**1.Modbus**

MODBUS는 1979년 Modicon에서 발표한 데이터 통신 프로토콜로, 산업용 전자 장치를 연결하는 표준 방식입니다. 공개 배포되며 로열티가 없어 인기가 높고, 시리얼 통신 및 이더넷을 사용하여 여러 장치와 연결할 수 있습니다. 주로 SCADA 시스템에서 사용되며, 단일 비트 출력을 코일, 입력을 discrete input이라고 합니다. 2004년부터 Modbus Organization이 관리하며, 호환 장치 사용을 촉진하고 있습니다.

**2. Modbus 객체 유형**

|  |  |  |  |
| --- | --- | --- | --- |
| 객체 유형 | 접근 | 크기 | 주소 공간 |
| Coil | Read-Write | 1bit | 00001-09999 |
| Discrete input | Read-only | 1bit | 10001-19999 |
| Input register | Read-only | 16bit | 30001-39999 |
| Holding register | Read-Write | 16bit | 40001-49999 |

**3. 프로토콜 버전**

MODBUS 프로토콜은 여러 변형이 있으며, 다음과 같은 주요 버전을 지원

1. **MODBUS RTU**:
   * 시리얼 통신에서 사용되며, 데이터의 컴팩트한 바이너리 표현과 CRC

체크섬을 사용합니다.

* + 연속적으로 문자 간 전송되어야 하며, idle 동안 프레임으로 구분됩니다.

1. **MODBUS ASCII**:
   * 시리얼 통신에 ASCII 문자를 사용하며, LRC 체크섬을 사용합니다.
   * 메시지는 콜론(":")으로 시작하고 개행(CR/LF)으로 끝납니다.
2. **MODBUS TCP/IP**:
   * TCP/IP 네트워크를 사용하며, 포트 502로 접속합니다. 체크섬 계산이

필요 없습니다.

1. **MODBUS over UDP**:
   * TCP의 오버헤드를 제거하고 UDP를 사용하는 실험적 변형입니다.
2. **MODBUS Plus**:
   * Schneider Electric 소유로, 여러 클라이언트 간 P2P 통신을 지원합니다.

고속 전송을 위해 전용 하드웨어가 필요합니다.

1. **Pemex MODBUS**:
   * 공정 제어용으로 Pemex 석유 및 가스 회사를 위해 설계된 MODBUS

확장입니다.

1. **Enron MODBUS**:
   * 32비트 정수와 부동 소수점 변수를 지원하는 MODBUS 확장으로,

API 산업 표준을 준수합니다.

각 변형은 데이터 모델과 함수 호출은 동일하지만, 프레임 형식이 다르므로 상호 운용할 수 없습니다.

**4. 통신 및 장치**

1. **장치 역할**:
   * **MODBUS RTU, MODBUS ASCII, MODBUS Plus**: 클라이언트로 지정된 노드만 명령을 시작할 수 있으며, 나머지 장치는 서버로 요청에 응답합니다.
   * **MODBUS TCP**: 모든 장치가 클라이언트 역할을 할 수 있으며, 명령을 보낼 수 있습니다.
2. **프로토콜 지원**:
   * MODBUS는 다양한 모뎀과 게이트웨이에서 지원되며, 일부는 이 프로토콜을 위해 특별히 설계되었습니다.
   * 다른 구현에서는 ISM 대역, SMS, GPRS와 같은 유선 및 무선 통신도

사용됩니다.

1. **무선 네트워크**:
   * 일반적으로 mesh 네트워크 설계를 사용할 수 있으며, 설계자는 높은 대기

시간과 타이밍 문제를 해결해야 합니다.

MODBUS는 간단하고 복사하기 쉬운 프로토콜로, 다양한 통신 방식에서 활용됩니다.

