Protronic 100/500/550
Digitric 500

MODBUS
Interface description

Manual

42/62-50040 EN

Rev. 04

ABB

Protronic 100/500/550
Digitric 500

MODBUS interface description

Manual

Document No. 42/62-50040 EN

Edition: 11.01
Revision: 04

Manufacturer:

ABB Automation Products GmbH
Hoeseler Platz 2

D-42579 Heiligenhaus

Phone: +49 (0) 20 56 - 12 - 51 81
Fax: +49 (0) 20 56 - 12 - 50 81

1 Description

Serial communication of controllers Protronic 100/500/550 and Digitric 500 is effected acc. to the Modbus protocol specification.

The Protronic/Digitric controllers are always "slaves" in the communication, i.e. they react only if the higher-level system, the "master", issues a corresponding command.

Protronic/Digitric supports only the RTU process, and from it only the functions of importance to Protronic/Digitric.

Protronic/Digitric supports only some of the Modbus functions.

Please consult the following documents for more information on the Modbus protocol:

GOULD MODICON MODBUS PROTOCOL

Reference Guide
Gould Inc., Programmable Control Division
P.O Box 3083
Andover, Massachussetts, 01810
PI-MBUS -300 Rev A, November 93

2 Interface module

2.1 RS 485

Technical data

– electrically isolated from the controller electronics
– max. 32 subscribers (including PC)
– line structure without deviations, stub lines to individual subscribers < 0,3 m
– Cable length < 1200 m
– At least three-conductor shielded bus cables with a twisted conductor pair for data transmission and an extra insulated conductor for potential equalisation between the terminals “module zero” of all electrically isolated bus subscribers. To operate non-electrically isolated bus subscribers, an additionally isolated conductor with a large cross-section is needed generally parallel to the data cable.
– Connect the shield of the data cable with the controller case using the shield terminal plate supplied. This is necessary for compliance with the radio-interference limit values and for enhancing the interference immunity of the interface..

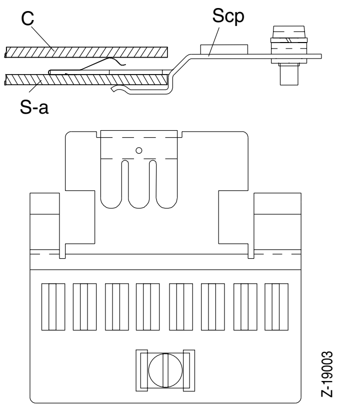

Fig. 2-1 Mounting the shield terminal plate at Protronic 100/500/550
G Housing M Sub-assembly S Shield connection plate

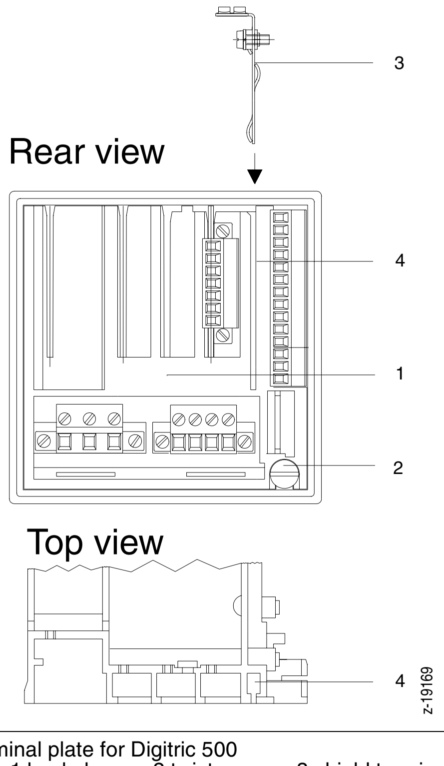

Fig. 2-2 Mounting the shield terminal plate for Digitric 500
↓ Direction of insertion 1 backplane 2 twist screw 3 shield terminal board 4 groove

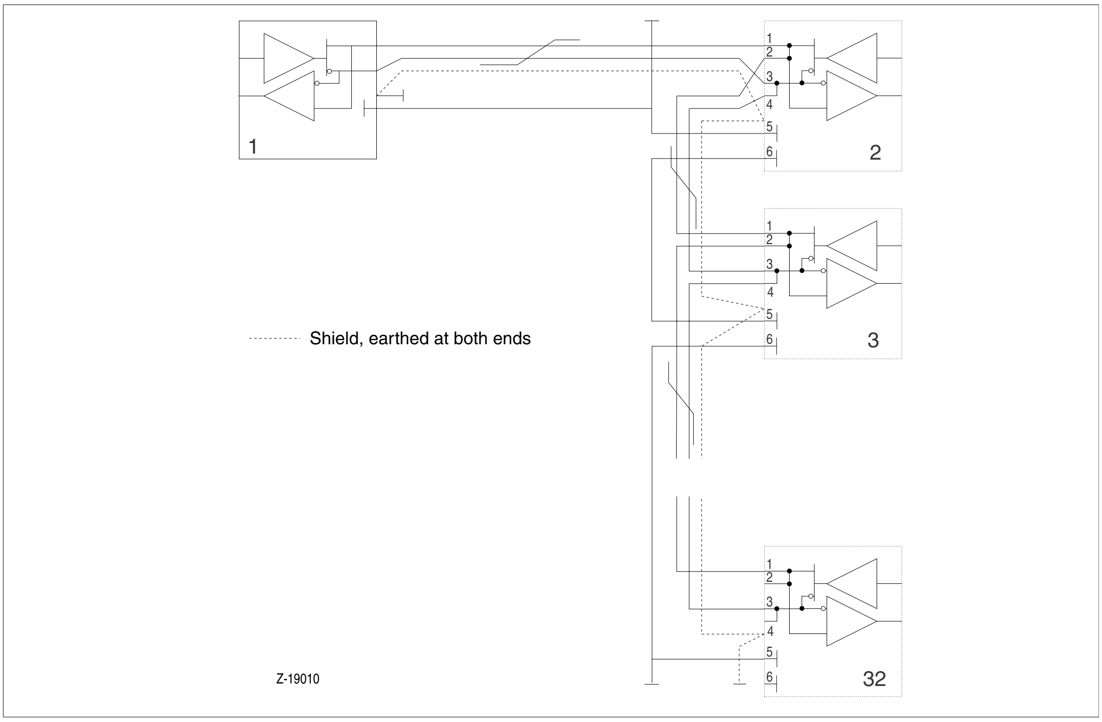

Fig. 2-3 Connection diagram RS 485

2.2 RS 232

Technical data

– Electrically isolated from the controller electronics for direct connection of a configuration PC or modem with 9-pin
– Sub-D connector)
– Max. cable length 10 m

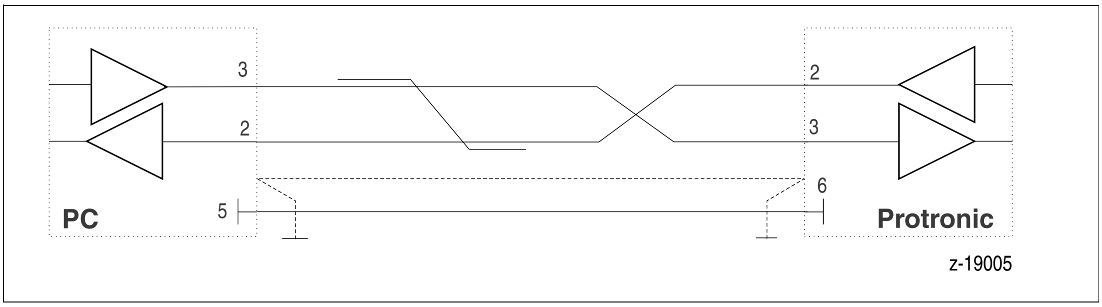

Fig. 2-1 Connection diagram RS 232

3 Date transfer

3.1 General information

Any number of subscribers conforming to the Modbus specification can be operated on one bus. The number of subscribers depends on the transmission technology used.

Via the module fitted to the rear, Protronic features the following interfaces:
– RS 232 for connecting one master (configuration PC or modem)
– RS 485 for connecting max. 32 m subscribers (including master)

A combination of telegram characters are combined to form one or several telegrams for data transmission. These telegrams also take on the “Hand-Shake function“, decreeing that each telegram from the master to the slave must be answered before a new telegram can be sent.

The computer needs appropriate monitoring mechanisms to ensure that subscribers that are not issuing answers are eliminated (Time-Out monitoring).

The timeout time is based on the baud rate in use and on the reaction time of the connected subscribers.

3.2 Telegram characters (frame)

The telegrams comprise a series of 1/0 information items. The values to be transmitted are divided into bytes (= 8 bits). Each of these bits is supplemented by

1 start bit
optionally 1 parity bit (even number of “1“)
1 stop bit

In the following description the expression “byte“ is used, even if 10 or 11 bits are actually transmitted, including the start, stop and parity bits.

3.3 Transmission conventions

The quiescent state of the data line corresponds to logic “1“. Before beginning data transmission, the quiescent state must have prevailed for the duration of at least 3 bytes on the data line.

The distance between the bytes of a telegram must not be greater than 3 bytes, since a distance of more than 3.5 bytes is defined as a separation between two telegrams.

3.4 Telegrams

The Modbus telegrams feature the following structure:

Pause	Address	Function	Data	Checksum	Pause
	1 Byte	1 Byte	n Bytes	2 Bytes	

The figures 1 to 255 are permitted as addresses in the bus subscribers.
Address 0 is the global address (Broadcast address). If this address is used in a telegram, all subscribers accept the telegram, but send no acknowledgement to the master.

3.5 CRC checksum

The checksum is computed via all bytes of a telegram (without start, stop or parity bits).

Exemplary programs are listed in the appendix for determination of the checksum. Please consult the MODBUS original documentation for details.

3.6 Functions

Protronic/Digitric supports the following functions:

Code	Designation	Function
01	READ COIL STATUS	read binary values
02	READ INPUT STATUS	corresponds in Protronic to function 01.
03	READ HOLDING REGISTERS	read REAL-, INT-, DINT- or LONG values
04	READ INPUT REGISTERS	corresponds in Protronic/Digitric to function 03, which is used preferably.
05	FORCE SINGLE COIL	set a single binary value
06	PRESET SINGLE REGISTER	set a single integer, for DINT or REAL two of these telegrams are needed
08	LOOPBACK DIA-GNOSTIC TEST	test telegram for diagnosis of communication capabilities of the slave
15	FORCE MULTIPLE COILS	set several successive binary values
16	PRESET MULTIPLE REGISTERS	set several successive values

3.6.1 Function 01

This function is used for polling several successive binary values from the controllers. The Broadcast address 0 is not permitted.

Example:
This telegram requests the binary status of binary inputs BE01 (151) to BE34 (170) i.e. 20 values.
.

