분의 Modbus 디바이스는 일반 함수 코드를 구현합니다. 각각의 일반 함수 코드들은 잘 정의된 함수와 연관되어 있습니다. 일반 함수 코드에 대한 간략한 개요는 표 2에서 확인할 수 있습니다. 일반 함수 코드에 대한 상세 정보는 www. modbus. org에서 확인할 수 있습니다.

				Function Codes			
				Code	Sub Code	(hex)	Section
Data Access	Bit Access	Physical Discrete Inputs	Read Discrete Inputs	02		02	6.2
		Internal Bits or Physical Coils	Read Coils	01		01	6.1
			Write Single Coil	05		05	6.5
			Write Multiple Coils	15		0F	6.11
	16 bits access	Physical Input Registers	Read Input Register	04		04	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03	6.3
			Write Single Register	06		06	6.6
			Write Multiple Registers	16		10	6.12
			Read/Write Multiple Registers	23		17	6.17
			Write Mask Register	22		16	6.16
			Read FIFO Queue	24		18	6.18
	File Record Access		Read File Record	20		14	6.14
			Write File Record	21		15	6.15
Diagnostics			Read Exception Status	07		07	6.7
			Diagnostic	08		08	6.8
			Get Com Event Counter	11	00-18,20	0B	6.9
			Get Com Event Log	12		0C	6.10
			Report Slave ID	17		11	6.13
			Read Device Identification	43	14	2B	6.21
Other			Encapsulated Interface Transport	43	13,14	2B	6.19

표 2: 일반 함수 코드[1]

이전 예에서, 0x02 함수 코드를 이용하여 Discrete Input을 읽었습니다. 이 함수는 일반 함수 코드이기 때문에 함수 코드를 자세히 살펴볼 수 있습니다.

요청		
함수 코드	1 바이트	0x02
시작 주소	2 바이트	0x0000 to 0xFFFF
입력 양	2 바이트	1 to 2000 (0x7D0)

응답		
함수 코드	1 바이트	0x02
바이트 수	1 바이트	N*

입력 상태	N* x 1 바이트	

*나머지가 0 ˙ N = N+1과 다를 경우, N=입력 양 / 8

에러		
에러 코드	1 바이트	0x82

표 3. 상세 함수 코드 예[1]

표 3에서 확인할 수 있듯, 함수 코드는 반드시 시작 주소용 2 바이트와 마스터가 필요한 입력 개수용 2 바이트가 반드시 뒤따라야 합니다. 슬레이브는 반드시 함수 코드와 함께 응답해야 하고 전송된 바이트 수를 표현하는 1 바이트와 Discrete Input 값 후에 응답해야 합니다. 완성된 트랜잭션은 그림 6에서 확인할 수 있습니다.

반면, 사용자가 정의한 코드는 각 Modbus 디바이스에 고유합니다. 이 코드들은 특정 디바이스에서만 사용할 수 있는 특수 세트의 함수들과 함께 묶여있습니다. 자세한 정보는 디바이스 제조업체 매뉴얼에서 자세히 다룹니다.

하드웨어 구현

시리얼 구현

Modbus 어플리케이션 계층이 따를 수 있는 시리얼 모드는 두 가지인 RTU와 ASCII가 있습니다. RTU에서 데이터는 바이너리 형식으로 표현되며, ASCII 모드는 사람이 읽을 수 있는 데이터를 표현합니다. 그림 7과 8은 이 두 모드간의 차이를 보여줍니다.

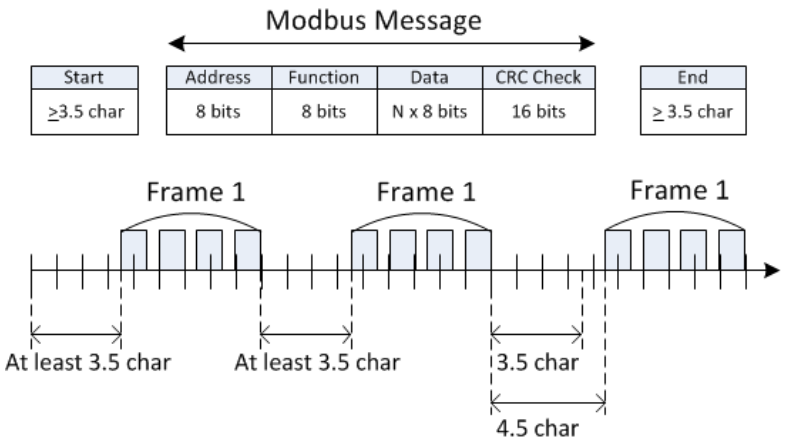

그림 7: Modbus RTU 시리얼 프레임[1]

Start	Address	Function	Data	LRC	End
1 char	2 chars	2 chars	0 up to 2x252 chars(s)	2 Chars	2 chars CR, LF

그림 8: Modbus ASCII 시리얼 프레임[1]

Modbus와 함께 사용하는 가장 일반적인 시리얼 프로토콜은 RS- 232와 RS- 485입니다. 이 프로토콜에 대한 상세 정보는 이 링크에서 확인할 수 있습니다.

