ZIEHL industrie – elektronik GmbH + Co KG
Daimlerstr.13, 74523 Schwäbisch Hall, Germany
+ 49 791 504-0, info@ziehl.de, www.ziehl.de

Operating Manual STWA4MH

updated: 2020-05-06 / sm
from Firmware: 0-00

- RS485 interface with modbus protocol

Table of contents

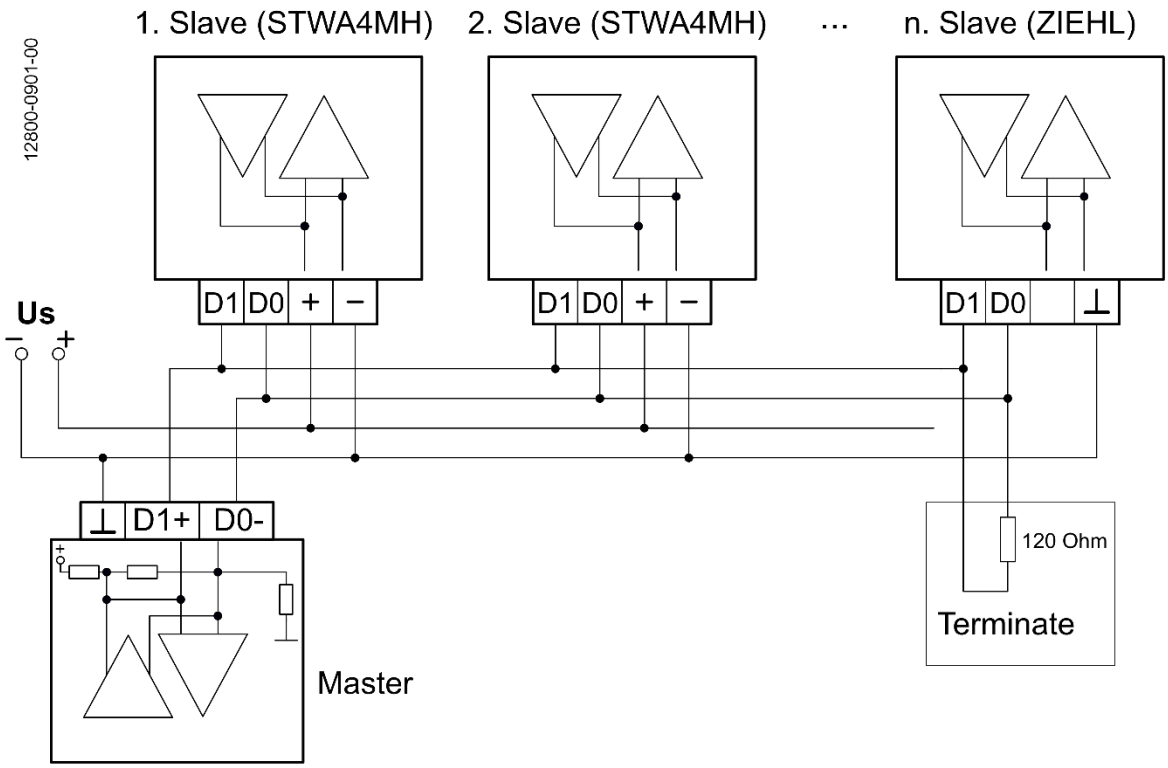
1 Important Information

Please read the general operating manual of the STWA4MH carefully and observe the safety instructions.

LED- function:

- flashes 0,5s: for 60s after switching Us on (additional Modbus address 247 is active), device is ready for operation
- LED on: device is ready for operation
- Pulses briefly: Modbus communication with STWA4MH

2 Connecting diagram for RS485-BUS



3 RS485-BUS interface parameter

BUS- address	baud rate	Data bits	Parity	Stop bit
1 … 246	4800, 9600, 19200, 57600, 115200	8	even, odd, none	1, 2

Factory setting interface parameters:

- address = 248 (invalid Modbus address, must be changed during commissioning)
- 9600 baud, 8 data bits, even parity, 1 stop bit

The STWA4MH can be reached for 60s each time it is switched on (Us on) at address 247 (LED flashes 0,5s).
During this time, the Modbus address can be reprogrammed (see Modbus function 0x10).
ATTENTION: No other device in the RS485 bus may use address 247 at this time!