Address	Function	Start adress		Number		CRC checksum	
		HByte	LByte	HByte	LByte	LByte	HByte
11	01	00	151	00	20		

(all specifications are decimal)

The binary information is packed into a few bytes in the answer, the required number is calculated from the requested number divided by 8.

Address	Function	Number of bytes	Status 151 to 158	Status 159 to 160	Status 167 to170	CRC checksum	
						LByte	HByte
11	01	03	8 Bit	8 Bit	8 Bit		

(all specifications are decimal)
The number of bytes indicates the number of data bytes following.

Status 151 to 158: Status of binary inputs BE01 to BE14 (the status can be either „0" or „1").

Bit	7	6	5	4	3	2	1	0
Address	158	157	156	155	154	153	152	151
BE..	14	13	12	11	04	03	02	01
Status								

Status 159 to 166: Status of binary inputs BE15 to BE26 (the status can be either “0" or “1").

Bit	7	6	5	4	3	2	1	0
Address	166	165	164	163	162	161	160	159
BE..	26	25	24	23	22	21	16	15
Status								

Status 167 to 170:
Since in this byte only 4 binary information items are transmitted, bits 7 to 4 are assigned "0". Bits 3 to 0 contain the desired data (the status can be either “0" or “1").

Bit	7	6	5	4	3	2	1	0
Address					170	169	168	167
BE..					34	33	32	31
Status	0	0	0	0				

3.6.2 Function 03

This function is used for polling several successive analog values from the controllers. The Broadcast address 0 is not permitted.

REAL values

Real values are encoded in 32 bits in the controller. Hence twice the number of registers is needed to read these values.

Example:
Read analog input AE01 from registers 0 and 1.

Address	Function	Start address		Number		CRC checksum	
		HByte	LByte	HByte	LByte	LByte	HByte
11	03	00	00	00	02		

(all specifications are decimal)

The answer has the following structure:

Address	Function	Number of bytes	Value of AE01				CRC ckecksum	
			Data [0] Address 0		Data [1] Adress 1			
11	03	04	HByte	LByte	HByte	LByte	LByte	HByte

For concurrent polling of several REAL, DINT or LONG values, the number must be increased by 2 per value. The answer telegram is prolonged by 4 bytes per REAL value.
trollers react to it.

Conversion of the 4 bytes to REAL values is described in the next chapter.

INTEGER values

To read integers, the same telegrams are used.

Example:
Read current segment of time scheduler from register 803 (0323H).

Address	Function	Start address		Number		CRC checksum	
		HByte	LByte	HByte	LByte	LByte	HByte
11	03	03H	23H	00	01		

The answer has the following structure:

Address	Function	Number of bytes	Value of segment		CRC checksum	
			Data [0] Addresse 803			
11	03	02	HByte	LByte	LByte	HByte

For concurrent polling of several INT values, the number must be increased by 1 per value. The answer telegram is prolonged by 2 bytes per INT value.

It is possible in principle to poll REAL and INT values in one telegram. To evaluate the answer, the different number of bytes per value must then be taken into consideration.

3.6.3 Function 05

This function is used to set a single binary value. If the Broadcast address "0" is used, all connected controllers react to it.

Example:
Set binary output BA76 (Address 266 = 10AH)

Address	Function	Addresse		Value		CRC checksum	
		HByte	LByte	HByte	LByte	LByte	HByte
11	05	01H	0AH	FFH	01		

To set a binary value, FFH is always sent to reset 00H in the HByte of the value. The LByte of the value is always 0.
The telegram is sent back completely as an answer once the command has been issued.

3.6.4 Function 06

This function is used for writing a single register. To modify an analog value, two such telegrams must be sent in two successive addresses (registers).
If the Broadcast address "0" is used, all connected controllers react to it.

Example:
Write the computer setpoint for L1 (address 228+229 = F4H + F5H).

Address	Function	Address		Value Data [0]		CRC ckecksum	
11	06	00	228 = F5H	HByte	LByte	LByte	HByte

The complete telegram is returned as an answer, after storage of the first partial value in the register.

The second telegram follows and has the following structure:

Address	Function	Address		Value Data [1]		CRC checksum	
11	06	00	229 = F5H	HByte	LByte	LByte	HByte

This telegram is also returned as confirmation

3.6.5 Function 08

This function is used for communication diagnosis.
Details will be given later

3.6.6 Function 15

This function is used to set several successive binary values in the controllers. If the Broadcast address 0 is used, the values apply for all controllers.

Example:
Set binary inputs BA11 to BA26 (adresses 225 to 236 = E1H to EBH).

Address	Function	Start address		Number		Number of bytes	Data 1	Data 2	CRC checksum	
11	15	00	E1H	00	0BH	2	x	y	Lbyte	HByte

Data 1 sets binary outputs BA11 to BA22 (the status can be either „0" or „1").

Bit	7	6	5	4	3	2	1	0
Address	232	231	230	229	228	227	226	225
BE..	22	21	16	15	14	13	12	11
Status x=								

Data 2 sets the remaining binary outputs. (The status can be „0" or „1",
the unused bits must always be „0".)

Bit	7	6	5	4	3	2	1	0
Address					236	235	234	233
BE..					26	25	24	23
Status	0	0	0	0				

3.6.7 Function 16

This function is used for setting several successive analog values in the controllers. If the Broadcast address 0 is used, the values apply for all controllers.
Up to 60 registers or 30 REAL values can be written with one telegram.

For one value, the telegram has the following structure:

Addr.	Funct.	Start address		Number		Numb.	REAL value[1]				Checksum	
							Data [0]		Data [1]		CRC	
11	16	HByte	LByte	HByte	LByte	Byte	HByte	LByte	HByte	LByte	LByte	HByte

Each REAL value is transmitted with a total of 4 bytes in 2 registers (Data [0]) and Data item [1]). For each additional value, the number must be increased by "2" and the number of bytes by "4".

Integers are transmitted as a 16-bit word, data [0].

Addr.	Funct.	Start address		Number		Number	INTvalue [1]		Checksum	
							Data [0]		CRC	
11	16	HByte	LByte	HByte	LByte	Byte	HByte	LByte	LByte	HByte

It is possible in principle to modify REAL, INT and DINT values in one telegram. The various lengths of the values must then be taken into consideration in the structure of the telegrams.

The telegram is returned without data as an answer.

4 Value ranges

REAL	-1,175.494.35E-38 ... 0 ... 3,402.823.47 E+39 are saved in 2 registers (= 4 Bytes)

INT = INT16-32.768 ... 0 ... 32.767

In Protronic/Digitric and IBIS-R other data types are used:
DINT, LONG and TIME are of type INT32.

DINT	-2.147.483.647 ... 0 ... 2.147.483.647
LONG	0 ... 4.294.967.294 (time in ms)
BOOL	0 and 1

5 Computation of data

5.1 INT, DINT, LONG values

INT and DINT values need no special conversion. INT values are written and read as a single register (=2 bytes), DINT and LONG values are read or written as a sequence of two registers (= 4 Bytes).

5.2 REAL values

The Modbus protocol makes provision for only 16-bit integers with signs as transmission values. The Real numbers of the controller must therefore be processed accordingly.

The IEEE format used in the controllers corresponds to that of PCs.

```
|-----data[1]--------|------data[0]------|
31.30....23.22.....16|15.................0
+----------+----------------------------+
|s  8 bits |msb   23  bit mantissa lsb   |
+----------+----------------------------+
|     |              |
|     |              +-------------Mantissa
|     +----------------------------Exponent (7fh)
+----------------------------------Sign bit
                                   (0=Pos,1=Neg)
```

The real exponent value is the exponent minus 7Fh for the IEEE 4 byte Real format.

Depending on the programming language used in the PC, the individual bits of the REAL values can be accessed directly or indirectly. For computer systems using another numerical format or having no access rights to the various component of the REAL values, exemplary programs in C and Basic are listed in the appendix for conversion of Real numbers into the byte pattern of the IEEE format.

Two data formats, which differ only with respect to precision of the numerical value, are available for data transmission.

The variable addresses must be selected acc. to which of these two methods is used.

5.3 Pair of registers method

In order to also be able to transmit floating point numbers with maximal precision, a new method has been implemented in Protronic/Digitric for transmitting a 32-bit value.

The Pair of Registers method is also supported by the ABB Control System. Here two REAL values (4-byte IEEE format) are transmitted in two successive 16-bit registers, i.e. the 4 successive bytes denoting a Real number are divided into 2 x 2 bytes, and no format conversion is effected. Registers with an even address transmit the lower-value WORD, while registers with uneven address (even +1) transmit the higher-value WORD (16-bit register). To retain consistency of representation, both registers must always be transmitted successively for trans-mission of a 32-bit value:

data[0] low word		data[1] hi word	
HB	LB	HB	LB
even		even+1	

5.3.1 Send a pairs of registers to Protronic/Digitric

Convention governing the division of a (4-byte IEEE) Real value in 2 register value of 16 bits, with the even register values to be written being featured in data [0] and the uneven register value in data item [1]:

The following program directly accesses the *pdata available in the IEEE format in the PC (exemplary value = 133,5).

```
int          data[2];
unsigned long *pdata;
float            value;

wert         = 133.5;
pdata        = (void *)&value;
data[0]      = (unsigned)(*pdata & 0xFFFF);
data[1]      = (unsigned)(*pdata >>16);
```

The values must be sent with two telegrams with the function 05 or with one telegram with the function 15, **with data [0] always being sent before data [1].**

5.3.2 Read a pair of registers from Protronic/Digitric

In the Modbus addresses destined for this purpose, the values are available for reading in the mantissa-exponent representation.

Using one telegram, the values can be read with function 03.

Conventions governing the assembly of 2 16-bit register values in one (4-byte IEEE) Real value, with the even register value read being featured in data [0] and the uneven register value in data item [1]:

```
float         *ptrReal
int           data[2]

ptrReal = (float *)&data[0]
```

5.4 Exponent-Mantissa format

This method is also used by the controllers Contric CM1 and C1 and by the ABB Control System Freelance.
From the value range given for REAL, the following values can be transmitted using this method:

+0,0001 to +3.2767 E38
-0,000 to -3.2768 E38
and value 0.

The value sign is featured in the mantissa.

8000H......FFFFH ...0000H......7FFFH
-32.768 -1 0 32.767

Mantissa and exponent are each read and written as a register (16 bits).

Eaxamples:

Value	Exponent	Mantissa
65432	5	6543
12345	5	12345
-1234	4	-1234
1234	4	1234
123.4	3	1234
12.34	2	1234
1.234	1	1234
0.1234	0	1234
0.01234	0	123
0.001234	0	12
0.0001234	0	1

5.4.1 Send mantissa and exponent to Protronic/Digitric

Conventions governing the division of a (4-byte IEEE) Real value into 2 register values of 16 bits (Mantissa and exponent):

```
     exponent = 0
     while (abs(real value>= 1.0 ) {
         realvalue = real value/ 10
         exponent = exponent + 1
     }
     real value = real value * 10000.0
     value = (int)real value
// Observe rounding-off error
     if (value > 0)
         value = value + 0.5;
     else
         value = value - 0.5;
     mantissa = (int)value;

data[0] = mantissa
data[1] = exponent
```

These values must be sent with two telegrams with the function 05 or one telegram with the function 15, with the mantissa always being sent before the exponent.