TCP 구현

대부분의 TCP 어플리케이션과 마찬가지로 첫 번째로 필요한 것은 마스터와 슬레이브를 연결하는 것입니다. 연결이 이루어지면, 마스터는 슬레이브에 대한 요청을 구축할 수 있습니다. 이 요청에는 MPAB 헤더 뒤의 PDU (위에서 설명한 Modbus 프레임)가 포함되어 있습니다 (그림 9). 그림 10은 MPAB 헤더에 대한 템플릿을 나타냅니다.

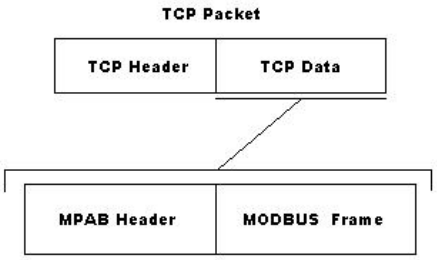

그림 9: Modbus TCP 프레임[1]

	설명	크기	예
MBAP 헤더	Transaction Identifier Hi	1	0x15
	Transaction Identifier Lo	1	0x01
	Protocol Identifier	2	0x0000
	Length	2	0x0006
	Unit Identifier	1	0xFF

그림 10: MBAP 헤더[1]

Transaction Identifier는 패킷이 연관된 Modbus 트랜잭션을 기록하기 위해 사용하는 “TCP Sequence Number”와 같을 수 있습니다. 이 내용이 중요한 이유는 Modbus TCP에서 슬레이브는 동시에 여러 요청들을 처리할 수 있기 때문입니다. 하지만 Modbus Serial에서는 불가능합니다.

Unit Identifier는 일반적으로 Modbus 슬레이브를 다루기 위해 사용됩니다. Modbus TCP를 이용할 때, 슬레이브의 주소는 Modbus TCP의 주소이며 MBAP 헤더의 Unit Identifier는 사용되지 않습니다. 그림 11은 완성된 Modbus TCP 트랜잭션을 보여줍니다.