The STWA4MH RS485 acts as a slave in the bus system.
RTU-Mode is used.

4 Structure of the telegram

After Modbus specification.
For details, please refer to the original Modbus documentation, which can be found at
http://www.modbus.org

5 Supported function codes

Function code	description	use
0x03	Read Holding Registers	Read data from register
0x10	Write Multiple Registers	Write data to register
0x2B	Read Device Identification	Read device information

6 Function code 0x03 – Read data from register

Request from the master				
Byte no.	**definition**		**1. example**	**2. example**
1	slave- address		0x01	0xF7
2	function		0x03	0x03
3	start-address	Hi-Byte	0x00	0x00
4		Lo-Byte	0x11	0x00
5	No. words (Bytes / 2)	Hi-Byte	0x00	0x00
6		Lo-Byte	0x07	0x04
7	checksum CRC-16	Lo-Byte	0x54	0x50
8		Hi-Byte	0x0D	0x9F

Answer from Slave (STWA4MH)				
Byte no.	**definition**		**1. example**	**2. example**
1	slave- address		0x01	0xF7
2	function		0x03	0x03
3	No. Bytes (n) (words x 2)		0x0E	0x08
4	1. word (2 Bytes)	Hi-Byte	0x00	0x00
5		Lo-Byte	0x00	0xF8
6	2. word (2 Bytes)	Hi-Byte	0x00	0x00
7		Lo-Byte	0x00	0x01
8	3. word (2 Bytes)	Hi-Byte	0x00	0x00
9		Lo-Byte	0x00	0x01
10	n- words (2 Bytes)	Hi-Byte	0x00	0x00
11		Lo-Byte	0x00	0x01
: :	: :			
3 + (n + 1)	checksum CRC-16	Lo-Byte	0x2B	0x5C
3 + (n + 2)		Hi-Byte	0x37	0x53

6.1 Function code 0x03 – Register table

Add. hex	**Data typ**	**Register**	**value / value range**
0000	unsigned int 16	Modbus address	1 … 246
0001	unsigned int 16	Baud rate	0 = 4800Bd, 1 = 9600Bd, 2 = 19200Bd, 3 = 57600Bd, 4 = 115200Bd
0002	unsigned int 16	Parity	0 = no Parity, 1 = even Parity, 2 = odd Parity
0003	unsigned int 16	Stop bit	1 = 1 Stop bit, 2 = 2 Stop bit
0004	unsigned int 16	Transducer factor	1 … 10
0005	unsigned int 16	-	
0006	unsigned int 16	-	
0007	unsigned int 16	-	
0008	unsigned int 16	-	
0009	unsigned int 16	-	
000A	unsigned int 16	-	
000B	unsigned int 16	-	

