# lamaPLC Communication: Modbus

**Not to be confused with M-Bus!**

Modbus is a data communications protocol first released by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). It has become a standard communication protocol and is now a widely used way to connect industrial electronic devices.

Modbus is widely used in industrial settings because it is openly published and free of royalties. It was designed for industrial use, is easier to deploy and maintain than other standards, and imposes few restrictions on the data format.

The Modbus protocol uses serial character communication lines, Ethernet, or the Internet protocol suite as a transport layer. Modbus supports communication with multiple devices connected to the same cable or Ethernet network. For example, a device that measures temperature and another that measures humidity can be connected to the same cable, both transmitting data to the same computer via Modbus.

Modbus is commonly used to connect a plant or system supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many data types are named based on industrial control of factory devices, such as ladder logic because of its role in operating relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or contact.

The development and updates of the Modbus protocols have been overseen by the Modbus Organization since April 2004, when Schneider Electric transferred rights to that organization. The Modbus Organization is a group of users and suppliers of Modbus-compatible devices that support the ongoing use of the technology. Modbus Organization, Inc. is a trade association dedicated to promoting and developing the Modbus protocol.

## Modbus TCP

Modbus TCP/IP or Modbus TCP – a Modbus variant used for communications over TCP/IP networks, connecting over port 502. It does not require a checksum calculation, as lower layers already provide checksum protection.

Important: Modbus over TCP/IP, Modbus over TCP, or Modbus RTU/IP – a variant that differs from Modbus TCP in that a checksum is included in the payload, as with Modbus RTU.

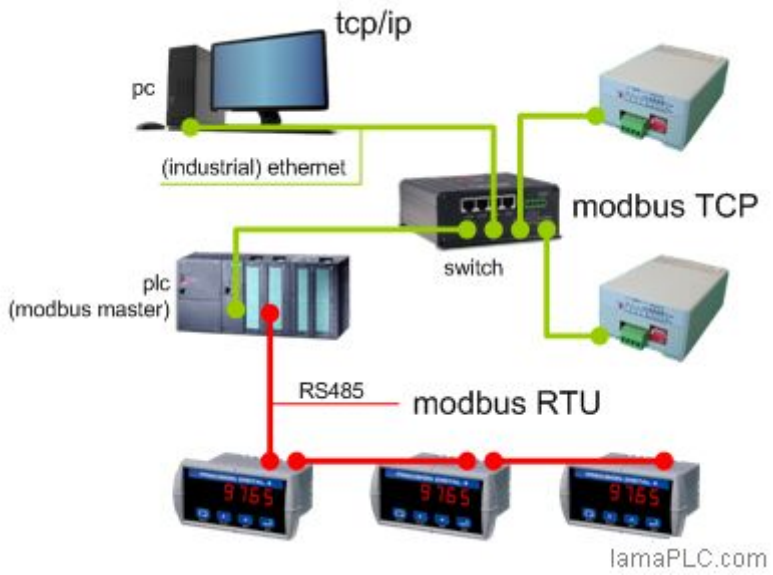
# Modbus RTU

Modbus RTU (*Remote Terminal Unit*) – used in serial (typically **RS485 2W-cabling** or **RS-232**) communication, and is the most common implementation available for Modbus. Modbus RTU uses a compact binary representation of data for protocol communication.