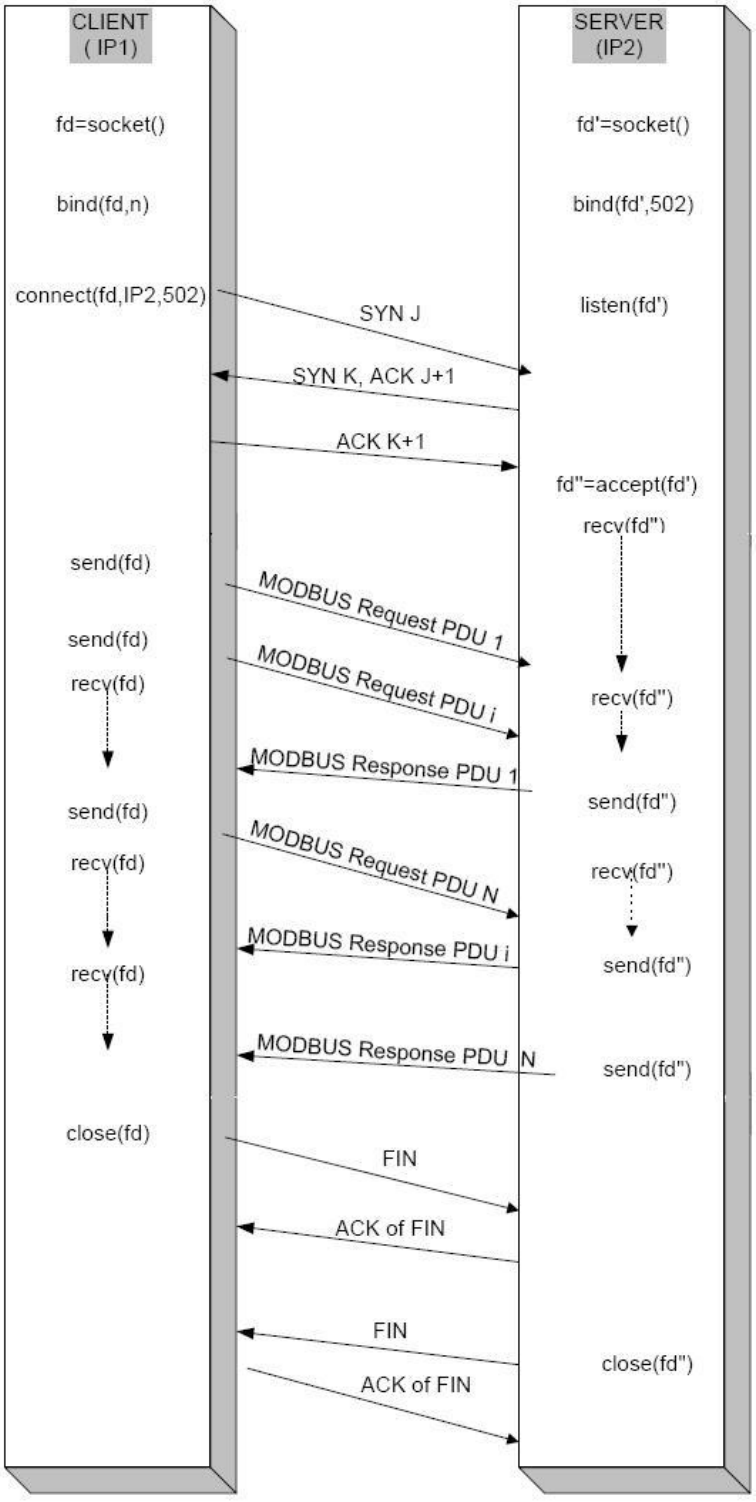

그림 11: 완성된 Modbus TCP 트랜잭션[1]

Modbus와 LabVIEW 이용

Modbus I/O Server

LabVIEW와 Modbus 디바이스의 통신에 선호되는 방법은 LabVIEW Datalogging and Supervisory Control (DSC) Module을 이용하는 것입니다. DSC Module은 I/O Server를 이용하여 마스터와 슬레이브간 모든 프로토콜을 처리합니다. I/O Server는 Shared Variable Engine (SVE)에서 사용되며, SVE가 Modbus 마스터와 슬레이브 둘 다 될 수 있도록 합니다. Modbus I/O 서버는 Real- Time Module에도 포함되어 있기 때문에 CompactRIO는 다른 Modbus 디바이스와 통신하는데 사용할 수 있습니다.

Modbus Master I/O Server

SVE는 Modbus Master I/O Server를 생성하여 Modbus Slave와 통신할 수 있습니다. 일반적인 어플리케이션은 LabVIEW 어플리케이션을 실행하는 컴퓨터가 있어야 하고, SVE가 Modbus Slave로 설정되어 Modbus Slave로 설정된 PLC와 통신해야 합니다. 그림 12는 I/O Server와 SVE가 함께 작동하여 LabVIEW가 Modbus 코일과 PLC의 레지스터에 접근하는 방법을 나타냅니다.

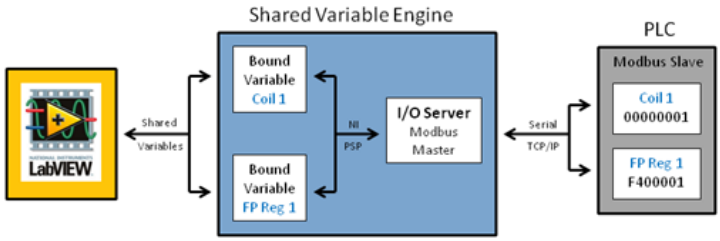

그림 12: LabVIEW와 Modbus 네트워크

SVE는 Modbus Master I/O 서버를 이용하여 PLC와 통신합니다. 이 PLC는 시리얼 또는 이더넷을 통해 Modbus Standard를 이용합니다. I/O Server는 로우레벨 통신 프로토콜을 처리하여 보다 빠른 개발 시간을 제공합니다. SVE는 NI Publish- Subscribe Protocol (NI-PSP) 을 이용하여 각 Modbus 주소를 위한 PSP URL을 제공합니다. SPS URL을 이용하면, Shared Variables는 앨리어싱을 활성화하여 Modbus 주소에 묶일 수 있습니다. Shared Variables가 SVE에 배포되고 Modbus 주소와 통신하고 난 후, LabVIEW는 Shared Variable에 쉽게 읽고 쓸 수 있습니다 (그림 13).