5.4.2 Read mantissa and exponent from Protronic/Digitric

In the Modbus addresses destined for this purpose, the values are available for reading in the mantissa-exponent representation.

Using one telegram, the values can be read with function 03.

Conventions governing the assembly of 2 16-bit register values (mantissa and exponent) in one (4-byte IEEE) Real value

```
     real value = mantissa
     real value = real value/ 10000.0
     for(i=0;i < exponent; i=i+1)
         real value = real value * 10.0
```

6 Assignment of Protronic/Digitric variables to the MODBUS registers

The lists featured in the appendix show the assignment of the various parameters and dynamic variables to the registers. All register numbers are given in decimal notation.
The different registers can be subdivided into 3 sections.

6.1 Global variables

Registers 0-2000 have been reserved for reading and writing global variables. The pair of registers method is used for reading and writing 32-bit values. A table describing the various variables is featured in the appendix.

If the values are to read with the exponent/mantissa method (from FW 1.149), an offset of 2000 must be added to the respective register (registers 2000-4000).

6.2 Online parameters (from FW 1.149)

Online parameters should only be read and written.

Registers 10000-20000 have been reserved for reading or writing local online parameters with the pair of registers method. If the values are to be read with the exponent/mantissa method, an offset of 10000 must be added to the respective register (registers 20000-30000).

Here the parameters are interpreted as follows acc. to the device manual (42/62-50012):

Device parameter table values: 10000 - 10999

(Convention: 10000 + 2 × Parameter number)

Online parameter Loop 1: 11000 - 11999
Online parameter Loop 2: 12000 - 12999
Online parameter Loop 3: 13000 - 13999
Online parameter Loop 4: 14000 - 14999

(Convention: 10000 + Loop × 1000 + 2 × Parameter number)

L1_Kp has No. 1
Address = 10000 + 1 × 1000 + 2 × 1 = 11002

Programs 1..10
P1 : 15000 - 15199
P2 : 15200 - 15399
....
P10 : 16800 - 16999

(Convention: 15000 + (Prg - 1) × 200 + 2 × Parameter number)

6.3 Special registers

Keyboard remote operation: Register 900
The following hex values simulate a key press; in each case only one code can be transmitted to register 900.

KEY ENTER	0x0200
KEY MENU	0x0100
KEY IND	0x0080
KEY LOOP	0x0040
KEY SPW	0x0020
KEY MAC	0x0010
KEY LEFT	0x0008
KEY RIGHT	0x0004
KEY DOWN	0x0002
KEY UP	0x0001

7 Appendix 1

7.1 MODBUS register table of global variables REAL, INT, DINT, LONG

Register	Short designation	Data type	Description
0	.AE01	REAL	Basic device analog input 1
2	.AE02	REAL	Basic device analog input 2
4	.AE11	REAL	Slot 1 Analog input 1
6	.AE12	REAL	Slot 1 Analog input 2
8	.AE13	REAL	Slot 1 Analog input 3
10	.AE14	REAL	Slot 1 Analog input 4
12	.AE21	REAL	Slot 2 Analog input 1
14	.AE22	REAL	Slot 2 Analog input 2
16	.AE23	REAL	Slot 2 Analog input 3
18	.AE24	REAL	Slot 2 Analog input 4
20	.AE31	REAL	Slot 3 Analog input 1
22	.AE32	REAL	Slot 3 Analog input 2
24	.AE33	REAL	Slot 3 Analog input 3
26	.AE34	REAL	Slot 3 Analog input 4
28	.AE41	REAL	Slot 4 Analog input 1
30	.AE42	REAL	Slot 4 Analog input 2
32	.AE43	REAL	Slot 4 Analog input 3
34	.AE44	REAL	Slot 4 Analog input 4
36	.AE51	REAL	Slot 5 Analog input 1
38	.AE52	REAL	Slot 5 Analog input 2
40	.AE53	REAL	Slot 5 Analog input 3
42	.AE54	REAL	Slot 5 Analog input 4
44	.AE61	REAL	Slot 6 Analog input 1
46	.AE62	REAL	Slot 6 Analog input 2
48	.AE63	REAL	Slot 6 Analog input 3
50	.AE64	REAL	Slot 6 Analog input 4
52	.AE71	REAL	Slot 7 Analog input 1
54	.AE72	REAL	Slot 7 Analog input 2
56	.AE73	REAL	Slot 7 Analog input 3
58	.AE74	REAL	Slot 7 Analog input 4
60 - 69	unassigned		
70	.AA01	REAL	Basic device analog output 1
72	.AA11	REAL	Slot 1 Analog output 1
74	.AA12	REAL	Slot 1 Analog output 2
76	.AA13	REAL	Slot 1 Analog output 3
78 - 79	unassigned		
80	.AA21	REAL	Slot 2 Analog output 1
82	.AA22	REAL	Slot 2 Analog output 2
84	.AA23	REAL	Slot 2 Analog output 3
86 - 87	unassigned		
88	.AA31	REAL	Slot 3 Analog output1
90	.AA32	REAL	Slot 3 Analog output 2
92	.AA33	REAL	Slot 3 Analog output 3
94 - 95	unassigned		
96	.AA41	REAL	Slot 4 Analog output 1
98	.AA42	REAL	Slot 4 Analog output 2
100	.AA43	REAL	Slot 4 Analog output 3
102 - 103	unassigned		
104	.AA51	REAL	Slot 5 Analog output 1
106	.AA52	REAL	Slot 4 Analog output 2
108	.AA53	REAL	Slot 4 Analog output 3
110 - 111	unassigned		

Register	Short designation	Data type	Description
112	.AA61	REAL	Slot 6 Analog output 1
114	.AA62	REAL	Slot 6 Analog output 2
116	.AA63	REAL	Slot 6 Analog output 3
118 - 119	unassigned		
120	.AA71	REAL	Slot 7 Analog output 1
122	.AA72	REAL	Slot 7 Analog output 2
124	.AA73	REAL	Slot 7 Analog output 3
126 - 149	unassigned		
150	.L1_ES1	REAL	General input
152	.L1_ES2	REAL	General input
154	.L1_ES3	REAL	General input
156	.L1_ES4	REAL	General input
158	.L1_ES5	REAL	General input
160	.L1_WAKT	REAL	Current setpoint
162	.L1_YTRACK	REAL	Analog input for tracking
164	.L1_XDIGI	REAL	Digital display PV
166	.L1_XANA	REAL	Analog display PV
168	.L1_D	REAL	Value for D-action
170	.L1_XW	REAL	Control deviation Err
172	.L1_WANA	REAL	Analog display SP
174	.L1_WDIGI	REAL	Digital display Sp
176	.L1_K1	REAL	Constant K1
178	.L1_K2	REAL	Constant K2
180	.L1_K3	REAL	Constant K3
182	.L1_K4	REAL	Constant K4
184	.L1_PID_Y_OUT	REAL	Output of PID controller
186	.L1_XW_EU	REAL	Control deviation in EU
188	.L1_XW_PRZ	REAL	Control deviation in %
190	.L1_YMAX	REAL	Output limit max.
192	.L1_YMIN	REAL	Output limit min.
194	.L1_TIME_DPS_MAN	DINT	Value for time to switch on output in MAN mode
196	.L1_YHAND	REAL	Correction value manual
198	.L1_KP_STEUER	REAL	Parameter control G
200	.L1_KS_STEUER	REAL	Parameter control Gs
202	.L1_TN_STEUER	REAL	Parameter control Tr
204	.L1_TV_STEUER	REAL	Parameter control Td
206	.L1_Y0_STEUER	REAL	Parameter control MR
208	.L1_TT_STEUER	REAL	Parameter control T_t
210	.L1_T1_STEUER	REAL	Parameter control T_1
212	.L1_PID_I_OUT	REAL	Integrator of control module
214	.L1_PID_D_OUT	REAL	D-output of control module
216	.L1_YIN	REAL	Analog input for OUT-external
218 - 223	unassigned		
224	.L1_BA_YOUT	REAL	Duty cycle of on/off controller as 0...100 %
226	.INDS_LOOP1	INT	Display loop position
227	unassigned		
228	.L1_WCOMPUTER	REAL	Computer setpoint
230	.L1_WSOLL0	REAL	Target setpoint 1
232	.L1_WSOLL1	REAL	Target setpoint 2
234	.L1_WSOLL2	REAL	Target setpoint 3
236	.L1_WSOLL3	REAL	Target setpoint 4
238	.L1_WW	REAL	Active setpoint
240	.L1_V	REAL	Ratio setpoint
242	.L1_VISTDIGI	REAL	Display value ratio actual value
244 - 247	unassigned		

Register	Short designation	Data type	Description
248	.L1_LAMBDA	REAL	No function
250	.L1_XANA_SKAL	REAL	Scaled value for PV-display
252	.L1_WANA_SKAL	REAL	Scaled value for SP-display
254	.L1_YCOMPUTER	REAL	Output variable for DDC
256	.L1_W_FOLGE	REAL	Setpoint for slave controller for cascade
258	.L1_YMIN_BR	REAL	OUT-Min-selection override controller Override
260	.L1_YMAX_BR	REAL	OUT-Max-selection override controller Override
262	.L1_YMIN_HR	REAL	OUT-Min selection master controller Override
264	.L1 YMAX_HR	REAL	OUT-Max selection master controller Override
266	.L1_WEXT	REAL	External setpoint
268	.L1-SKAL	REAL	Scaling factor for load/air regulation
270	.L1_R1	REAL	Free REAL variable
272	.L1_R2	REAL	Free REAL variable
274	.L1_R3	REAL	Free REAL variable
276	.L1_R4	REAL	Free REAL variable
278	.L1_R5	REAL	Free REAL variable
280	.L1_R6	REAL	Free REAL variable
282	.L1_R7	REAL	Free REAL variable
284	.L1_R8	REAL	Free REAL variable
286	.L1_T1	LONG	Free LONG (Time) variable
288	.L1_T2	LONG	Free LONG (Time) variable
290	.L1_D1	LONG	Free LONG (DINT) variable
292	.L1_D2	LONG	Free LONG (DINT) variable
294	.L1_D3	LONG	Free LONG (DINT) variable
296	.L1_D4	LONG	Free LONG (DINT) variable
297 - 299	unassigned		
300	.L2_ES1	REAL	General input
302	.L2_ES2	REAL	General input
304	.L2_ES3	REAL	General input
306	.L2_ES4	REAL	General input
308	.L2_ES5	REAL	General input
310	.L2_WAKT	REAL	Current setpoint
312	.L2_YTRACK	REAL	Analog input for tracking
314	.L2_XDIGI	REAL	Digital display PV
316	.L2_XANA	REAL	Analog display PV
318	.L2_D	REAL	Value for D-action
320	.L2_XW	REAL	Control deviation Err
322	.L2_WANA	REAL	Analog display SP
324	.L2_WDIGI	REAL	Digital display SP
326	.L2_K1	REAL	Constant K1
328	.L2_K2	REAL	Constant K2
330	.L2_K3	REAL	Constant K3
332	.L2_K4	REAL	Constant K4
334	.L2_PID_Y_OUT	REAL	Output of PID controller
336	.L2_XW_EU	REAL	Control deviation in EU
338	.L2_XW_PRZ	REAL	Control deviation in %
340	.L2_YMAX	REAL	Output limit max.
342	.L2_YMIN	REAL	Output limit min.
344	.L2_TIME_DPS_MAN	DINT	Value for time to switch on output in MAN mode
346	.L2_YHAND	REAL	Correction value manual
348	.L2_KP_STEUER	REAL	Parameter control G
350	.L2_KS_STEUER	REAL	Parameter control Gs
352	.L2_TN_STEUER	REAL	Parameter control Tr
354	.L2_TV_STEUER	REAL	Parameter control Td
356	.L2_Y0_STEUER	REAL	Parameter control MR
358	.L2_TT_STEUER	REAL	Parameter control T_t
360	.L2_T1_STEUER	REAL	Parameter control T_1
362	.L2_PID_I_OUT	REAL	Integrator of control module
364	.L2_PID_D_OUT	REAL	D-output of control module
366	.L2_YIN	REAL	Analog input for OUT-external
368 - 373	unassigned		
374	.L2_BA_YOUT	REAL	Duty cycle of on/off controller as 0... 100 %

