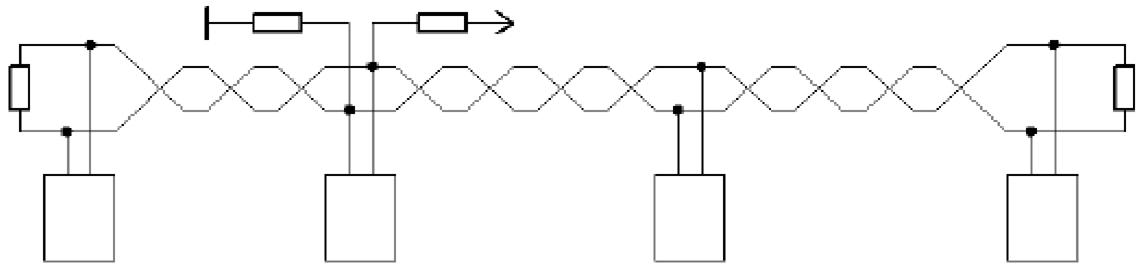
BPC
Boddingtons Pφwer Controls
Unit 1 Zone D Chelmsford Road I.E, DUNMOW CM6 1XG Essex
TEL: 01371 876543., fax: 01371 875460.
sales@boddingtonspowercontrols.co.uk

EMM5 MODBUS COMMUNICATION



EMM-5 MODBUS

CONTENTS

Important Information!

! **If the above sign appears besides a text passage in the manual the reader is strongly advised to read the corresponding information, as it may be very important for usage of the device.**

It can contain safety advice or other information for the correct handling of the device.

If the information is disregarded, the device may be inoperable or even damaged!

Additional documentation for the MODBUS protocol can be found at www.modbus.org.

The MODBUS standards are also available from there.

1 Overview

The MODBUS extension of the EMM5 offers the possibility to read values from the device via a remote connection by a computer system for further investigations and computations.

This document describes the transmission by use of the MODBUS-protocol. This protocol defines methods for data transmission and access control, but doesn't restrict the user to one single physical transmission system. In case of the EMM5, RS485 is used on the physical layer. As this is a bus-capable interface it is possible to connect more than one EMM5 to a single pair of wire and access the units by use of an ID number.

A lot of commercial devices and PLCs are able to use the MODBUS protocol, either as bus master or slave. Various SCADA solutions are also available from different vendors. So, the integration of the EMM5 in an existing bus-system or setting up a new bus system is only a minor issue.

2 MODBUS / RS485

The implementation basically consists of two parts:

- The RS485 transmission is used for serial data transport. It is able to interconnect more than one device in a bus-like configuration. The RS485 protocol offers its “services” to the higher-level MODBUS protocol.
- The MODBUS protocol uses the underlying serial data transport layer (RS485 in this case) to communicate with several bus devices. It defines commands, address structures and data structures to access the slave device.

2.1 The physical layer - RS485 (defined in EIA485/ISO8482)

RS485 offers basic serial data transport to the higher-level MODBUS protocol layers. It is therefore called the “physical layer” of the bus system. Higher layers use the lower physical layer as a basic “service“ for data transport.

RS485 uses two data wires for serial transmission. Each of them is driven to 0V or 5V by the transmitting device. The two data wires always have different voltage levels. One state (one wire 5V, other wire GND) represents the logic “OFF” state. The two wires exchange their voltages for the logic “ON” state. This differential transmission mode makes the RS485 bus very resistant against electro-magnetic distortions and therefore allows long transmission distances of more than 1000 metres.

The data transmission rates of the EMM5 can be selected between 1200, 2400, 9600, 19200 or 38400 baud. The parity can be selected between even, odd and no parity. All bus devices need to use the same settings. Standard settings are: 9600 baud and even parity.

There exist two different types of RS485:

- 2-wire RS485: This type uses only two data wires, which form one data channel. This means, that, after sending a request, the bus master has to deactivate its transmitter to make the data line free for the answering device. (Half-duplex mode)
- 4-wire RS485: this types uses one data line (=two wires) for the master->slave direction and another one (two more wires) for the slave ->master direction. The EMM5 does not support 4-wire RS485.

Both types, 2wire and 4wire, need another line to be connected, although it is not mentioned explicitly: the common ground GND. So, for the 2wire version you need a cable with 3 wires, for the 4wire version one with 5 wires! You should use a shielded cable, but never use the shield for GND connection. It should only be connected to protective ground to reduce electromagnetic influences.

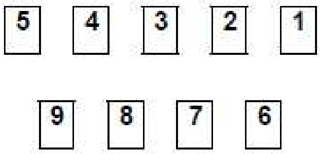
The RS485 bus interconnects more than one device (typically up to 32). To accomplish this, the several data signals have to be interconnected for all bus devices. These are the two data lines and the common ground GND. All devices are connected to the bus in parallel. Avoid using taps, as they tend to be the source of transmission errors if they are too long. You should always prefer direct connection of the device to the main bus wire.

