General Modbus Protocol Considerations for IMx Devices

Modbus is a request/reply protocol that allows communication between multiple industrial devices connected on different types of buses or networks.

The Modbus protocol is currently implemented using serial transmission lines (RTU, ASCII), TCP/IP over Ethernet and Modbus Plus.

Modbus over serial line

This Modbus variant designed for data transmission over a serial line, consists in principle of two wires where every data line has only two possible values, 0 or 1:

- Tx: wire for data transmission (transmitter)
- Rx: wire for data reception (receiver)

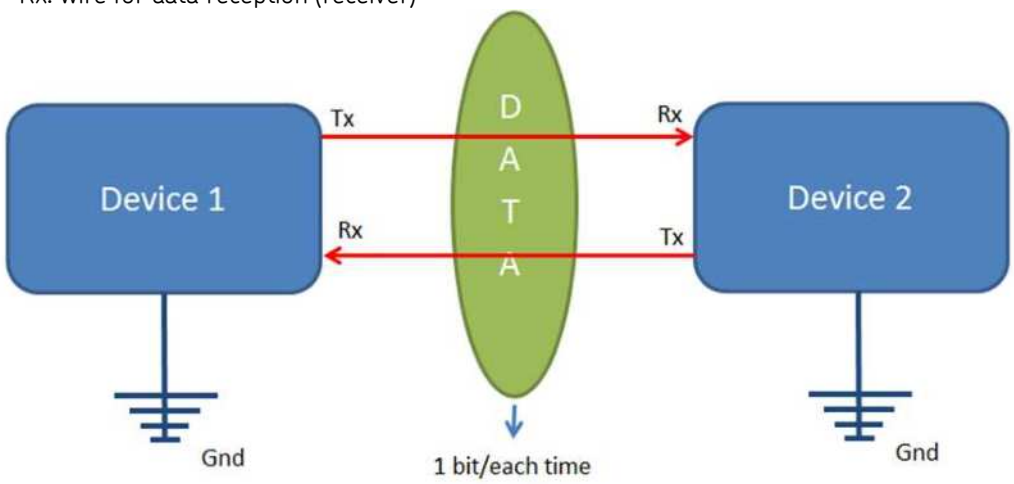

Fig. 1. General principle of the Modbus communication protocol.

To assure interoperability between both devices, each device must have the same transmission mode and serial port parameters.

PUB CM3226 EN February 2017

Notes:

- When sending text, ASCII code converted to binary is used.
- Each byte is always sent from the least significant bit (LSB) to the most significant bit (MSB).
- It is usually denoted from left to right LSB to MSB.
- Data is sent in serial blocks of 8 bits, plus 3 additional bits: 1 start bit, 1 stop bit and 1 bit for parity. The set of 11 bits sent is known as a frame.

Example: Sending the number 65 from one device to another (Number 65 = 1000001).

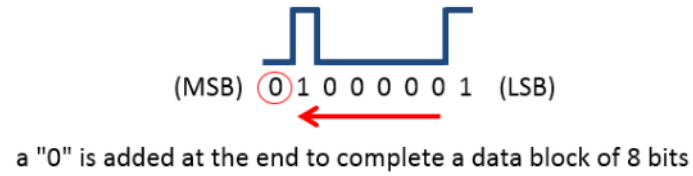

Fig. 2. Example of sending 8 data bits.

Start bit and stop bit

To each data block of 8 bits, it is required to add 2 more bits, one at the beginning and one at the end to indicate the receiver where the data starts (start bit) and ends (stop bit).

Example: Sending the number 65 with start and stop bits.

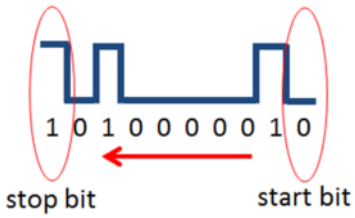

Fig. 3. Example of sending start and stop bits.

Parity bit

Parity bit is an additional bit that is used to detect errors on the sent frames. The available options are:

- None
- Even
- Odd

Example: Sending the number 65 with even parity.

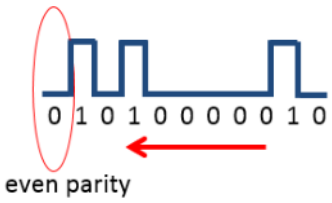

Fig. 4. Example of sending parity bit.