**5.명령**

1. **값 변경: Coil과 Holding 레지스터에 저장된 값을 변경합니다.**
2. **데이터 읽기: Discrete Input이나 Coil에서 데이터를 읽습니다.**
3. **값 전송: Coil과 Holding 레지스터에 포함된 하나 이상의 값을**

**다시 보내도록 장치에 명령합니다.**

**주요 사항:**

* **주소 지정: MODBUS 명령에는 해당 장치의 MODBUS 주소(1 ~ 247)가 포함됩니다. 주소가 지정된 장치만 명령에 응답하고 작동하며, 노드 0에 보내는 특수 브로드**

**캐스트 명령은 응답하지 않습니다.**

* **체크 섬 정보: 모든 MODBUS 명령에는 수신자가 전송 오류를 감지할 수 있도록**

**하는 체크 섬 정보가 포함되어 있습니다.**

**6.1 Modbus RTU**

|  |  |  |
| --- | --- | --- |
| **이름** | **길이** | **기능** |
| **Start** | **3.5\*8** | **최소3\*1/2 문자 시간(28bit)의 침묵** |
| **Address** | **8** | **위치 주소** |
| **Function** | **8** | **‘Read Coil’와 같은 기능** |
| **Data** | **N^8** | **Data+Length 메시지 유형에 따라 채워짐** |
| **CRC** | **16** | **Cyclic Redundancy Check** |
| **End** | **3.5\*8** | **프레임 사이 최소 3\*1/2 문자 시간의 침묵** |