One bus cable with all its devices is called a “bus segment”. Several segments can be interconnected by using “repeaters”.

2.1.1 Connection

The MODBUS interface exists in two variants:

a) connection with 9-pin sub-d



PIN1 +5V (only for data line bias, **do never supply any other external circuits from this voltage output!**)

PIN2 GROUND for biasing and as common ground for all bus participants.

PIN5 D (B) - data signal B

PIN9 D (A) - data signal A

b) connection with 3-pin connector

This connection variant uses a 3-pin connector. The connections can be seen in the picture. To use the MODBUS, one must connect the data lines + and - and the common ground (middle pin).

2.1.2 Line Termination

One very important issue is the termination of the bus line. This is definitely needed for a working bus system to cancel out echoes from the line ends that would distort data signals. To terminate the bus cable, one must add a resistor at each end of the bus cable. The value of the resistor must match the cables impedance. At most times, 120 Ω is a good value to start with. Connect the termination resistor between data wires at each end of the bus segment.

Some devices, especially bus converters have built-in resistors. Please check the manuals for all devices used on the bus. If these internal resistors cannot be disabled, this has a very important influence on your bus: you must place these devices at one of the ends of the bus! As the bus has only two ends, you can use only two devices with fixed resistors per bus segment!

2.1.3 Line biasing

Another important issue is line biasing. If no device is actually transmitting, the data wires are left at floating. Because of the termination resistors, both will have nearly the same voltage. This could result in spurious data signals because of external influences. Line biasing is used to give the data wires a defined "off"-state in this case.

Two resistors of approx. 500-600Ω have to be connected from D(+) line to +5 V and from D(-) to GND. The two bias resistors are needed only once per bus, the position of the bias resistors is not important. They may be placed anywhere on the bus, even in the "middle". Please check in the manuals of all bus devices, if resistors are internally provided!

When the device is equipped with connection variant a) (9-pin sub-d), the voltages 5V and GND are available on the bus connector, so these two resistors can be soldered inside the connector case.

Unfortunately, this is not possible for connection variant b) (3pin). **Attention at different producers the description A = + and B = - is not correct. This has to be checked from case to case.**

2.1.4 Communication Indicator

The yellow LED on the backside of the device indicates an active transmission. It flashes only, if the device is actually communication with the bus master.

The communication indicator LED is available for both connector variants.

To be able to control the transmission, the PDU is extended with additional blocks of data for transmission control purposes. For RS485, this extension results in the "application data unit" (ADU).

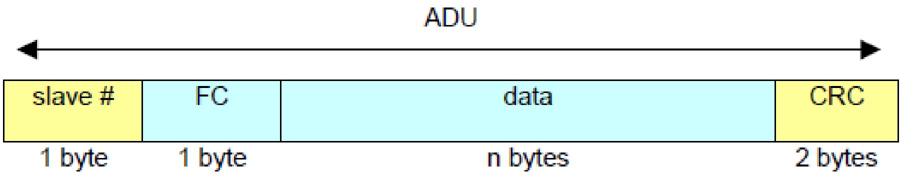


Illustration 2 : "application data unit" - ADU

The application data unit, as it is used with the serial transmission over RS485, contains two additional blocks of data:

- The first field specifies the target for the data block, the "slave number" (=slave address)
- The transmission is additionally secured by a CRC16 error correction code.

2.2.3 Serial transmission modes

The protocol defines two different encodings for the frames' data contents. **The EMM5 always uses the RTU-mode! ASCII mode is not implemented and is mentioned here only for the purpose of completeness.**

"remote terminal unit" (RTU)

In this transmission mode every 8bit data word contains two 4-bit hexadecimal numbers. They are transmitted as one complete byte; a maximum transmission density is reached. With every data word, the following information is transmitted:

- 1 start bit
- 8 data bits, "least significant bit" first
- 1 parity bit (if set)
- 1 stop bit for parity even or odd / 2 if parity is none to compensate missing parity bit

"American standard code for information interchange" (ASCII)

In ASCII-mode, the two 4-byte nibbles of an 8bit data word are transmitted separately in ASCII-code representation. A data byte which contents $5B_{hex}$ will be divided into two parts and transmitted separately as one byte each. The result is that TWO data bytes are transmitted with contents 35_{hex} (=ASCII-Code "5") and 42_{hex} (=ASCII-Code "B"). This data mode is intended for compatibility reasons and makes debugging on the transmission line easier but it also decreases transmission speed significantly.