Note: The Modbus Specification V1.02 recommends to use two stop bits if parity bit is None, but most Modbus applications have implemented the 10-bit frame, leaving out the second stop bit, to increase data transfer. This frame is commonly known as 8-N-1, in which there are eight (8) data bits, no (N) parity bit, and one (1) stop bit.

Transmission speed

Transmission speed is the speed at which the bits are sent. The data transmission speed indicates the number of bits transmitted in one second and is measured in bauds. The most common baud rates are 9600 and 19200 bps.

Physical layer

Systems that employ Modbus over serial line may use different physical interfaces like RS485 or RS232. IMx devices have the RS485 interface implemented.

Modbus over RS485

The RS485 standard specifies the electrical characteristics of transmitter and receivers for use in serial communication systems. This standard is widely used to interconnect devices separated by long distances and in electrical noisy environments.

There are two options available in order to create an RS485 network:

- Full-duplex (uses four wires)
- Half-duplex (uses two wires)

RS485 full-duplex

Two devices can have serial communication using two wires, one called receiver (Rx) and the other called transmitter (Tx). Adding a device called transceiver is required for the standard RS485.

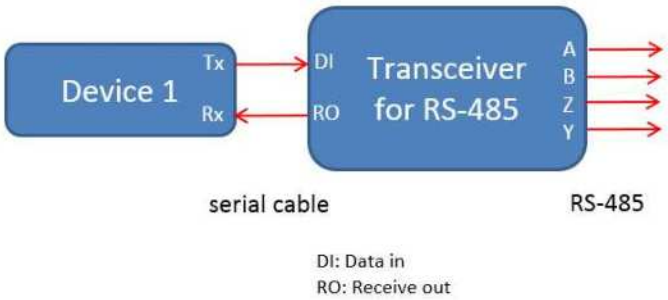

Fig. 5. One transceiver used for full-duplex communication.

At full-duplex communication:

- Transmitter and receiver are independent of each other.
- Both can send and receive data simultaneously (full-duplex).

RS485 half-duplex

Half-duplex communication requires two transceivers, one per device.

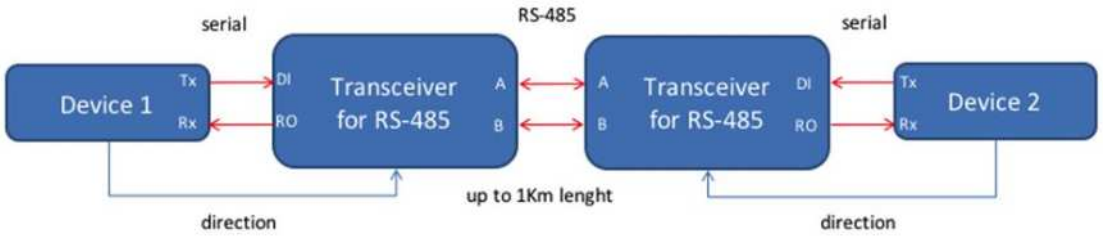

Fig. 6. Two transceivers used for half-duplex communication.

At half-duplex communication:

- The transmitter and receiver cannot work simultaneously in a transmitter with two wires.
- The connection is slower than with four wires.
- Only two wires are required.

One advantage of the RS485 is that it allows connecting additional devices to the same network. When multiple devices are connected to the same network, one of them is called master and the others are called slaves.

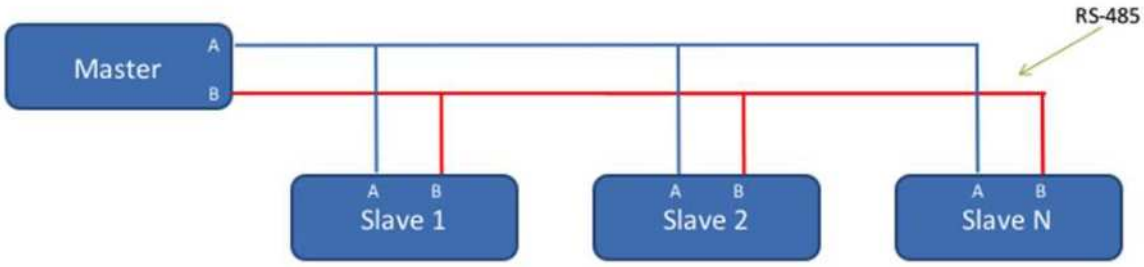

Fig. 7. Master and slave configuration.