Register	Short designation	Data type	Description
376	.INDS_LOOP2	INT	Display loop position
377	unassigned		
378	.L2_WCOMPUTER	REAL	Computer setpoint
380	.L2_WSOLL0	REAL	Target setpoint 1
382	.L2_WSOLL1	REAL	Target setpoint 2
384	.L2_WSOLL2	REAL	Target setpoint 3
386	.L2_WSOLL3	REAL	Target setpoint 4
388	.L2_WW	REAL	Active setpoint
390	.L2_V	REAL	Ratio setpoint
392	.L2_VISTDIGI	REAL	Ratio actual value
394 - 397	unassigned		
398	.L2_LAMBDA	REAL	No function
400	.L2_XANA_SKAL	REAL	Scaled value for PV-display
402	.L2_WANA_SKAL	REAL	Scaled value for SP-display
404	.L2_YCOMPUTER	REAL	Output variable for DDC
406	.L2_W_FOLGE	REAL	Setpoint for slave controller for cascade.
408	.L2_YMIN_BR	REAL	OUT-Min-selection override controller Override
410	.L2_YMAX_BR	REAL	OUT-Max-selection override controller Override
377	unassigned		
416	.L4_WEXT	REAL	External setpoint
377	unassigned		
420	.L2_R1	REAL	Free REAL variable
422	.L2_R2	REAL	Free REAL variable
424	.L2_R3	REAL	Free REAL variable
426	.L2_R4	REAL	Free REAL variable
428	.L2_R5	REAL	Free REAL variable
430	.L2_R6	REAL	Free REAL variable
432	.L2_R7	REAL	Free REAL variable
434	.L2_R8	REAL	Free REAL variable
436	.L2_T1	LONG	Free LONG (Time) variable
438	.L2_T2	LONG	Free LONG (Time) variable
440	.L2_D1	LONG	Free LONG (DINT) variable
442	.L2_D2	LONG	Free LONG (DINT) variable
444	.L2_D3	LONG	Free LONG (DINT) variable
446	.L2_D4	LONG	Free LONG (DINT) variable
448 - 449	unassigned		
450	.L3_ES1	REAL	General input
452	.L3_ES2	REAL	General input
454	.L3_ES3	REAL	General input
456	.L3_ES4	REAL	General input
458	.L3_ES5	REAL	General input
460	.L3_WAKT	REAL	Current setpoint
462	.L3_YTRACK	REAL	Analog input for tracking
464	.L3_XDIGI	REAL	Digital display PV
466	.L3_XANA	REAL	Analog display PV
468	.L3_D	REAL	Value for D-action
470	.L3_XW	REAL	Control deviation Err
472	.L3_WANA	REAL	Analog display SP
474	.L3_WDIGI	REAL	Digital display SP
476	.L3_K1	REAL	Constant K1
478	.L3_K2	REAL	Constant K2
480	.L3_K3	REAL	Constant K3
482	.L3_K4	REAL	Constant K4
484	.L3_PID_Y_OUT	REAL	Output of PID controller
486	.L3_XW_EU	REAL	Control deviation in EU
488	.L3_XW_PRZ	REAL	Control deviation %
490	.L3_YMAX	REAL	Output limit Max.
492	.L3_YMIN	REAL	Output limit Min.
494	.L3_TIME_DPS_MAN	DINT	Value for time to switch on output in MAN mode
496	.L3 YHAND	REAL	Setpoint manual
498	.L3 KP_STEUER	REAL	Parameter control G

Register	Short designation	Data type	Description
500	.L3 KS_STEUER	REAL	Parameter control Gs
502	.L3_TN_STEUER	REAL	Parameter control Tr
504	.L3_TV_STEUER	REAL	Parameter control Td
506	.L3_Y0_STEUER	REAL	Parameter control MR
508	.L3_TT_STEUER	REAL	Parameter control T_t
510	.L3_T1_STEUER	REAL	Parameter control T_1
512	.L3_PID_I_OUT	REAL	Integrator of control module
514	.L3_PID_D_OUT	REAL	D-output of control module
516	.L3_YIN	REAL	Analog input for OUT-external
518 - 523	unassigned		
524	.L3_BA_YOUT	REAL	Duty cycle of on/off controller as 0...100 %
526	.INDS_LOOP3	INT	Display loop position
527	unassigned		
528	.L3_WCOMPUTER	REAL	Computer setpoint
530	.L3_WSOLL0	REAL	Target setpoint 1
532	.L3_WSOLL1	REAL	Target setpoint 2
534	.L3_WSOLL2	REAL	Target setpoint 3
536	.L3_WSOLL3	REAL	Target setpoint 4
538	.L3_WW	REAL	Active setpoint
540	.L_3V	REAL	Ratio setpoint
542	.L3_VISTDIGI	REAL	Ratio actual value
544 - 547	unassigned		
548	.L3_LAMBDA	REAL	No function
550	.L3_XANA_SKAL	REAL	Scaled value for PV-display
552	.L3_WANA_SKAL	REAL	Scaled value for SP-display
554	.L3_YCOMPUTER	REAL	Output variable for DDC
556	.L3_W_FOLGE	REAL	Setpoint for slave controller for cascade
558	.L3_YMIN_BR	REAL	OUT-Min-selection override controller Override
560	.L3_YMAX_BR	REAL	OUT-Max-selection override controller Override
562- 565	unassigned		
566	.L3_WEXT	REAL	External setpoint
568 - 569	unassigned		
570	.L3_R1	REAL	Free REAL variable
572	.L3_R2	REAL	Free REAL variable
574	.L3_R3	REAL	Free REAL variable
576	.L3_R4	REAL	Free REAL variable
578	.L3_R5	REAL	Free REAL variable
580	.L3_R6	REAL	Free REAL variable
582	.L3_R7	REAL	Free REAL variable
584	.L3_R8	REAL	Free REAL variable
586	.L3_T1	LONG	Free LONG (Time) variable
588	.L3_T2	LONG	Free LONG (Time) variable
590	.L3_D1	LONG	Free LONG (DINT) variable
592	.L3_D2	LONG	Free LONG (DINT) variable
594	.L3_D3	LONG	Free LONG (DINT) variable
596	.L3_D4	LONG	Free LONG (DINT) variable
598 - 599	unassigned		

Register	Short designation	Data type	Description
600	.L4_ES1	REAL	General input
602	.L4_ES2	REAL	General input
604	.L4_ES3	REAL	General input
606	.L4_ES4	REAL	General input
608	.L4_ES5	REAL	General input
610	.L4_WAKT	REAL	Current setpoint
612	.L4_YTRACK	REAL	Analog input for tracking
614	.L4_XDIGI	REAL	Digital display PV
616	.L4_XANA	REAL	Analog display PV
618	.L4_D	REAL	Value for D-action
620	.L4_XW	REAL	Control deviation Err
622	.L4_WANA	REAL	Analog display SP
624	.L4_WDIGI	REAL	Digital display SP
626	.L4_K1	REAL	Constant K1
628	.L4_K2	REAL	Constant K2
630	.L4_K3	REAL	Constant K3
632	.L4_K4	REAL	Constant K4
634	.L4_PID_Y_OUT	REAL	Output of PID controller
636	.L4_XW_EU	REAL	Control deviation in EU
638	.L4_XW_PRZ	REAL	Control deviation in %
640	.L4_YMAX	REAL	Output limit max.
642	.L4_YMIN	REAL	Output limit min.
644	.L4_TIME_DPS_MAN	DINT	Value for time to switch on output in MAN mode
646	.L4_YHAND	REAL	Manual correction value
648	.L4_KP_STEUER	REAL	Parameter control G
650	.L4_KS_STEUER	REAL	Parameter control Gs
652	.L4_TN_STEUER	REAL	Parameter control Tr
654	.L4_TV_STEUER	REAL	Parameter control Td
656	.L4_Y0_STEUER	REAL	Parameter control MR
658	.L4_TT_STEUER	REAL	Parameter control T_t
660	.L4_T1_STEUER	REAL	Parameter control T_1
662	.L4_PID_I_OUT	REAL	Integrator of control module
664	.L4_PID_D_OUT	REAL	D-output of control module
666	.L4_YIN	REAL	Analog input for OUT-external
668 - 673	unassigned		
674	.L4_BA_YOUT	REAL	Duty cycle of on/off controller as 0 ... 100 %
676	.INDS_LOOP4	INT	Display loop position
677	unassigned		
678	.L4_WCOMPUTER	REAL	Computer setpoint
680	.L4_WSOLL0	REAL	Target setpoint 1
682	.L4_WSOLL1	REAL	Target setpoint 2
684	.L4_WSOLL2	REAL	Target setpoint 3
686	.L4_WSOLL3	REAL	Target setpoint 4
688	.L4_WW	REAL	Active setpoint
690	.L4_V	REAL	Ratio setpoint
692	.L4_VISTDIGI	REAL	Ratio actual value
694 - 697	unassigned		
698	.L4_LAMBDA	REAL	No function
700	.L4_XANA_SKAL	REAL	Scaled value for PV-display
702	.L4_WANA_SKAL	REAL	Scaled value for SP-display
704	.L4_YCOMPUTER	REAL	Output variable for DDC
706	.L4_W_FOLGE	REAL	Setpoint for slave controller for cascade
708	.L4_YMIN_BR	REAL	OUT-Min-selection override controller Override
710	.L4_YMAX_BR	REAL	OUT-Max-selection override controller Override
712 - 715	unassigned		
716	.L4_WEXT	REAL	External setpoint
718 - 719	unassigned		