2.2.4 Function codes

As already mentioned, the data packet contains "function codes" which specify a command from the bus master to the bus slave. The slave executes the command (if possible) and then answers with the same function code in the reply to acknowledge the command. The valid range for function codes is specified from 1 to 255, but only a part of it is actually really used. Please refer to the MODBUS specifications for detailed Information. If it is impossibly for a slave to execute a command, it replies with an exception (=an error code). The function code of an exception packet is the function code of the received command, which caused the error, but it was changed in a certain way: The most significant bit is set by the slave to signal the error condition to the master. The contents of the data block specify the error in more detail.

The EMM5 supports function codes 03_{hex} (read holding register), 04_{hex} (read input register) and 06_{hex} (write single register).

2.2.5 Exception codes

If a slave is not able to execute a command, which was sent by the master, it answers with exception codes. A full list of codes can be found in the MODBUS specification. We do not include this list here, because the master software will be able to handle most exceptions automatically. If one has to program the MODBUS master stack by himself he will need the full specifications, and with that, he gets the full list of ERROR codes.

2.2.6 Master-Slave protocol

For communication, a master-slave protocol is used. Only the bus master is permitted to initiate a transfer. The "master" starts data exchange by sending a command to a slave by transmitting a data frame with the corresponding function code (=command) to the slave, which will then execute it.

- The unicast-mode is normally used to communicate on a Modbus system. One single slave is addressed by the slave number in the master's data packet. The valid address range is between 1 and 247. The slave then executes the command and answers by sending a data packet as acknowledge back to the master.

There are different types of addresses:

- **The MODBUS address always starts with 0 and can go up to 65535. It can be used with any function code.**
- **Certain PLCs lack correct handling of the 0 and therefore add 1 to the address. So their addresses (MODBUS address +1) always start with 1.**
- **Some SCADA tools add an offset to determine the function code, which shall be used to access the device at the given address. They also sometimes add 1 to the MODBUS address. As an example, address 40001 would be "read MODBUS address 0 with function code 03_{hex}", 30012 would be "read MODBUS address 11 with function code 04_{hex}".**

The following tables always give the MODBUS addresses mentioned first in above list.

2.2.9 Measured values

The measured values are available beginning from address 0 in intervals of 6 data words. For each value in this table, the maximum and minimum values are also available. To read the maximum, just add 2 to the address of a value, for the minimum add 4. (Example: to get the minimum voltage L1-N, read from address 00034 = 00030+4). All these values can be accessed with function codes 03_{hex} and 04 $_{hex}$.

Address	Value	Words	Type	Unit
00000	Frequency (taken from voltage L1-N)	2	REAL	Hz / cps
00006	Current I-L1	2	REAL	A
00012	Current I-L2	2	REAL	A
00018	Current I-L3	2	REAL	A
00024	Current I-N	2	REAL	A
00030	Voltage L1-N	2	REAL	V
00036	Voltage L2-N	2	REAL	V
00042	Voltage L3-N	2	REAL	V
00048	Voltage L1-L2	2	REAL	V
00054	Voltage L2-L3	2	REAL	V
00060	Voltage L3-L1	2	REAL	V
00066	Fundamental current If-L1	2	REAL	A
00072	Fundamental current If-L2	2	REAL	A
00078	Fundamental current If-L3	2	REAL	A

00084	Fundamental current If-N	2	REAL	A
00090	Apparent power S-L1	2	REAL	VA
00096	Apparent power S-L2	2	REAL	VA
00102	Apparent power S-L3	2	REAL	VA
00108	Apparent power S-sum	2	REAL	VA
00114	Active power P-L1	2	REAL	W
00120	Active power P-L2	2	REAL	W
00126	Active power P-L3	2	REAL	W
00132	Active power P-sum	2	REAL	W
00138	Reactive power Q-L1	2	REAL	var
00144	Reactive power Q-L2	2	REAL	var
00150	Reactive power Q-L3	2	REAL	var
00156	Reactive power Q-sum	2	REAL	var
00162	Power factor pf-L1	2	REAL	-
00168	Power factor pf-L2	2	REAL	-
00174	Power factor pf-L3	2	REAL	-
00180	Power factor pf-sum	2	REAL	-
00186	Fundamental phase angle phi-L1	2	REAL	degrees
00192	Fundamental phase angle phi-L2	2	REAL	degrees
00198	Fundamental phase angle phi-L3	2	REAL	degrees
00204	Fundamental cos φ cp-L1	2	REAL	-
00210	Fundamental cos φ cp-L2	2	REAL	-
00216	Fundamental cos φ cp-L3	2	REAL	-
00222	Total harmonic distortion THD-I1	2	REAL	%
00228	Total harmonic distortion THD-I2	2	REAL	%
00234	Total harmonic distortion THD-I3	2	REAL	%
00240	Total harmonic distortion THD-I-N	2	REAL	%
00246	Total harmonic distortion THD-U1	2	REAL	%
00252	Total harmonic distortion THD-U2	2	REAL	%
00258	Total harmonic distortion THD-U3	2	REAL	%
00264	Ambient temperature TEMP	2	REAL	°C
00270	Damped current Ith-L1	2	REAL	A
00276	Damped current Ith-L2	2	REAL	A
00282	Damped current Ith-L3	2	REAL	A