**CRC 계산**

* **CRC-16-ANSI는 CRC-16-IBM이라고도 하고, 일반 16진 대수 다항식은 8005이고 역순 A001**
* **초기 값 : 65535**
* **16진수 프레임 예제 : 01 04 02 FF FF B8 80 (01 ~ FF까지 5 바이트에 대한 CRC-16-ANSI 계산은 최하위 바이트부터 먼저 전송된 80B8 이다.)**

**6.2 MODBUS ASCII 프레임 형식**

**주로 7비트나 8비트 asynchronous serial lines에서 사용된다.**

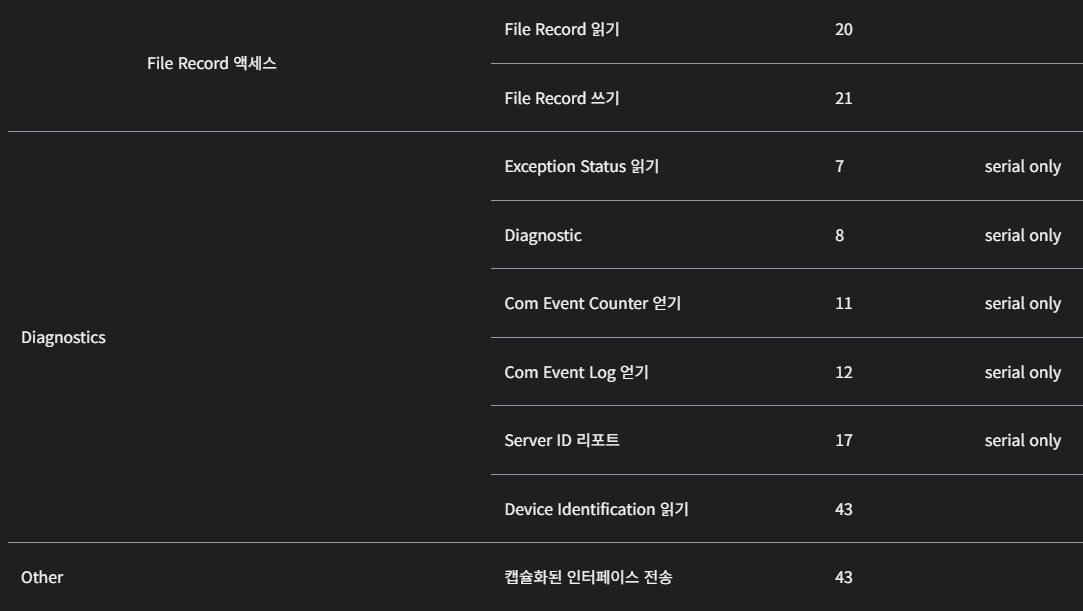
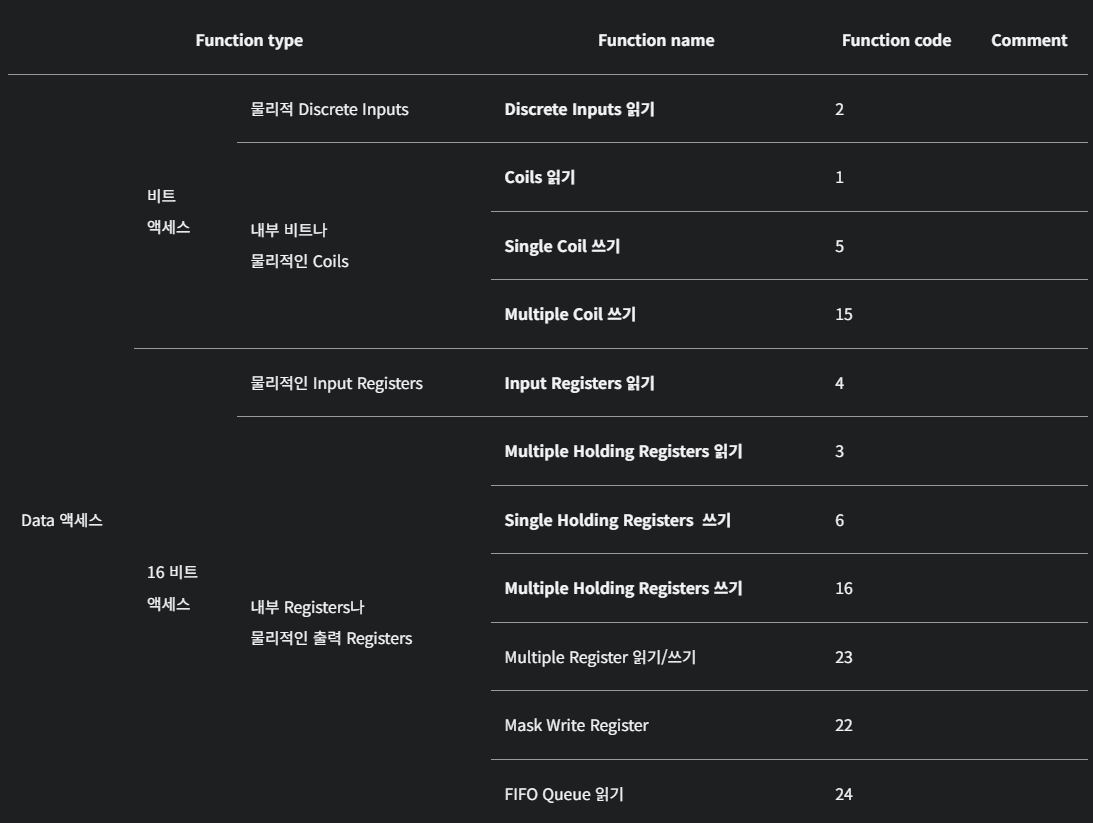
|  |  |  |
| --- | --- | --- |
| **이름** | **길이** | **기능** |
| **Start** | **3.5\*8** | **콜론: (ASCII 값으로 0x3A)** |
| **Address** | **8** | **위치 주소** |
| **Function** | **8** | **‘Read Coil’와 같은 기능** |
| **Data** | **N^8** | **Data+Length 메시지 유형에 따라 채워짐** |
| **CRC** | **16** | **Cyclic Redundancy Check** |
| **End** | **3.5\*8** | **Carriage Return + Line Feed(CR/LF) 쌍 (ASCII 값으로 0x0D, 0x0A)** |