000C	unsigned int 16	-	
000D	unsigned int 16	-	
000E	unsigned int 16	-	
000F	unsigned int 16	-	
0010	unsigned int 16	state	0 = RMS measurement, 1 = average measurement (from approx. < 0,5A), 2 = over range (> 65A)
0011	unsigned int 16	Measurement value I (Measurement value I / Transducer factor)	0 … 65000 [mA]
0012	unsigned int 16	Measurement value I, moving range 200 ms	0 … 65000 [mA]
0013	unsigned int 16	Measurement value I, moving range 1 s	0 … 65000 [mA]
0014	unsigned int 16	-	
0015	unsigned int 16	-	
0016	unsigned int 16	frequency	0 … 9999 [0,01 Hz]
0017	unsigned int 16	Period counter (continuous counter)	0 … 65535
0018	unsigned int 16	Ring buffer buffer overflow between two Modbus reads	0 = no ring buffer overflow 1 = ring buffer overflow
0019	unsigned int 16	Index ring buffer	0 … 49 (last measurement value in ring buffer...)
001A	unsigned int 16	Last read Index ring buffer	0 … 49 (value "last read index ring buffer" from the last Modbus request)
001B	unsigned int 16	ring buffer [0]	0 … 65000 [mA]
001C	unsigned int 16	ring buffer [1]	0 … 65000 [mA]
001D	unsigned int 16	ring buffer [2]	0 … 65000 [mA]
001E	unsigned int 16	ring buffer [3]	0 … 65000 [mA]
001F	unsigned int 16	ring buffer [4]	0 … 65000 [mA]
0020	unsigned int 16	ring buffer [5]	0 … 65000 [mA]
0021	unsigned int 16	ring buffer [6]	0 … 65000 [mA]
0022	unsigned int 16	ring buffer [7]	0 … 65000 [mA]
0023	unsigned int 16	ring buffer [8]	0 … 65000 [mA]
0024	unsigned int 16	ring buffer [9]	0 … 65000 [mA]
0025	unsigned int 16	ring buffer [10]	0 … 65000 [mA]
0026	unsigned int 16	ring buffer [11]	0 … 65000 [mA]
0027	unsigned int 16	ring buffer [12]	0 … 65000 [mA]
0028	unsigned int 16	ring buffer [13]	0 … 65000 [mA]
0029	unsigned int 16	ring buffer [14]	0 … 65000 [mA]
002A	unsigned int 16	ring buffer [15]	0 … 65000 [mA]
002B	unsigned int 16	ring buffer [16]	0 … 65000 [mA]
002C	unsigned int 16	ring buffer [17]	0 … 65000 [mA]
002D	unsigned int 16	ring buffer [18]	0 … 65000 [mA]
002E	unsigned int 16	ring buffer [19]	0 … 65000 [mA]
002F	unsigned int 16	ring buffer [20]	0 … 65000 [mA]
0030	unsigned int 16	ring buffer [21]	0 … 65000 [mA]
0031	unsigned int 16	ring buffer [22]	0 … 65000 [mA]
0032	unsigned int 16	ring buffer [23]	0 … 65000 [mA]
0033	unsigned int 16	ring buffer [24]	0 … 65000 [mA]
0034	unsigned int 16	ring buffer [25]	0 … 65000 [mA]
0035	unsigned int 16	ring buffer [26]	0 … 65000 [mA]
0036	unsigned int 16	ring buffer [27]	0 … 65000 [mA]

0037	unsigned int 16	ring buffer [28]	0 … 65000 [mA]
0038	unsigned int 16	ring buffer [29]	0 … 65000 [mA]
0039	unsigned int 16	ring buffer [30]	0 … 65000 [mA]
003A	unsigned int 16	ring buffer [31]	0 … 65000 [mA]
003B	unsigned int 16	ring buffer [32]	0 … 65000 [mA]
003C	unsigned int 16	ring buffer [33]	0 … 65000 [mA]
003D	unsigned int 16	ring buffer [34]	0 … 65000 [mA]
003E	unsigned int 16	ring buffer [35]	0 … 65000 [mA]
003F	unsigned int 16	ring buffer [36]	0 … 65000 [mA]
0040	unsigned int 16	ring buffer [37]	0 … 65000 [mA]
0041	unsigned int 16	ring buffer [38]	0 … 65000 [mA]
0042	unsigned int 16	ring buffer [39]	0 … 65000 [mA]
0043	unsigned int 16	ring buffer [40]	0 … 65000 [mA]
0044	unsigned int 16	ring buffer [41]	0 … 65000 [mA]
0045	unsigned int 16	ring buffer [42]	0 … 65000 [mA]
0046	unsigned int 16	ring buffer [43]	0 … 65000 [mA]
0047	unsigned int 16	ring buffer [44]	0 … 65000 [mA]
0048	unsigned int 16	ring buffer [45]	0 … 65000 [mA]
0049	unsigned int 16	ring buffer [46]	0 … 65000 [mA]
004A	unsigned int 16	ring buffer [47]	0 … 65000 [mA]
004B	unsigned int 16	ring buffer [48]	0 … 65000 [mA]
004C	unsigned int 16	ring buffer [49]	0 … 65000 [mA]