Register	Short designation	Data type	Description
720	.L4_R1	REAL	Free REAL variable
722	.L4_R2	REAL	Free REAL variable
724	.L4_R3	REAL	Free REAL variable
726	.L4_R4	REAL	Free REAL variable
728	.L4_R5	REAL	Free REAL variable
730	.L4_R6	REAL	Free REAL variable
732	.L4_R7	REAL	Free REAL variable
734	.L4_R8	REAL	Free REAL variable
736	.L4_T1	LONG	Free LONG (Time) variable
738	.L4_T2	LONG	Free LONG (Time) variable
740	.L4_D1	LONG	Free LONG (DINT) variable
742	.L4_D2	LONG	Free LONG (DINT) variable
744	.L4_D3	LONG	Free LONG (DINT) variable
746	.L4_D4	LONG	Free LONG (DINT) variable
748 - 749	unassigned		
750	.TAB01	REAL	Output of Table 1
752	.TAB02	REAL	Output of Table 2
754	.TAB03	REAL	Output of Table 3
756	.TAB04	REAL	Output of Table 4
758	.TAB4AE	REAL	Input Table 4 for ESx
760 - 769	unassigned		
770	.ZK01	REAL	Output of state correction 1
772	.ZK02	REAL	Output of state correction 2
774 - 794	unassigned		
795	.WW_LOOP1	INT	Index of selected setpoint
796	.WW_LOOP2	INT	Index of selected setpoint
797	.WW_LOOP3	INT	Index of selected setpoint
798	.WW_LOOP4	INT	Index of selected setpoint
799	.A_LOOP	INT	Loop in the display (0=Loop 1,...)
800	.PG_NR_AKT	INT	Number of active program
801	.PG_SCHNELL	INT	Velocity of time scheduler
802	.PG_NR_SEL	INT	Number of actual program 0...9
803	.PG_SEG	INT	Current segment of time scheduler
804	.PG_LAUF	LONG	Run time of time scheduler since start
806	.W_P	REAL	Time scheduler setpoint
808 - 809	unassigned		
810	.LATERALNR	INT	Address lateral communication
811	.LATERAL1	INT	Status laterale communication no. 1
812	.LATERAL2	INT	Status laterale communication no. 2
813	.LATERAL3	INT	Status laterale communication no. 3
814	.LATERAL4	INT	Status laterale communication no. 4
815	.LATERAL5	INT	Status laterale communication no. 5
816	.LATERAL6	INT	Status laterale communication no. 6

Register	Short designation	Data type	Description
902	.INT_01	INT	free INT variable for communication
903	.INT_02	INT	free INT variable for communication
904	.INT_03	INT	free INT variable for communication
905	.INT_04	INT	free INT variable for communication
906	.INT_05	INT	free INT variable for communication
907	.INT_06	INT	free INT variable for communication
908	.INT_07	INT	free INT variable for communication
909	.INT_08	INT	free INT variable for communication
910	.INT_09	INT	free INT variable for communication
911	.INT_10	INT	free INT variable for communication
912	.INT_11	INT	free INT variable for communication
913	.INT_12	INT	free INT variable for communication
914	.INT_13	INT	free INT variable for communication
915	.INT_14	INT	free INT variable for communication
916	.INT_15	INT	free INT variable for communication
917	.INT_16	INT	free INT variable for communication
918	.INT_17	INT	free INT variable for communication
919	.INT_18	INT	free INT variable for communication
920	.INT_19	INT	free INT variable for communication
921	.INT_20	INT	free INT variable for communication
922	.INT_21	INT	free INT variable for communication
923	.INT_22	INT	free INT variable for communication
924	.INT_23	INT	free INT variable for communication
925	.INT_24	INT	free INT variable for communication
926	.INT_25	INT	free INT variable for communication
927	.INT_26	INT	free INT variable for communication
928	.INT_27	INT	free INT variable for communication
929	.INT_28	INT	free INT variable for communication
930	.INT_29	INT	free INT variable for communication
931	.INT_30	INT	free INT variable for communication
932	.INT_31	INT	free INT variable for communication
933	.INT_32	INT	free INT variable for communication

Copy of important registers for fast datatransfer

Register	Short designation	Data type	Description
820	.L1_WW	REAL	Active setpoint
822	.L1_WAKT	REAL	Current setpoint
824	.L1_XDIGI	REAL	Digital display PV
826	.L1_D	REAL	Value of D-action
828	.L1_XW	REAL	Control deviation Err
830	.L1_PID_Y_OUT	REAL	Output of PID controller
840	.L2_WW	REAL	Active setpoint
842	.L2_WAKT	REAL	Current setpoint
844	.L2_XDIGI	REAL	Digital display PV
846	.L2_D	REAL	Value of D-action
848	.L2_XW	REAL	Control deviation Err
850	.L2_PID_Y_OUT	REAL	Output of PID controller
860	.L3_WW	REAL	Active setpoint
862	.L3_WAKT	REAL	Current setpoint
864	.L3_XDIGI	REAL	Digital display PV
866	.L3_D	REAL	Value of D-action
868	.L3_XW	REAL	Control deviation Err
870	.L3_PID_Y_OUT	REAL	Output of PID controller
880	.L4_WW	REAL	Active setpoint
882	.L4_WAKT	REAL	Current setpoint
884	.L4_XDIGI	REAL	Digital display PV
886	.L4_D	REAL	Value of D-action
888	.L4_XW	REAL	Control deviation Err
890	.L4_PID_Y_OUT	REAL	Output of PID controller

7.2 MODBUS Coil Table for global variables Boolean

Coil(Status)	Brief designation	Data type	Description
0	.NOCONNECT_B0	BOOL	Binary zero
1	.AE01ERR	BOOL	Error AE01
2	.AE02ERR	BOOL	Error AE02
3-10	unassigned		
11	.AE11ERR	BOOL	Error AE11
12	.AE12ERR	BOOL	Error AE12
13	.AE13ERR	BOOL	Error AE13
14	.AE14ERR	BOOL	Error AE14
15 - 20	unassigned		
21	.AE21ERR	BOOL	Error AE21
22	.AE22ERR	BOOL	Error AE22
23	.AE23ERR	BOOL	Error AE23
24	.AE24ERR	BOOL	Error AE24
25 - 30	unassigned		
31	.AE31ERR	BOOL	Error AE31
32	.AE32ERR	BOOL	Error AE32
33	.AE33ERR	BOOL	Error AE33
34	.AE34ERR	BOOL	Error AE34
35 - 40	unassigned		
41	.AE41ERR	BOOL	Error AE41
42	.AE42ERR	BOOL	Error AE42
43	.AE43ERR	BOOL	Error AE43
44	.AE44ERR	BOOL	Error AE44
45 - 50	unassigned		
51	.AE51ERR	BOOL	Error AE51
52	.AE52ERR	BOOL	Error AE52
53	.AE53ERR	BOOL	Error AE53
54	.AE54ERR	BOOL	Error AE54
55 - 60	unassigned		
61	.AE61ERR	BOOL	Error AE61
62	.AE62ERR	BOOL	Error AE62
63	.AE63ERR	BOOL	Error AE63
64	.AE64ERR	BOOL	Error AE64
65 - 70	unassigned		
71	.AE71ERR	BOOL	Error AE71
72	.AE72ERR	BOOL	Error AE72
73	.AE73ERR	BOOL	Error AE73
74	.AE74ERR	BOOL	Error AE74
75 - 99	unassigned		
99	.AA01BUE	BOOL	Error AA01
100	.AA11BUE	BOOL	Error AA11
101	.AA12BUE	BOOL	Error AA12
102	.AA13BUE	BOOL	Error AA13
103 - 104	unassigned		

Coil(Status)	Brief designation	Data type	Description
105	.AA21BUE	BOOL	Error AA21
106	.AA22BUE	BOOL	Error AA22
107	.AA23BUE	BOOL	Error AA23
108	unassigned		
109	.AA31BUE	BOOL	Error AA31
110	.AA32BUE	BOOL	Error AA32
111	.AA33BUE	BOOL	Error AA33
112	unassigned		
113	.AA41BUE	BOOL	Error AA41
114	.AA42BUE	BOOL	Error AA42
115	.AA43BUE	BOOL	Error AA43
116	unassigned		
117	.AA51BUE	BOOL	Error AA51
118	.AA52BUE	BOOL	Error AA52
119	.AA53BUE	BOOL	Error AA53
120	unassigned		
121	.AA61BUE	BOOL	Error AA61
122	.AA62BUE	BOOL	Error AA62
123	.AA63BUE	BOOL	Error AA63
124	unassigned		
125	.AA71BUE	BOOL	Error AA71
126	.AA72BUE	BOOL	Error AA72
127	.AA73BUE	BOOL	Error AA73
128 - 150	unassigned		

Coil(Status)	Brief designation	Data type	Description
151	.BE01	BOOL	Binary input01
152	.BE02	BOOL	Binary input02
153	.BE03	BOOL	Binary input03
154	.BE04	BOOL	Binary input04
155	.BE11	BOOL	Binary input11
156	.BE12	BOOL	Binary input12
157	.BE13	BOOL	Binary input13
158	.BE14	BOOL	Binary input14
159	.BE15	BOOL	Binary input15
160	.BE16	BOOL	Binary input16
161	.BE21	BOOL	Binary input21
162	.BE22	BOOL	Binary input22
163	.BE23	BOOL	Binary input23
164	.BE24	BOOL	Binary input24
165	.BE25	BOOL	Binary input25
166	.BE26	BOOL	Binary input26
167	.BE31	BOOL	Binary input31
168	.BE32	BOOL	Binary input32
169	.BE33	BOOL	Binary input33
170	.BE34	BOOL	Binary input34
171	.BE35	BOOL	Binary input35
172	.BE36	BOOL	Binary input36
173	.BE41	BOOL	Binary input41
174	.BE42	BOOL	Binary input42
175	.BE43	BOOL	Binary input43
176	.BE44	BOOL	Binary input44
177	.BE45	BOOL	Binary input45
178	.BE46	BOOL	Binary input46
179	.BE51	BOOL	Binary input51
180	.BE52	BOOL	Binary input52
181	.BE53	BOOL	Binary input53
182	.BE54	BOOL	Binäry input54
183	.BE55	BOOL	Binary input55
184	.BE56	BOOL	Binary input56
185	.BE61	BOOL	Binary input61
186	.BE62	BOOL	Binary input62
187	.BE63	BOOL	Binary input63
188	.BE64	BOOL	Binary input64
189	.BE65	BOOL	Binary input65
190	.BE66	BOOL	Binary input66
191	.BE71	BOOL	Binary input71
192	.BE72	BOOL	Binary input72
193	.BE73	BOOL	Binary input73
194	.BE74	BOOL	Binary input74
195	.BE75	BOOL	Binary input75
196	.BE76	BOOL	Binary input76
197 - 220	unassigned		