**6.3 Modbus TCP 프레임 형식**

|  |  |  |
| --- | --- | --- |
| **이름** | **길이** | **기능** |
| **Transaction Identifler** | **2** | **서버와 클라이언트의 메시지 동기화를 위해** |
| **Protocol Identifier** | **2** | **Modbus/TCP의 경우 0** |
| **Length Field** | **2** | **이 프레임의 남은 바이트 수** |
| **Unit Identifier** | **1** | **서버 주소(사용하지 않으면 255)** |
| **Function Code** | **1** | **다른 변형과 같은 기능 코드** |
| **Data Bytes** | **N** | **응답이나 명령 데이터** |

**6.4 기능과 명령**

* **Coils : 읽기, 쓰기 가능, 1 비트 (off/on)**
* **Discrete Inputs : 읽기 전용, 1 비트 (off/on)**
* **Input Registers : 측정과 상태 읽기 전용, 16 비트 (0 ~ 65535)**
* **Holding Registers : 읽기, 쓰기 가능한 구성(configuration) 값, 16 비트 (0 ~ 65535)**

****

**7.요청 및 응답 형식**

**7.1 Function Code 1 and 2**

* **요청:**
  + **읽을 첫 번째 Coil/Discrete Input의 주소 (16비트)**
  + **읽을 Coils/Discrete Inputs의 수 (16비트)**
* **정상 응답:**
  + **뒤따르는 Coil/Discrete Input 값의 바이트 수 (8비트)**
  + **Coil/Discrete Input 값 (바이트당 8개의 Coils/Discrete Inputs)**
    - **각 Coil/Discrete Input의 값은 바이너리로, off는 0, on은 1입니다.**
    - **첫 번째 요청된 Coil/Discrete Input은 응답에서 첫 번째 바이트의 최하위 비트에 저장됩니다.**
    - **Coils/Discrete Inputs의 수가 8의 배수가 아닐 경우, 마지막 바이트의 최상위 비트는 0으로 채워집니다.**
* **예시:**
  + **11개의 Coils 요청 시, 2바이트의 값이 필요합니다.**
  + **상태 예: on, off, on, off, off, on, on, on, off, on, on → 응답: 16진수로 02 E5 06 (바이트 수 2, 값 0xE5는 11100101).**
* **최대 읽기 수:**
  + **응답 메시지에서 반환된 바이트 수는 8비트(256)이며, 프로토콜 오버헤드는 5바이트(256-1)입니다.**
  + **따라서 한 번에 최대 2008(251 \* 8)개의 Discrete Inputs 또는 Coils를 읽을 수 있습니다.**

**Function Code 5 (Force/Write Single Coil)**

* **요청:**
  + **Coil 주소 (16비트)**
  + **Force/Write 값: off의 경우 0, on의 경우 65,280 (16진수로 FF00)**
* **정상 응답:**
  + **요청과 동일한 메시지를 반환합니다.**

**7.3 Function Code 15 (Force/Write Multiple Coils)**

* **요청:**
  + **첫 번째 Coil의 주소 (16비트)**
  + **Force/Write할 Coil의 수 (16비트)**
  + **뒤 따르는 Coil 값의 바이트 수 (8비트)**
  + **Coil 값 (바이트당 8개의 Coil 값): 각 Coil의 값은 이진수로 표현되며, off는 0, on은 1입니다.**
  + **요청에서 첫 번째 Coil은 첫 번째 바이트의 최하위 비트에 저장됩니다. Coil 수가 8의 배수가 아닐 경우, 마지막 바이트의 최상위 비트는 0으로 채워야 합니다.**
* **정상 응답:**
  + **첫 번째 Coil의 주소 (16비트)**
  + **Coil의 수 (16비트)**

**7.4 Function Code 4 (Read Input Registers) 및 3 (Read Holding Registers)**

* **요청:**
  + **읽을 첫 번째 Register의 주소 (16비트)**
  + **읽을 Registers의 수 (16비트)**
* **정상 응답:**
  + **뒤 따르는 Register 값의 바이트 수 (8비트)**
  + **Register 값 (Register당 16비트)**
* **최대 요청 가능 Register 수:**
  + **RTU 형식: 최대 125개의 Register**
  + **TCP 형식: 최대 123개의 Register**

**7.5 Function Code 6 (Preset/Write Single Holding Register)**

* **요청:**
  + **Preset/Write할 Holding Register의 주소 (16비트)**
  + **Holding Register의 새로운 값 (16비트)**
* **정상 응답:**
  + **요청과 동일**

**7.6 Function Code 16 (Preset/Write Multiple Holding Registers)**

* **요청:**
  + **Preset/Write할 첫 번째 Holding Register의 주소 (16비트)**
  + **Preset/Write할 Holding Register의 수 (16비트)**
  + **뒤 따르는 Holding Register 값의 수 (8비트)**
  + **Holding Registers의 새로운 값 (Holding Register당 16비트)**
* **정상 응답:**
  + **Preset/Written한 Holding Register의 첫 번째 주소 (16비트)**
  + **Preset/Written한 Holding Registers의 수 (16비트)**
* **최대 요청 가능 Holding Register 수:**
  + **최대 123개의 Registers**