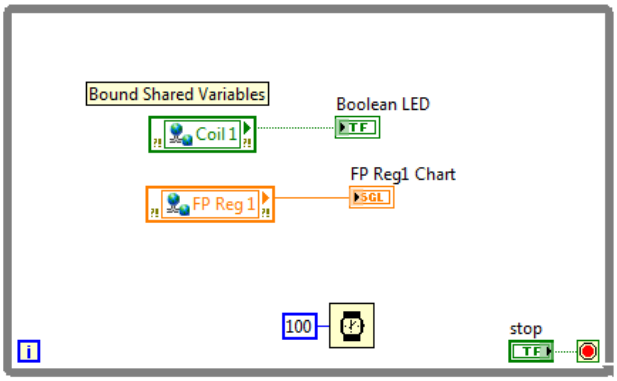

그림 13: LabVIEW에서 묶인 Shared Variable에서 읽기

Shard Variable Engine이 Shared Variable을 Modbus 주소에 묶게 되면, 블록 다이어그램에서 Shared Variable을 직접 읽고 쓰기함으로써 읽고 쓰기가 간단해집니다. 연결을 개시하고 닫는데는 추가적으로 코드가 필요하지 않습니다. 묶인 Shared Variable은 네트워크에 출판할 수 있기 때문에 LabVIEW가 설치된 컴퓨터는 묶인 Shared Variable에 읽고 쓸 수 있습니다. Shared Variable Engine을 위한 Modbus I/O Server 설정 방법에 대한 보다 상세한 단계들은 Developer Zone: Modbus를 이용한 PLC에 연결에서 확인할 수 있습니다.

Modbus Slave I/O Server

SVE는 Modbus Slave I/O Server를 이용하여 Modbus Master 디바이스와 통신이 가능합니다. 일반 데스크탑 컴퓨터에서 LabVIEW 어플리케이션을 실행할 어플리케이션을 제작할 때, 일반적으로 Modbus Slave I/O Server를 이용하지 않습니다. 보다 일반적인 어플리케이션에서는 CompactRIO와 같은 리얼타임 타겟이 CompactRIO의 SVE에 설정된 Modbus Slave I/O Server와 함께 슬레이브로 설정되어 있습니다 (그림 14).

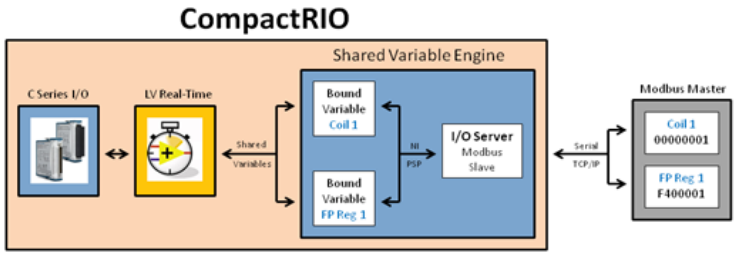

그림 14: CompactRIO에 Modbus Slave I/O Server 이용

CompactRIO 타겟은 PLC를 대체하여 원격으로 배포할 수 있고 Modbus 어플리케이션에 통합이 가능합니다. SVE와 Modbus I/O 서버 지원은 반드시 CompactRIO 리얼타임 컨트롤러에 설치되어 있어야 합니다. LabVIEW Real- Time은 C 시리즈 모듈의 물리 I/O에 접속한 후 값을 Shared Variable에 쓰기 위해 사용할 수 있습니다. 다음으로 Shared Variable은 NI-PSP 및 Modbus Slave I/O Server를 이용하여 Modbus 주소에 묶입니다. LabVIEW Real- Time은 그림 16에서 사용한 방식과 동일하게 Shared Variable을 통해 Modbus 주소와 연동합니다. 이 방식에서 많은 CompactRIO 타겟들이 시리얼이나 이더넷을 통해 단일 Modbus와 통신하는 Modbus Slave로 배포될 수 있습니다.

NI Modbus 라이브러리

LabVIEW용 Modbus 라이브러리는 무료로 다운로드할 수 있습니다. NI는 LabVIEW Datalogging and Supervisory Control (DSC) Module 또는 Real- Time Module에서 제공하는 Modbus I/O Server를 이용할 것을 권장합니다. 이 모듈들은 Modbus TCP와 시리얼에 사용하기 편리한 Shared Variable을 제공합니다. NI Modbus 라이브러리를 사용하기로 결정했다면, 이 라이브러리와 함께 제공되는 예제들을 살펴보는 것이 어플리케이션 개발을 시작하는 데 큰 도움이 됩니다. Modbus 프로토콜의 시리얼 및 TCP 구현을 위한 마스터와 슬레이브를 위한 예제들이 있습니다..

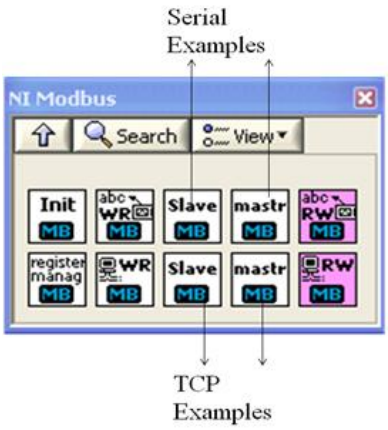

그림 15. Modbus 라이브러리 예제

[1]리소스

이 문서에서 사용된 주 리소스들은 www. modbus. org 의 자료를 사용하였습니다.

관련 링크

- NI Developer Zone: LabVIEW에서 I/O Server 를 이용하는 방법
- NI Developer Zone: Modubs를 이용하여 PLC를 LabVIEW에 연결하는 방법