Coil(Status)	Brief designation	Data type	Description
221	.BA01	BOOL	Binary output01
222	.BA02	BOOL	Binary output02
223	.BA03	BOOL	Binary output03
224	.BA04	BOOL	Binary output04
225	.BA11	BOOL	Binary output11
226	.BA12	BOOL	Binary output12
227	.BA13	BOOL	Binary output13
228	.BA14	BOOL	Binary output14
229	.BA15	BOOL	Binary output15
230	.BA16	BOOL	Binary output16
231	.BA21	BOOL	Binary output21
232	.BA22	BOOL	Binary output22
233	.BA23	BOOL	Binary output23
234	.BA24	BOOL	Binary output24
235	.BA25	BOOL	Binary output25
236	.BA26	BOOL	Binary output26
237	.BA31	BOOL	Binary output31
238	.BA32	BOOL	Binary output32
239	.BA33	BOOL	Binary output33
240	.BA34	BOOL	Binary output34
241	.BA35	BOOL	Binary output35
242	.BA36	BOOL	Binary output36
243	.BA41	BOOL	Binary output41
244	.BA42	BOOL	Binary output42
245	.BA43	BOOL	Binary output43
246	.BA44	BOOL	Binary output44
247	.BA45	BOOL	Binary output45
248	.BA46	BOOL	Binary output46
249	.BA51	BOOL	Binary output51
250	.BA52	BOOL	Binary output52
251	.BA53	BOOL	Binary output53
252	.BA54	BOOL	Binary output54
253	.BA55	BOOL	Binary output55
254	.BA56	BOOL	Binary output56
255	.BA61	BOOL	Binary output61
256	.BA62	BOOL	Binary output62
257	.BA63	BOOL	Binary output63
258	.BA64	BOOL	Binary output64
259	.BA65	BOOL	Binary output65
260	.BA66	BOOL	Binary output66
261	.BA71	BOOL	Binary output71
262	.BA72	BOOL	Binary output72
263	.BA73	BOOL	Binary output73
264	.BA74	BOOL	Binary output74
265	.BA75	BOOL	Binary output75
266	.BA76	BOOL	Binary output76
267 - 289	unassigned		
			positiv flank switches display (Ind) to:
290	.STEPS_B	BOOL	previous display
291	.STEPS_F	BOOL	next display
292	unassigned		
293	.STEPW_F	BOOL	setpoint
294	.SLH_LOOP1	BOOL	channel 1
295	.SLH_LOOP2	BOOL	channel 2
296	.SLH_LOOP3	BOOL	channel 3
297	.SLH_LOOP4	BOOL	channel 4
298	.POS_WW	BOOL	active setpoint
299	.POS_Y	BOOL	correction value
300	.REMOTE	BOOL	remote control via RS-232/485
301	.FLAG_1	BOOL	Display Flag 1
302	.FLAG_2	BOOL	Display Flag 2
303	.FLAG_3	BOOL	Display Flag 3
304	.FLAG_4	BOOL	Display Flag 4
305	.FLAG_5	BOOL	Display Flag 5
306	.FLAG_6	BOOL	Display Flag 6
307 - 308	unassigned		

Coil(Status)	Brief designation	Data type	Description
309	.PG_BETRIEB	BOOL	1 = Programmer is running
310	.MACCOUNT	BOOL	Internal time counter for OM changeover
311	.COMAKTIV	BOOL	1 as long as communication not interrupted
312	.WW_UM	BOOL	Reserved
313	.CAS_TRACK	BOOL	Tracking of master controller if not cascade
314	.PG_RESET	BOOL	Time scheduler reset
315	.PRG_ENDE	BOOL	Binary output, program quit
316	.PRG_BA1	BOOL	Binary output 1 of time scheduler
317	.PRG_BA2	BOOL	Binary output 2 of time scheduler
318	.PRG_BA3	BOOL	Binary output 3 of time scheduler
319	.PRG_BA4	BOOL	Binary output 4 of time scheduler
320	.L1_B1	BOOL	Changeover ES1/ES2 to fixed value ES
321	.L1_A_VORB	BOOL	Automatic prepared
322	.L1_M_VORB	BOOL	Manual prepared
323	.L1_C_VORB	BOOL	Cascade prepared
324	.L1_BETART_UM	BOOL	Input for OM changeover
325	.L1_REGLER_AUTO	BOOL	1 = controller on automatic
326	.L1_REGLER_MAN	BOOL	1 = controller on manual
327	.L1_REGLER_C	BOOL	1 = controller on cascade
328	.L1_HAND_M	BOOL	Step controller output “more”
329	.L1_HAND_W	BOOL	Step controller output “less”
331	.L1_W_STATUS	BOOL	Setpoint status
332	.L1_V_F	BOOL	Status: fixed value/ratio
333	.L1_GW1_OUT	BOOL	Output alarm value transition 1
334	.L1_GW2_OUT	BOOL	Output alarm value transition 2
335	.L1_GW3_OUT	BOOL	Output alarm value transition 3
336	.L1_GW4_OUT	BOOL	Output alarm value transition 4
337	.L1_PID_PS	BOOL	Changeover signal parameter set 1 <--> 2
338	.L1_SPAKTIV	BOOL	1 as long as self-tune active
339	.L1_MAN_AUTO	BOOL	1 if manual or automatic
340	.L1_MAN_CAS	BOOL	1 if manual or cascade
341	.L1_WEXT_AKTIV	BOOL	1 if external setpoint selected
342 - 359	unassigned		
360	.L2_B1	BOOL	Changeover ES1/ES2 to fixed value ES
361	.L2_A_VORB	BOOL	Automatic prepared
362	.L2_M_VORB	BOOL	Manual prepared
363	.L2_C_VORB	BOOL	Cascade prepared
364	.L2_BETART_UM	BOOL	Input for OM changeover
365	.L2_REGLER_AUTO	BOOL	1 = controller on automatic
366	.L2_REGLER_MAN	BOOL	1 = controller on manual
367	.L2_REGLER_C	BOOL	1 = controller on cascade
368	.L2_HAND_M	BOOL	Step controller output “more”
369	.L2_HAND_W	BOOL	Step controller output “less”
371	.L2_W_STATUS	BOOL	Setpoint status
372	.L2_V_F	BOOL	Status: fixed value/ratio
373	.L2_GW1_OUT	BOOL	Output alarm value transition 1
374	.L2_GW2_OUT	BOOL	Output alarm value transition 2
375	.L2_GW3_OUT	BOOL	Output alarm value transition 3
376	.L2_GW4_OUT	BOOL	Output alarm value transition 4
377	.L2_PID_PS	BOOL	Changeover signal parameter set 1 <--> 2
378	.L2_SPAKTIV	BOOL	1 as long as self-tune active
379	.L2_MAN_AUTO	BOOL	1 if manual or automatic
380	.L2_MAN_CAS	BOOL	1 if manual or cascade
381	.L2_WEXT_AKTIV	BOOL	1 if external setpoint selected

Coil(Status)	Brief designation	Data type	Description
382 - 399	unassigned		
400	.L3_B1	BOOL	Changeover ES1/ES2 to fixed value ES
401	.L3_A_VORB	BOOL	Automatic prepared
402	.L3_M_VORB	BOOL	Manual prepared
403	.L3_C_VORB	BOOL	Cascade prepared
404	.L3_BETART_UM	BOOL	Input for OM changeover
405	.L3_REGLER_AUTO	BOOL	1 = controller on automatic
406	.L3_REGLER_MAN	BOOL	1 = controller on manual
407	.L3_REGLER_C	BOOL	1 = controller on cascade
408	.L3_HAND_M	BOOL	Step controller output “more”
409	.L3_HAND_W	BOOL	Step controller output “less”
411	.L3_W_STATUS	BOOL	Setpoint status
412	.L3_V_F	BOOL	Status: fixed value/ratio
413	.L3_GW1_OUT	BOOL	Output alarm value transition 1
414	.L3_GW2_OUT	BOOL	Output alarm value transition 2
415	.L3_GW3_OUT	BOOL	Output alarm value transition 3
416	.L3_GW4_OUT	BOOL	Output alarm value transition 4
417	.L3_PID_PS	BOOL	Changeover signal parameter set 1 <--> 2
418	.L3_SPAKTIV	BOOL	1 as long as self-tune active
419	.L3_MAN_AUTO	BOOL	1 if manual or automatic
420	.L3_MAN_CAS	BOOL	1 if manual or cascade
421	.L3_WEXT_AKTIV	BOOL	1 if external setpoint selected
422 - 439	unassigned		
440	.L4_B1	BOOL	Changeover ES1/ES2 to fixed value ES
441	.L4_A_VORB	BOOL	Automatic prepared
442	.L4_M_VORB	BOOL	Manual prepared
443	.L4_C_VORB	BOOL	Cascade prepared
444	.L4_BETART_UM	BOOL	Input for OM changeover
445	.L4_REGLER_AUTO	BOOL	1 = controller on automatic
446	.L4_REGLER_MAN	BOOL	1 = controller on manual
447	.L4_REGLER_C	BOOL	1 = controller on cascade
448	.L4_HAND_M	BOOL	Step controller output “more”
449	.L4_HAND_W	BOOL	Step controller output “less”
451	.L4_W_STATUS	BOOL	Setpoint status
452	.L4_V_F	BOOL	Status: fixed value/ratio
453	.L4_GW1_OUT	BOOL	Output alarm value transition 1
454	.L4_GW2_OUT	BOOL	Output alarm value transition 2
455	.L4_GW3_OUT	BOOL	Output alarm value transition 3
456	.L4_GW4_OUT	BOOL	Output alarm value transition 4
457	.L4_PID_PS	BOOL	Changeover signal parameter set 1 <--> 2
458	.L4_SPAKTIV	BOOL	1 as long as self-tune active
459	.L4 MAN_AUTO	BOOL	1 if manual or automatic
460	.L4_MAN_CAS	BOOL	1 if manual or cascade
461	.L4_WEXT_AKTIV	BOOL	1 if external setpoint selected