1. **Entity 번호와 주소**
2. **주소 지정 규칙:**
   * **각 데이터 항목은 0에서 65535까지 주소가 지정됩니다.**
   * **MODBUS 데이터 모델은 4개의 블록으로 구성되며, 각 요소는 1에서 n까지 번호가 부여됩니다.**
3. **엔티티 번호와 주소 구분:**
   * **엔티티 번호: 설명 테이블 내에서 엔티티 유형과 위치를 결합합니다.**
   * **엔티티 주소: MODBUS 프레임의 데이터 부분에서 16비트 값으로 0~65535 범위의 시작 주소입니다.**
4. **엔티티 번호 규칙:**
   * **Coil 번호: 0으로 시작, 범위 00001 ~ 09999**
   * **Discrete Input 번호: 1로 시작, 범위 10001 ~ 19999**
   * **Holding Register 번호: 4로 시작, 범위 40001 ~ 49999**
5. **주소 변환:**
   * **엔티티 위치(1 ~ 9999)는 1을 빼서 0부터 시작하는 주소(0 ~ 9998)로 변환됩니다.**
   * **예: Holding Registers 40001을 읽으려면 주소 0을 사용합니다.**
6. **확장된 범위:**
   * **각 엔티티의 주소를 9999개로 제한하던 기존 규칙이 65536으로 확장되었습니다.**
   * **확장된 범위:**
     + **Coil: 000001 ~ 065536**
     + **Discrete Input: 100001 ~ 165536**
     + **Input Register: 300001 ~ 365536**
     + **Holding Register: 400001 ~ 465536**
   * **모든 번호는 6자리 숫자를 사용하여 혼동을 피해야 합니다.**
7. **16진수 주소 사용:**
   * **16진수 값을 사용하여 주소를 기록할 수 있습니다:**
     + **Coil: 0x0000 ~ 0xFFFF**
     + **Discrete Input: 1x0000 ~ 1xFFFF**
     + **Input Register: 3x0000 ~ 3xFFFF**
     + **Holding Register: 4x0000 ~ 4xFFFF**

**JBUS 매핑**

* **JBUS는 MODBUS와 밀접하게 관련된 프로토콜로, 1985년에 정의되었습니다.**
* **JBUS는 MODBUS의 기능 코드를 지원하지만, 엔티티 번호와 주소가 다르게**

**매핑됩니다.**

**숫자 0(주소 0)은 지원되지 않으며, 해당 번호와 주소에서 서버는 null 값을 반환하거나 오류를 발생시킵니다.**

**MODBUS의 Holding Register #40010은 Holding Register #9로, JBUS에서는 주소 9로 표현됩니다.**

**10. 구현**

1. **IEEE Floating-Point 수:**
   * **32비트 부동 소수점 숫자로, 일반적으로 과학적 계산 및 정밀한 수치**

**표현에 사용됩니다.**

1. **32-bit 정수:**
   * **32비트 정수형 데이터로, 다양한 수치 데이터를 표현할 수 있습니다.**
2. **8-bit 데이터:**
   * **8비트 크기의 데이터로, 작은 정수나 상태 정보를 나타내는 데 사용됩니다.**
3. **혼합된 데이터 유형:**
   * **다양한 데이터 유형이 혼합되어 사용될 수 있으며, 특정 응용 프로그램에**

**맞춰 조합할 수 있습니다.**

1. **정수 Bit Fields:**
   * **여러 개의 비트로 구성된 필드로, 특정 플래그나 상태 정보를 표현하는 데**

**유용합니다.**

**데이터 변환 및 승수**

* **정수 to/from 변경하는 승수:**
  + **데이터를 변환할 때 사용할 수 있는 승수로, 예를 들어 10, 100, 1000, 256 등을 사용하여 데이터를 스케일링 할 수 있습니다.**

**프로토콜 확장**

1. **16-bit 서버 주소:**
   * **각 서버에 대한 고유한 16비트 주소를 사용하여 통신을 관리합니다.**
2. **32-bit 데이터 크기:**
   * **각 주소가 32비트 데이터 크기를 반환하며, 이는 더 큰 데이터 블록을**

**처리할 수 있게 합니다.**

1. **Word-swapped 데이터:**
   * **데이터의 바이트 순서를 변경하여, 서로 다른 시스템 간의 호환성을 높이는 방식입니다. 이 방법은 일반적으로 데이터가 전송되는 형식에 따라 다르게**

**적용됩니다.**

**11. Modbus Plus**

* **독점 사양: MODBUS Plus는 Schneider Electric의 독점 사양으로, 특허는 없지만 공개되지 않은 형태로 개발되었습니다.**
* **맞춤형 칩셋: 일반적으로 Schneider Electric의 파트너이게만 제공되는 맞춤형 칩셋을 사용하여 구현됩니다.**
* **토큰 전달 방식: 이 프로토콜은 데이터 전송을 위해 토큰 전달 방식을 사용하여, 여러 장치 간의 효율적인 통신을 지원합니다.**