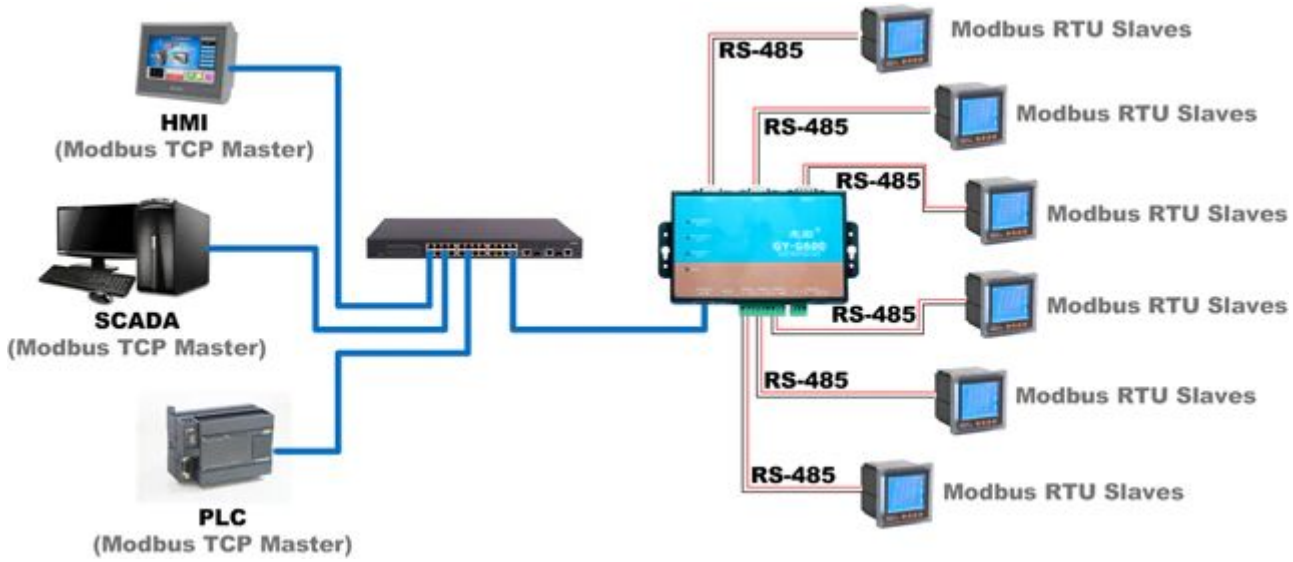
The RTU format uses a cyclic redundancy check (CRC) to verify data integrity. A Modbus RTU message must be transmitted continuously without delays between characters. Modbus messages are separated by idle (silent) periods.

# Limitations

- Since Modbus was created in the late 1970s to communicate with programmable logic controllers, its data types are limited to those understood by PLCs at that time. Large binary objects are not supported.
- There is no standard way for a node to obtain the description of a data object, such as learning that a register value indicates a temperature between 30 and 175 degrees.
- Since Modbus is a client/server (formerly master/slave) protocol, field devices cannot send data through the event handler mechanism, except over Ethernet TCP/IP, known as open-mbus. Instead, the client node must regularly poll each device and check for data changes. This approach consumes bandwidth and network time, which can be costly in applications with limited bandwidth, such as low-bit-rate radio links.
- Modbus is limited to addressing 247 devices on a single data link, which restricts the number of field devices that can be connected to a parent station (again, Ethernet TCP/IP is an exception).
- Modbus protocol itself provides no security against unauthorized commands or interception of data
- **[Addressing]** Originally, valid address ranges for Modbus were 0 to 9999 for each of the listed register types. The current specification now allows ranges from 0 to 65,535.
- When using extended register referencing, all register references must be exactly six digits.
- **[Data Endianness]** Multiregister data, like a single-precision floating-point value, can be easily transferred in Modbus by splitting the data across two registers. Because this is not specified by the standard, the endianness (or byte order) of this split is not defined. Although each unsigned word must be sent in network (big-endian) byte order to comply with the standard, many devices reverse the byte order for multibyte data.
- **[Strings]** Strings can be easily stored in Modbus registers. For simplicity, some implementations require string lengths to be multiples of two, with any extra space filled with null characters. Byte order can also vary in string interactions. The string format may or may not include a NULL as the final character.
- **[Monomaster]** The original Modbus assumes either a monomaster network or a point-to-point connection. In both scenarios, a master and at least one slave are necessary for communication.




- **[Multimaster]** Several masters can be present on the Modbus network simultaneously, but in this case, we must include a multimaster gateway in the communication network, and the masters can only be connected over Modbus TCP.




# Modbus Object Types and Addresses

| Object type | Access | Size | Original address space | Extended addressing* |
|---|---|---|---|---|
| **Coil** | Read-write | 1 bit | 00001 – 09999 | 000001-065535 |
| **Discrete input** | Read-only | 1 bit | 10001 – 19999 | 100001-165535 |
| **Input register** | Read-only | word (16 bits) | 30001 – 39999 | 300001-365535 |
| **Holding register** | Read-write | word (16 bits) | 40001 – 49999 | 400001-465535 |

*: for example, with Simatic

# Modbus Function Codes

The Modbus protocol defines several function codes for accessing registers. It specifies four different data blocks, and the addresses or register numbers in each overlap. Therefore, fully identifying a piece of data requires both the address (or register number) and the function code (or register type).