7 Function code 0x10 – Write data in register

Request from the master				
Byte no.	**definition**		**1. example**	**2. example**
1	slave- address		0x01	0xF7
2	function		0x10	0x10
3	Start-address	Hi-Byte	0x00	0x00
4		Lo-Byte	0x00	0x00
5	No. Bytes (Bytes / 2)	Hi-Byte	0x00	0x00
6		Lo-Byte	0x04	0x04
7	No. Bytes (n)		0x08	0x08
8	1. Register	Hi-Byte	0x00	0x00
9		Lo-Byte	0x01	0x01
10	2. Register	Hi-Byte	0x00	0x00
11		Lo-Byte	0x04	0x01
12	3. Register	Hi-Byte	0x00	0x00
13		Lo-Byte	0x00	0x01
14	4. Register	Hi-Byte	0x00	0x00
15		Lo-Byte	0x02	0x01
: :	: :			
7 + (n + 1)	checksum CRC-16	Lo-Byte	0xD6	0x7D
7 + (n + 2)		Hi-Byte	0xBB	0x3C

Answer from slave (STWA4MH)				
Byte no.	**definition**		**1. example**	**2. example**
1	slave- address		0x01	0xF7
2	function		0x10	0x10
3	Start-address	Hi-Byte	0x00	0x00
4		Lo-Byte	0x00	0x00
5	No. Bytes (n) (Bytes / 2)	Hi-Byte	0x00	0x00
6		Lo-Byte	0x04	0x04
7	checksum CRC-16	Lo-Byte	0xC1	0xD5
8		Hi-Byte	0xCA	0x5C

7.1 Function code 0x10 – Register table

Add. hex	**Data typ**	**Register**	**value / value range**
0000	unsigned int 16	Modbus address	1 … 246
0001	unsigned int 16	baud rate	0 = 4800Bd, 1 = 9600Bd, 2 = 19200Bd, 3 = 57600Bd, 4 = 115200Bd
0002	unsigned int 16	Parity	0 = no Parity, 1 = even Parity, 2 = odd Parity
0003	unsigned int 16	Stop bit	1 = 1 Stop bit, 2 = 2Stop bit
0004	unsigned int 16	Transducer factor	1 … 10

0101	unsigned int 16	device reset	0 = no Reset, 1 = carry out reset

8 Function code 0x2B – Read device information

Request fom Master				
Byte no.	**definition**		**1. example**	**2. example**
1	slave- address		0x01	0x0A
2	function		0x2B	0x2B
3	MEI Type (always 0x0E) *1		0x0E	0x0E
4	Read Device ID code *2		0x01	0x02
5	Object Id		0x00	0x03
6	checksum CRC-16	Lo-Byte	0x70	0x95
7		Hi-Byte	0x77	0x47

*1 MEI = MODBUS Encapsulated Interface (see Modbus documentation, http://www.modbus.org)

*2 0x01: Query of “Basic” device information (stream access)
0x02: Query of “ Regular” device information (stream access)
0x03: Query of “ Extended” device information (stream access)
0x04: Querying individual device information (individual access)

8.1 Function code 0x2B - Objects

Object Id	**Object-name / description**	**content**	**type**	**category**
0x00	Manufacturer name	ZIEHL industrie- elektronik GmbH + Co KG	ASCII String	Basic
0x01	Product- (article-) number	S225570		
0x02	Revision Firmware	00-00		
0x03	Manufacturer URL	www.ziehl.com	ASCII String	Regular
0x04	Product Name	AC Electronic current transformer		
0x05	Product designation	STWA4MH		
0x80	Serial number	xxxxxxxx	ASCII String	Extended
0x81	Revision Hardware	00-00		
0x82	Revision Bootloader	10-00		

9 Error messages

The telegram sent by the master is checked by the slave (STWA4MH). In the event of an error, an error message is generated and sent back to the master. The 7th bit in the function byte is set to “1”.

Error telegram:

Byte no.	definition		1. example	2. example
1	slave- address		0x01	0x0A
2	function		0x81	0x90
3	Error code		0x02	0x03
4	checksum	Lo-Byte	0xC1	0x7D
5	CRC-16	Hi-Byte	0x91	0xC3

Following error codes are possible:

- 1 (01H) invalid function
- 2 (02H) invalid start address
- 3 (03H) invalid data value
- 4 (04H) Slave- device error

Errors that are not recognized by the slave (telegram is rejected):

- Wrong checksum CRC-16
- Unknown clave address

10 Checksum CRC-16

The checksum is appended to each Modbus telegram and is used to identify transmission errors. It is 2 bytes long and is calculated from all bytes of a telegram. The Lo byte is transferred first and then the Hi byte.

For details, please refer to the original Modbus documentation, which can be found at:
http://www.modbus.org