Serial transmission modes

The transmission modes define the way the data packages are transferred between the master and the slaves. The two modes are:

- Modbus RTU (remote terminal unit) - communication between devices using binary data
- Modbus ASCII (American standard code for information interchange) - communication between devices using ASCII code

Notes:

– IMx devices only use RTU as serial transmission mode, not ASCII

– IMx serial communication is half-duplex

– Inputs A and B in figure 7 may be labelled in some other manner on equipment that are not IMx devices. If the connection does not work, try reversing the cable.

Master-slave communication rules

- There can be only one master but there can be one or more slaves. The master controls the communication with the different slaves.
- The slaves are limited to return the information requested by the master.
- The master sends the messages and the slave answers them.
- Every slave must have a unique address that goes from 1 to 247.

System requirements for RS485

- A Modbus network may theoretically have a single master and up to 247 slaves, according to the Modbus specification. However, the actual number possible depends on the individual devices on the bus. IMx devices for instance, allow up to 32 devices (1 master + 31 slaves) because the IMx has a >12 kΩ receiver input resistance. This is referred to as 1 unit load in RS485 terminology. Other devices may have 1/2 load, 1/4 load or 1/8 load, for example, which allows more devices on a bus.
- RS485 configuration has one trunk cable, also known as bus, along which devices are connected directly (daisy chain) or by short derivation cable (never more than 20 m).
- RS485 can span distances up to 1,200 m (4,000 ft). However the maximum length of the bus depends on the baud rate, the cable, the number of loads and the serial communication.
- The "Common" circuit must be connected directly to protective ground.
- To minimize the reflections from the end of the RS485 cable, it is required to place a *line termination* at each end of the bus. Most IMx devices control the line termination via DIP switches; consult the *Modbus for IMx User Manual* for more information.

Modbus TCP

Modbus TCP, also known as Modbus TCP/IP, is a Modbus protocol with a TCP wrapper that uses Ethernet to transfer data between master and slave devices.

Notes:

- IMx devices also support data transmission over Modbus TCP.
- Standard TCP port 502 is used for communication.
- For existing Modbus RTU devices a gateway is required to convert from physical layer RS485 (or others) to Ethernet.
- TCP protocol uses big endian format.

Differences between Modbus RTU and Modbus TCP

The main differences between Modbus RTU and Modbus TCP are that the Modbus message TCP frame has an MBAP (Modbus Application) header, the 'Slave address' field is replaced by a single byte 'Unit ID' within the header, and the 'CRC' field is replaced by the error check code implemented on Ethernet.

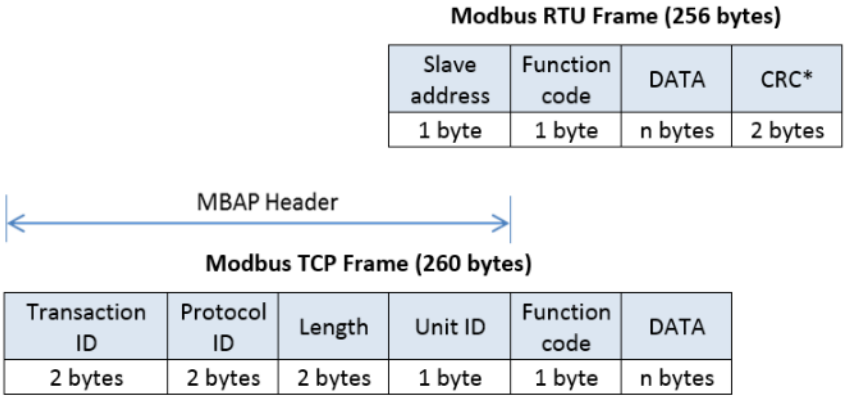

Fig. 8. Differences in the Modbus RTU and TCP message.

*CRC = Cyclic Redundancy Check. It is two bytes added to the end of every Modbus RTU message for error detection. Every byte in the message is used to calculate the CRC. The receiving device also calculates the CRC and compares it to the CRC from the sending device. If even one bit in the message is received incorrectly, the CRCs will be different and an error will result.

- Data field includes the "First register address" and the "Amount of registers" to read.

Data representation in Modbus

The data representation in the Modbus protocol has the following properties and usage:

- Data blocks to store the data from one slave
- Modbus uses four data blocks

Note: Naming conventions for the data blocks can vary from one application to another. For example:

- Holding registers / output registers
- Coils / discrete outputs / digital outputs

Table 1. Data representation in Modbus.