7.3 New variable introduced with library version 3.6

Coil(Status)	Brief designation	Data type	Description
220	.L1_SCAL_LO	REAL	Lower control loop scaling
222	.L1_SCAL_HI	REAL	Upper control loop scaling
244	.L1_ANA_LO	REAL	Lower bargraph scaling
246	.L1_ANA_HI	REAL	Upper bargraph scaling
342	.L1_SETZ_MAN	BOOL	Change-over to MANUAL mode
343	.L1_SETZ_AUTO	BOOL	Change-over to AUTOMATIC mode
344	.L1_SETZ_CASC	BOOL	Change-over to CASCADE mode
1048	.L1_SETZ_W	INT	Select setpoint source
942	.L1_K5	REAL	Evaluation factor K5
944	.L1_K6	REAL	Evaluation factor K6
946	.L1_K7	REAL	Evaluation factor K7
948	.L1_K8	REAL	Evaluation factor K8
950	.L1_K9	REAL	Evaluation factor K9
952	.L1_K10	REAL	Evaluation factor K10
954	.L1_K11	REAL	Evaluation factor K11
956	.L1_K12	REAL	Evaluation factor K12
958	.L1_K13	REAL	Evaluation factor K13
960	.L1_K14	REAL	Evaluation factor K14
962	.L1_K15	REAL	Evaluation factor K15
964	.L1_K16	REAL	Evaluation factor K16
370	.L2_SCAL_LO	REAL	Lower control loop scaling
372	.L2_SCAL_HI	REAL	Upper control loop scaling
394	.L2_ANA_LO	REAL	Lower bargraph scaling
396	.L2_ANA_HI	REAL	Upper bargraph scaling
342	.L2_SETZ_MAN	BOOL	Change-over to MANUAL mode
343	.L2_SETZ_AUTO	BOOL	Change-over to AUTOMATIC mode
344	.L2_SETZ_CASC	BOOL	Change-over to CASCADE mode
1049	.L2_SETZ_W	INT	Select setpoint source
966	.L2_K5	REAL	Evaluation factor K5
968	.L2_K6	REAL	Evaluation factor K6
970	.L2_K7	REAL	Evaluation factor K7
972	.L2_K8	REAL	Evaluation factor K8
974	.L2_K9	REAL	Evaluation factor K9
976	.L2_K10	REAL	Evaluation factor K10
978	.L2_K11	REAL	Evaluation factor K11
980	.L2_K12	REAL	Evaluation factor K12
982	.L2_K13	REAL	Evaluation facto K13r
984	.L2_K14	REAL	Evaluation factor K14
986	.L2_K15	REAL	Evaluation factor K15
988	.L2_K16	REAL	Evaluation factor K16
520	.L3_SCAL_LO	REAL	Lower control loop scaling
522	.L3_SCAL_HI	REAL	Upper control loop scaling
544	.L3_ANA_LO	REAL	Lower bargraph scaling
546	.L3_ANA_HI	REAL	Upper bargraph scaling
422	.L3_SETZ_MAN	BOOL	Change-over to MANUAL mode
423	.L3_SETZ_AUTO	BOOL	Change-over to AUTOMATIC mode
424	.L3_SETZ_CASC	BOOL	Change-over to CASCADE mode
1050	.L3_SETZ_W	INT	Select setpoint source
990	.L3_K5	REAL	Evaluation factor K5
992	.L3_K6	REAL	Evaluation factor K6
994	.L3_K7	REAL	Evaluation factor K7
996	.L3_K8	REAL	Evaluation factor K8
998	.L3_K9	REAL	Evaluation factor K9
1000	.L3_K10	REAL	Evaluation factor K10
1002	.L3_K11	REAL	Evaluation factor K11
1004	.L3_K12	REAL	Evaluation factor K12
1006	.L3_K13	REAL	Evaluation factor K13
1008	.L3_K14	REAL	Evaluation factor K14
1010	.L3_K15	REAL	Evaluation factor K15
1012	.L3_K16	REAL	Evaluation factor K16

Coil(Status)	Brief designation	Data type	Description
670	.L4_SCAL_LO	REAL	Lower control loop scaling
672	.L4_SCAL_HI	REAL	Upper control loop scaling
694	.L4_ANA_LO	REAL	Lower bargraph scaling
696	.L4_ANA_HI	REAL	Upper bargraph scaling
462	.L4_SETZ_MAN	BOOL	Change-over to MANUAL mode
463	.L4_SETZ_AUTO	BOOL	Change-over to AUTOMATIC mode
464	.L4_SETZ_CASC	BOOL	Change-over to CASCADE mode
1051	.L4_SETZ_W	INT	Select setpoint source
1014	.L4_K5	REAL	Evaluation factor K5
1016	.L4_K6	REAL	Evaluation factor K6
1018	.L4_K7	REAL	Evaluation factor K7
1020	.L4_K8	REAL	Evaluation factor K8
1022	.L4_K9	REAL	Evaluation factor K9
1024	.L4_K10	REAL	Evaluation factor K10
1026	.L4_K11	REAL	Evaluation factor K11
1028	.L4_K12	REAL	Evaluation factor K12
1030	.L4_K13	REAL	Evaluation factor K13
1032	.L4_K14	REAL	Evaluation factor K14
1034	.L4_K15	REAL	Evaluation factor K15
1036	.L4_K16	REAL	Evaluation factor K16
1038	.RTC_DATUM	DINT	Date and time [s]
1040	.RTC_ZEIT	DINT	Time [msec]
1054	.RTC_ERROR	INT	Clock error
1053	.RTC_STATUS	INT	Clock state
308	.SETZ_DATUM	BOOL	Set time
1042	.NEU_DATUM	DINT	Sync. time
934	.MOD0ERR	INT	Error of basic I/O unit
935	.MOD1ERR	INT	Error in module 1
936	.MOD2ERR	INT	Error in module 2
937	.MOD3ERR	INT	Error in module 3
938	.MOD4ERR	INT	Error in module 4
939	.MOD5ERR	INT	Error in module 5
940	.MOD6ERR	INT	Error in module 6
941	.MOD7ERR	INT	Error in module 7
307	.DPAKTIV	BOOL	Profibus DP communication is running
1044	.PG_NLAUF	DINT	Net run time of active program
1046	.PG_SEGZEIT	DINT	Segment run time of program
1052	.PG_ZYKLEN	INT	Processed repetition of program

8 Appendix 2

All the following examples for access to registers are in C, in order to ensure an exact and error-free example. Here transmission is effected in the RTU protocol. The functions modbus_read() and modbus_write() show how a telegram is structured, while all others explain handling of various data formats.

modbus_read

Fetch data from other Modbus subscribers (Read Output Register: Function 03)

```
void modbus_read(unsigned regnr, int anzahl, int *recdata)
{
     int            i,anz;
     unsigned       crc;

     sendbuf[0] = mod_adr;   /* MODBUS target address  */
     sendbuf[1] = 3;         /* Read Output Register   */
     sendbuf[2] = regnr>>8;  /* Hi Register Number     */
     sendbuf[3] = regnr;     /* Lo Register Number     */
     sendbuf[4] = 0;         /* Hi Number of registers */
     sendbuf[5] = anzahl;    /* Lo Number of registers */
     crc        = CRC16(sendbuf,6);
     sendbuf[6] = crc;
     sendbuf[7] = crc>>8;

     ComWrite(sendbuf,8);    /* Send 8 characters*/
     ComRead(receivebuf);    /* Receive data */

     // receivebuf[0];   Contains address
     // receivebuf[1];   Contains function code

     anz = receivebuf[2];    /*  Number of data bytes */

     // receivebuf[3+anz]; Contains address CRC
     // receivebuf[4+anz]; Contains address CRC

     for (i=0; i < anz; i+=2) {
          recdata[i+0] = receivebuf[4+i];
          recdata[i+1] = receivebuf[3+i];
     }
}
```

modbus_write

Send data to other Modbus subscribers (Write Single Register: Function 06)

```
void modbus_write(unsigned regnr, int data)
{
     unsigned       crc;

     sendbuf[0] = mod_adr;   /* MODBUS target address */
     sendbuf[1] = 6;         /* Write Single Register */
     sendbuf[2] = regnr>>8;  /* Hi Register Number    */
     sendbuf[3] = regnr;     /* Lo Register Number    */
     sendbuf[4] = data>>8;   /* Hi Data byte          */
     sendbuf[5] = data;      /* Lo Data byte          */
     crc        = CRC16(sendbuf,6);
     sendbuf[6] = crc;
     sendbuf[7] = crc>>8;
     ComWrite(sendbuf,8);    /* Send 8 characters */
     ComRead(receivebuf);    /* Receive acknowledgement*/
}
```

Programming example for determination of CRC sum of MODBUS-RTU telegram

```
unsigned short CRC16(
     void *data_p    /* Data range  */,
     unsigned len    /* Data length */
)
/* Compute 16 Bit CRC (MODBUS-RTU) of data_p */
{

#   define POLYNOM   0x0A001

     int             i,j;
     unsigned short  crc = 0xffff;
     unsigned char   *p = data_p;

     for (j=0; j < len; j++) {   /* for total buffer */
         for (crc ^=  *p++,i=0;  i < 8; i++) {  /* for one Byte */
             if ((crc & 0x0001))
                 crc = (crc >> 1) ^ POLYNOM;
             else
                 crc >>= 1;
         }
     }
     return (crc);
}
```

Determine control deviation with pair of register in loop1 (L1_XW, Register 170)

```
void read_float_split_merge()
{
     float   *fval;
     int     recdata[30];

     modbus_read(170, 2, &recdata[0]);
     fval = (void *)&recdata[0];
     printf("Float-Register 170/171 :  float =%6.3f", *fval);
}
```

Determine control deviation with pair of register in Loop1 (L1_XW, Register 170/171)

```
void read_float_split_merge()
{
     float   *fval;
     int     recdata[30];

     modbus_read(170, 1, &recdata[0]);
     modbus_read(171, 1, &recdata[1]);
     fval = (void *)&recdata[0];
     printf("Float-Register 170/171 :  float =%6.3f", *fval);
}
```

Determine control deviation acc. to exp & mantissa in Loop 1 (L1_XW, Register 2170)

```
void read_float_mantisse_exp()
{
     float   fval;
     int      recdata,i;
     int      man,exp;

     modbus_read(2170, 1, &recdata);
     man = recdata;

     modbus_read(2171, 1, &recdata);
     exp = recdata;
     fval = man;
     fval = fval / 10000.0;
     for(i=0;i < exp; i++)
         fval *= 10.;
     printf("Float-Register 2170/2171 :  float =%6.3f", fval);
}
```

Determine number of current program (Register 802)

```
void read_int()
{
     int recdata;

     modbus_read(802, 1, &recdata);
     printf("Integer-Register 802 :  int =%d", recdata);
}
```

Keyboard intervention: set manual/automatic/cascade (Register 900)

```
void write_int()
{
     modbus_write(900, 0x10);
}
```

Write online parameters, device, table 1, checkpoint 1 with pair of registers (Register 10022/23)

```
void write_float_split_merge()
{
     int             data[2];
     unsigned long   *pdata;
     float           value;

     value   = 133.5;
     pdata   = (void *)&value
     data[0] = (unsigned)(*pdata & 0xFFFF);
     data[1] = (unsigned)(*pdata >>16);
     modbus_write(10022,data[0]);
     modbus_write(10023,data[1]);
}
```