Most manufacturers only implement the “common” function codes, so you should always verify which codes are compatible with the specific equipment.

| Function Code | Register Type | frequency |
|---|---|---|
| **1** | Read Coil | common |
| **2** | Read Discrete Input | common |
| **3** | Read Holding Registers | common |
| **4** | Read Input Registers | common |
| **5** | Write Single Coil | common |
| **6** | Write Single Holding Register | common |
| **15** | Write Multiple Coils | common |
| **16** | Write Multiple Holding Registers | rare |
| **21** | Write File record | rare |
| **22** | Mask Write Register | rare |
| **23** | Read/Write Multiple Registers | rare |
| **24** | Read FIFO queue | rare |
| **8** | Diagnostic | rare |
| **11** | Get Com event counter | rare |
| **12** | Get Com Event Log | rare |
| **17** | Report Slave ID | rare |
| **43** | Read device Identification | rare |
| **43** | CANopen General Reference | rare |

# Exception codes

The exception codes as explained in the Modbus specification are:

| Exception Code | Name | Meaning |
|---|---|---|
| **01 (01 hex)** | Illegal Function | The function code received in the query is not a valid action for the slave. This could be because the function code is only supported on newer devices and was not implemented on the selected unit. It might also mean that the slave is in an incorrect state to handle this type of request, for example, if it is unconfigured and asked to return register values. If a Poll Program Complete command was issued, this code indicates that no program function was executed beforehand. |
| **02 (02 hex)** | Illegal Data Address | The data address received in the query is not a valid address for the slave. Specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 will succeed, but a request with offset 96 and length 5 will generate exception 02. |

| Exception Code | Name | Meaning |
|---|---|---|
| **03 (03 hex)** | Illegal Data Value | A value found in the query data field is not an acceptable value for the slave. This suggests a fault in the structure of the rest of a complex request, such as an incorrect implied length. It specifically does NOT mean that a data item submitted for storage in a register has a value outside what the application program expects, since the MODBUS protocol does not recognize the significance of any specific value in any register. |
| **04 (04 hex)** | Slave Device Failure | An unrecoverable error occurred while the slave was attempting to perform the requested action. |
| **05 (05 hex)** | Acknowledge | |

The slave has accepted the request and is processing it, but it will take a long time to complete. This response prevents a timeout error from occurring in the master. The master can then send a Poll Program Complete message to check if processing is finished. |

| | | |
|---|---|---|
| **06 (06 hex)** | Slave Device Busy | Specialized use with programming commands.<br>The slave is processing a long-duration program command. The master should retransmit the message later when the slave is free. |
| **07 (07 hex)** | Negative Acknowledge | The slave cannot execute the program function received in the query. This code is sent back for an unsuccessful programming request using function code 13 or 14 decimal. The master should request diagnostic or error information from the slave. |
| **08 (08 hex)** | Memory Parity Error | Specialized use with function codes 20 and 21 and reference type 6 to indicate that the extended file area failed a consistency check.<br>The slave attempted to read extended memory or record file but detected a parity error in memory. The master can retry the request, but service might be needed on the slave device. |
| **10 (0A hex)** | Gateway Path Unavailable | Specialized use with gateways indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually, this means the gateway is misconfigured or overloaded. |
| **11 (0B hex)** | Gateway Target Device Failed to Respond | Specialized use with gateways indicates that no response was received from the target device. Usually, it means that the device is not on the network. |

# Sources

Wikipedia (here)
MODBUS over Serial Line Specification & Implementation guide V1.0 here

# Modbus topics on lamaPLC

| Page | Date | Tags |
|---|---|---|
| • Eastron Modbus maps | 2026/04/23 21:51 | modbus, modbus rtu, eastron, modbus map, mid |