Data blocks	Prefix	Modbus address	Data Type	Master Access	Slave Access
Coils	0	1-9999	Boolean	Read/write	Read/write
Discrete inputs	1	10001-19999	Boolean	Read only	Read/write
Input registers	3	30000-39999	Unsigned word*	Read/write	Read/write
Holding registers	4	40001-49999	Unsigned word*	Read only	Read/write

* 16 bit register with a range from 0 to 65535

Definitions:

- Word: the largest piece of data that can be transferred in a single operation
- Unsigned word: 16-bit characters split into two bytes
- Floating points = Four bytes in two words

Notes:

- The terminology of the Modbus standards has changed over time. For example, “Data blocks” is a newer term, while the term “Prefix” is more common in previous versions. Both are commonly in use.
- Some standards refer to the first register as zero and some as one. Example: Register 1 = address 0000 or register 1 = address 0001
- The Modbus standard specifies only 16-bit integers. Floats and 32-bit integers are vendor extensions that use multiple registers.
- “Coils” and “Discrete inputs” are not used by IMx devices.

Function codes

Function codes are the actions that the Modbus devices can take, depending on the type of data block.

Table 2. Coils function codes.

Code (Hex)	Action
01 (0x01)	reads 1 or multiple coils
05 (0x05)	writes single coil
15 (0x0F)	writes multiple coils

Table 3. Discrete inputs.

Code (Hex)	Action
02 (0x02)	reads inputs only

Table 4. Input registers.

Code (Hex)	Action
04 (0x04)	reads input registers

Table 5. Holding registers.

Code (Hex)	Action
03 (0x03)	reads holding registers
06 (0x06)	writes single register
16 (0x10)	writes multiple registers

Note: IMx firmware supports only function codes 03, 04 and 16.

Endianness

Endianness, in terms of bytes, defines the byte order in which the data bigger than one byte will be stored.

Big endian: the MSB is stored first at the lowest storage address (from right to left).

Little endian: the LSB is stored first (from right to left).

For example, in hexadecimal the value 0x4A3B2C1D:

Big endian = {4A, 3B, 2C, 1D}

Little endian = {1D, 2C, 3B, 4A}

Swap

Swap is the action of interchanging values from two variables. It is used with the variable types bit, byte and word.

Byte swap: interchanges the order of two bytes (one word is two bytes).

Word swap: interchanges the order of two words.

The following table describes the order of bytes and words in the data stream, with example integer.

Table 6. Byte and word order of integers.

Integer 32-bit				**Integer 16-bit**		**Description**
0x0d0c0b0a=218893066				**0x0e0f=3599**		Example data
n+0 (word 1)		n+1 (word 2)		n+2 (word 1)		
d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	Byte stream
0d	0c	0b	0a	0e	0f	Big endian
0b	0a	0d	0c	0e	0f	Big endian word swap*
0a	0b	0c	0d	0f	0e	Little endian
0c	0d	0a	0b	0f	0e	Little endian word swap

* This is what IMx uses when outputting floats.

Notes:

- Big endian with and without word swap is used 99% of the time. SKF recommends using in the order of this table!
- In the table above, a 16-bit register is not affected by word swap because it has only one word.

Floating point numbers (floats) are represented according to IEEE-754. The following table describes the byte and word order in the data stream.

Table 7. Byte and word order of floats.

Value (dec)	Float 32-bit (hex)				Description
	n+0 (word 1)		n+1 (word 2)		
	d[0]	d[1]	d[2]	d[3]	Byte stream (Big endian)
-inf	ff	80	00	00	Negative infinity
-1.0	bf	80	00	00	
0.0	00	00	00	00	
1.0	3f	80	00	00	
+inf	7f	80	00	00	Positive infinity
nan	7f	[c-f]*	*	*	Not a number

Example: A slave device is outputting the decimal value 0,981 in big endian word swap format using Modbus registers 112 and 113.

Register 30112 = 22d1

Register 30113 = 3f7b

The IMx will read 0x3f7b22d1 = 0,981

SKF Condition Monitoring Center – Luleå
Aurorum 30, 977 75 Luleå, Sweden
Telephone: +46 (0) 31 337 10 00, Fax: +46 (0) 920 134 40
Web: www.skf.com

PUB CM3226 EN February 2017