00288	Damped current Ith-N	2	REAL	A
00294	Damped active power Pth-L1	2	REAL	W
00300	Damped active power Pth-L2	2	REAL	W
00306	Damped active power Pth-L3	2	REAL	W
00312	Damped active power Pth-sum	2	REAL	W

2.2.10 Harmonics

Harmonics are stored in separate arrays of FLOAT values for each current / voltage. The table below gives the corresponding base addresses. Each data array contains 63 values with 2 words length each. The first table entry holds the fundamental wave. After the fundamental wave (=harmonic of order 1), the other harmonics follow up to the 63^{d} order. Remember that each FLOAT value occupies 2 words, so always increase the address by 2 for the next value.

If the current or voltage is too small to calculate valid harmonics from it, the value at the base address (= the fundamental) reads 0.0%. This indicates, that the higher harmonics for the current or voltage are also invalid!

All these values can be accessed with function codes $03_{hex\ and}$ 04_{hex}.

Address	Value	Words	Type	Unit
768	Base address for harmonics I - L1	63*2	REAL	%
898	Base address for harmonics I - L2	63*2	REAL	%
1028	Base address for harmonics I - L3	63*2	REAL	%
1158	Base address for harmonics I - N	63*2	REAL	%
1288	Base address for harmonics U - L1-N	63*2	REAL	%
1418	Base address for harmonics U - L2-N	63*2	REAL	%
1548	Base address for harmonics U - L3-N	63*2	REAL	%

2.2.11 Work accu

The work counters/accumulators are arranged in a special way. This is necessary to protect them from precision degradation. Each counter consists of two parts:

1. A FLOAT-type base counter which simply accumulates/integrates the power. If this counter reaches 1000000.0, the extended counter is increased by one and 1000000.0 is subtracted from the base counter.
2. A LONG-type extended counter, which is used to count portions of MW / Mvar up to $(2^{32}-1)\cdot 10^6$.

To get the real work value, one has to multiply the extended counter by 1000000 and then add the result to the base counter. This keeps the precision of the FLOAT-type base counter in acceptable range, so no work is lost for big counter values.

All these values can be accessed with function codes $03_{hex\ and}$ 04_{hex}.

Address	Value	Words	Type	Unit
512	WP import L1 - base counter	2	REAL	Wh
514	WP import L1 - extended counter	2	LONG	MWh
516	WP import L2 - base counter	2	REAL	Wh
518	WP import L2 - extended counter	2	LONG	MWh
520	WP import L3 - base counter	2	REAL	Wh
522	WP import L3 - extended counter	2	LONG	MWh
524	WP import sum - base counter	2	REAL	Wh
526	WP import sum - extended counter	2	LONG	MWh
528	WP export L1 - base counter	2	REAL	Wh
530	WP export L1 - extended counter	2	LONG	MWh
532	WP export L2 - base counter	2	REAL	Wh
534	WP export L2 - extended counter	2	LONG	MWh
536	WP export L3 - base counter	2	REAL	Wh
538	WP export L3 - extended counter	2	LONG	MWh
540	WP export sum - base counter	2	REAL	Wh
542	WP export sum - extended counter	2	LONG	MWh
544	WQ inductive L1 - base counter	2	REAL	varh
546	WQ inductive L1 - extended counter	2	LONG	Mvarh
548	WQ inductive L2 - base counter	2	REAL	varh
550	WQ inductive L2 - extended counter	2	LONG	Mvarh

552	WQ inductive L3 - base counter	2	REAL	varh
554	WQ inductive L3 - extended counter	2	LONG	Mvarh
556	WQ inductive sum - base counter	2	REAL	varh
558	WQ inductive sum - extended counter	2	LONG	Mvarh
560	WQ capacitive L1 - base counter	2	REAL	varh
562	WQ capacitive L1 - extended counter	2	LONG	Mvarh
564	WQ capacitive L2 - base counter	2	REAL	varh
566	WQ capacitive L2 - extended counter	2	LONG	Mvarh
568	WQ capacitive L3 - base counter	2	REAL	varh
570	WQ capacitive L3 - extended counter	2	LONG	Mvarh
572	WQ capacitive sum - base counter	2	REAL	varh
574	WQ capacitive sum - extended counter	2	LONG	Mvarh

! **There are other values present in the devices memory than the ones mentioned above. They can contain important setup data for the device. Do never write to any address if you're not sure what data it contains!**