| | | |
|---|---|---|
| • lamaLib: #temp | 2026/04/23 21:52 | tia, scl, lamalibsimatic, source code, energy meter, modbus, register, word |
| • lamaLib: energyMeterToModbusRegs | 2026/04/23 21:52 | tia, scl, lamalibsimatic, source code, energy meter, modbus, register, word |
| • lamaPLC Communication: Modbus | 2026/04/23 21:51 | modbus, communication, bus, modicon, standard, rtu, tcp, multimaster, coil, register |
| • lamaPLC: B+G E-Tech DS100 Energy Meter with Modbus | 2026/06/05 15:59 | communication, modbus, b g, e-tech, ds100, energy meter, em |
| • lamaPLC: Communication with Eastron Smart X96 | 2026/06/05 15:59 | communication, modbus, energy meter, em, eastron, smart, x96 |
| • lamaPLC: DM56A04 / DM36B06 digital tube display with Modbus Communication | 2026/02/14 18:25 | dm56a04, dm36b06, eletechsup, 7-segment, display, modbus, rtu, modbus rtu, arduino |
| • LamaPLC: Eastron SDM 230 with Modbus Communication | 2026/06/05 15:50 | modbus, modbus rtu, eastron, modbus map, mid, sdm 230, sdm, arduino, code |
| • LamaPLC: Eastron SDM 630 Energy Meter with Modbus communication | 2026/06/05 15:50 | modbus, modbus rtu, eastron, modbus map, mid, sdm, sdm 630, arduino, code |
| • lamaPLC: PTA8C04 4-channel PT100 Modbus Modul | 2026/02/14 18:42 | pta8c04, sensor, modbus, rtu, rs-485, communication, platine, um72 |
| • lamaPLC: RP2040_ETH_Modul: Modbus TCP example | 2026/05/12 16:20 | code, micropython, 2026, rp2040 eth, modbus, test |
| • lamaPLC: RP2040_ETH_Modul: Modbus TCP sniffer | 2026/05/12 16:20 | code, micropython, 2026, rp2040 eth, modbus, sniffer |
| • LamaPLC: S7-1500 and Metrawatt EM2389 Modbus TCP communication | 2026/04/23 21:52 | simatic, s7, modbus, communication, metrawatt, em2389, source code, scl, mid |
| • LamaPLC: S7-1500 and Sicam Q200 Modbus TCP communication | 2026/04/23 21:52 | simatic, s7, modbus, tia portal, communication, sicam, q200, sicam q200, source code, scl, class a |
| • lamaPLC: S7-1500 and UICPAL Temp.humi.sensor Modbus TCP communication | 2023/06/19 23:24 | bus, communication, s7, simatic, s7 1500, s7 1200, scl, uicpal, temperature, humidity, modbus, example, download, tia portal |
| • lamaPLC: TM1650 7-Segment Display with I²C like or Modbus Communication | 2026/02/14 18:26 | tm1650, stc8g, tp8485e, hyuduo5x1b64edtk1244, 7-segment, display, modbus, rtu, modbus rtu, arduino |
| • lamaPLC: TTL to RS485 Module | 2026/02/14 23:49 | modbus, rtu, modbus rtu, hw-097, rs-485, max485 |
| • LamaPLC: UICPAL Temp.humi.sensor | 2023/06/25 00:43 | simatic, s7, modbus, communication, temperature, humidity, sensor |
| • LamaPLC: XTM35SC Energy meter with Modbus communication | 2026/06/05 15:59 | xtm35sc, modbus, modbus rtu, measuring, power, communication, current meter, voltmeter |
| • lamaPLC: YR-3180 - Weight sensor module with UART or Modbus communication | 2026/02/15 00:00 | communication, modbus, rtu, sensor, weight, yr-3180, hx710b, arduino, ttl, rs-485 |
| • Modbus for Grundfos pumps | 2026/04/23 21:51 | modbus, modbus tcp, modbus rtu, grundfos |
| • NT18B07: 7 Kanal RS485 Temperatur Sensor with Modbus RTU | 2026/02/14 18:49 | nt18b07, sensor, modbus, rtu, rs-485, communication, platine |

| | | |
|---|---|---|
| • Simatic Modbus S7 error- and statuscodes | 2026/04/23 21:52 | communication, bus, modbus, error, modbus error code, 7000, 7001, 7002, 7003, 7004, 7005, 7006, 80a1, simatic, s7, siemens, tia |
| • Waveshare | 2026/04/23 21:52 | waveshare, converter, modbus, modbus rtu, modbus tcp, communication |
| • XTM35SC current / voltage meter | 2026/04/23 21:52 | xtm35sc, modbus, modbus rtu, measuring, power, communication, current meter, voltmeter |

modbus, communication, bus, Modicon, standard, RTU, TCP, Multimaster, Coil, Register