Write online parameters, device, table 1, checkpoint 1 with exponent/mantissa (Register 20022/23)

```
void write_float_mantisse_exp()
{
     float           value;
     int             exp,man;
     ^
     value = 133.5;

     exp  = 0;
     while (fabs(value) >= 1.0 ) {
         value = value/ 10;
         exp++;
     }
     value = value * 10000.0;
     // Observe rounding-off error
     if (wert > 0)
         wert = value + 0.5;
     else
         wert = value - 0.5;
     man = (int)value;

     modbus_write(20022,man);    // first mantissa
     modbus_write(20023,exp);    // then exponent
}
```

Write time scheduler program 1, run time 1 (P17), long value (Reg 15034/35)

```
void write_long_split_merge()
{
     int             data[2];
     unsigned long   *pdata;
     long            value;

     value = 80000l;                    /* 80000 seconds */
     pdata   = (void *)&value;
     data[0] = (unsigned)(*pdata & 0xFFFF);
     data[1] = (unsigned)(*pdata >>16);
     modbus_write(15034,data[0]);
     modbus_write(15035,data[1]);
}
```

9 Appendix 3

9.1 Programming examples in Quickbasic 4.5

9.1.1 IEEE value computation with MKS$ and CSV function

```
'Demo program for processing IEEE value representation
'in Quick-Basic 4.5
'use the Quick-Basic functions MKS$ and CVS
'-------------------------------------------------------
DECLARE FUNCTION BINAER$ (z$)
DECLARE FUNCTION HEX2$ (x)
CLS
DO
     INPUT "Realvalue (E = End) "; Realvalue$
     IF UCASE$(Realvalue$) = "E" THEN END
     Realvalue! = VAL(Realvalue$)
'-------------------------------------------------------
'Pocess:
'-------------------------------------------------------
'    Realvalue in IEEE representation
     IEEE$ = MKS$(Realvalue!)             '4 Byte-String
     FOR I = 0 TO 3
         Byte(I) = ASC(MID$(IEEE$, I + 1, 1))
     NEXT
     Date0& = Byte(1) * 256 + Byte(0)
     Date1& = Byte(3) * 256 + Byte(2)
     'These 2 words must be properly incorporated into the send telegram.

'-------------------------------------------------------
'Control representations
     IEEE$ = HEX2$(Byte(3)) + HEX2$(Byte(2))
     IEEE$ = IEEE$ + HEX2$(Byte(1)) + HEX2$(Byte(0))
     PRINT IEEE$; " ="; BINAER$(IEEE$)
'=======================================================
'Recompute
'-------------------------------------------------------

'Bytes(0) to Byte(3) have been received
'------------------------------------------------------
     IEEEHEX$ = ""
     FOR I = 0 TO 3
         IEEEHEX$ = IEEEHEX$ + CHR$(Byte(I))
     NEXT
     Realvalue! = CVS(IEEEHEX$)
     PRINT "Recomputation = "; Realvalue!
     LOOP
     '----------------------------------------
'Conversion of a hex number in binary representation
'---------------------------------------------------
FUNCTION BINAER$ (z$)
DEFINT A-Z
FOR I = 1 TO LEN(z$)
     x1$ = ""
     x% = VAL("&H" + MID$(z$, I, 1))
     DO UNTIL x% = 0
         Y$ = LTRIM$(STR$(x% MOD 2))
         x% = x% \ 2
         x1$ = Y$ + x1$
     LOOP
     x1$ = RIGHT$("0000" + x1$, 4)
     x$ = x$ + " " + x1$
NEXT
BINAER$ = x$
END FUNCTION
'-----------------------------------------
DEFSNG A-Z
'Represents hex numbers as two digits
'-----------------------------------------
FUNCTION HEX2$ (x)
     HEX2$ = RIGHT$("00" + HEX$(x), 2)
END FUNCTION
```

9.1.2 IEEE value computation without special functions

```
'Demo program for processing IEEE value representation
'in Quick-Basic 4.5                              Version 1.0
'-------------------------------------------------------

DECLARE FUNCTION BINAER$ (z$)
CLS
DO UNTIL i = 127
     INPUT "Realvalue (e = end) "; Realvalue
     IF UCASE$(Realvalue$) = "E" THEN END
     Realvalue! = VAL(Realvalue$)
'----------------------------------------------------------
'Process:
'==========================================================
'Separate sign
     Sign = 0
     IF Realvalue! < 0 THEN
        Realvalue! = Realvalue! * (-1)
        Sign = -1
     END IF
'----------------------------------------------------------
'Determine exponent
     Exponent% = 0
     X! = Realvalue!
     IF X! > 1 THEN
        DO UNTIL X! < 1
           X! = X! / 2
           Exponent% = Exponent% + 1
     LOOP
     Exponent% = Exponent% - 1
ELSE
     DO UNTIL X! > 1
        X! = X! * 2
        Exponent% = Exponent% - 1
     LOOP
     PRINT Exponent%

END IF

'----------------------------------------------------------
'Determine mantissa
     Mantissa = Realvalue! * (2 ^ (23 - Exponent%))
     Mantissa = Mantissa AND &H7FFFFF
'----------------------------------------------------------
'Dtermine words and bytes or for telegram
     Exponent% = (Exponent% + &H7F) * 128

     Date0 = Mantissa MOD &H10000
     Date1 = Mantissa \ &H10000 + Exponent%

     Byte(0) = Date0 MOD 256
     Byte(1) = Date0 \ 256
     Byte(2) = Date1 MOD 256
     Byte(3) = Date1 \ 256 + ((-1) * sign) * &H80
'-------------------------------------------------------
'Control representation
     PRINT "IEEE-Value: ";
     FOR i = 3 TO 0 STEP -1
        PRINT BINAER$(HEX$(Byte(i)));
     NEXT
     PRINT
'=======================================================
'Recompute
'-------------------------------------------------------
'Bytes(0) to Byte(3) have been received
'-------------------------------------------------------
'Sign is encoded in bit 7 of byte(3)
```

```
Sign = 1
IF (Byte(3) AND &H80) = &H80 THEN sign = -1
'-------------------------------------------------------
'Determine exponent from bits 6 to 0 from byte(3)
'and bit 8 from byte(2)
Exponent = (Byte(3) AND &H7F) * 2 + (Byte(2) \ 128)

'-------------------------------------------------------
'Determine mantissa:
'Set bit 7 of byte(3),
'Compute mantissa from byte(0) to byte(3)
Mantissa = (Byte(2) OR &H80) * &H10000
Mantissa = Mantissa + Byte(1) * &H100 + Byte(0)
'-------------------------------------------------------
Realvalue! = sign * Mantissa / (2 ^ (23 - (Exponent - &H7F)))
PRINT "Recomputation = "; Realvalue!

LOOP
'---------------------------------------------------
'Conversion of hex number in binary representation
'---------------------------------------------------
FUNCTION BINAER$ (z$)
DEFINT A-Z

FOR i = 1 TO LEN(z$)
     x1$ = ""
     X% = VAL("&H" + MID$(z$, i, 1))
     DO UNTIL X% = 0
        Y$ = LTRIM$(STR$(X% MOD 2))
        X% = X% \ 2
        x1$ = Y$ + x1$
     LOOP
     x1$ = RIGHT$("0000" + x1$, 4)
     X$ = X$ + " " + x1$
NEXT

BINAER$ = X$

END FUNCTION
```

9.1.3 Computed examples

Exponent on basis 2 is computed through multiple multiplication with 2 or division by 2 such that a 24-digit binary value with a 1 as the highest (left) digit is obtained.

In the IEEE representation, this "1" is suppressed.

```
decim.hexadecimal  binary
                   s/Exponent /value                       /

-1.0  BF 80 00 00 1011 1111 1000 0000 0000 0000 0000 0000
-0.5  BF 00 00 00 1011 1111 0000 0000 0000 0000 0000 0000
-0.4  BE CC CC CD 1011 1110 1100 1100 1100 1100 1100 1101
-0.3  BE 99 99 9A 1011 1110 1001 1001 1001 1001 1001 1010
-0.2  BE 4C CC CD 1011 1110 0100 1100 1100 1100 1100 1101
-0.1  BD CC CC CD 1011 1101 1100 1100 1100 1100 1100 1101
 0.0  00 00 00 00 0000 0000 0000 0000 0000 0000 0000 0000
 0.1  3D CC CC CD 0011 1101 1100 1100 1100 1100 1100 1101
 0.2  3E 4C CC CD 0011 1110 0100 1100 1100 1100 1100 1101
 0.2  3E 99 99 9A 0011 1110 1001 1001 1001 1001 1001 1010
 0.4  3E CC CC CD 0011 1110 1100 1100 1100 1100 1100 1101
 0.5  3F 00 00 00 0011 1111 0000 0000 0000 0000 0000 0000
 1.0  3F 80 00 00 0011 1111 1000 0000 0000 0000 0000 0000
10.0  41 20 00 00 0100 0001 0010 0000 0000 0000 0000 0000
```

9.1.4 Computation of CRC sum

```
'Basic program for determination of the CRC checksum for Modbus
'RTU Telegrams
'Quickbasic 4.5                               Version 1.0
'-----------------------------------------------------------
DECLARE FUNCTION HEX2$ (x!)
CLS
MaxI = 2
PRINT "Enter telegram bytes in Hex 05H or decimal 5"
PRINT "consistently separated by blank or point"
DO
INPUT Tel$
i = 1
L = LEN(Tel$)
Tel$ = UCASE$(Tel$)
     DO UNTIL Tel$ = ""
          Tel$ = LTRIM$(Tel$)
          x = INSTR(Tel$, " ") + INSTR(Tel$, ",")
          IF x > 4 THEN Error = 1: EXIT DO
          IF x > 0 THEN
              Byte$(i) = LEFT$(Tel$, x)
              TEl$ = RIGHT$(Tel$, LEN(Tel$) - x + 1)
          ELSE
              Byte$(i) = Tel$
              Tel$ = ""
          END IF
          Byte$(i) = RTRIM$(Byte$(i))
          IF RIGHT$(Byte$(i), 1) <> "H" THEN
              Byte$(i) = HEX2$(VAL(Byte$(i)))
          ELSE
              Byte$(i) = LEFT$(Byte$(i), 2)
          END IF
          IF HEX2$(VAL("&H" + Byte$(i))) <> Byte$(i) THEN Error = 1: EXIT DO
          i = i + 1
     LOOP

     IF Error = 0 THEN EXIT DO
     SOUND 1000, .03
LOOP

MaxI = i - 1

x& = 65535
FOR i = 1 TO MaxI
     y& = (VAL("&H" + Byte$(i)) XOR x&)
     n = 1
     DO
          DO
              r = y& MOD 2
              y& = y& - r
              y& = y& / 2
              IF ABS(r) = 1 THEN EXIT DO
              n = n + 1
              IF (n = 9) AND (i = MaxI) THEN EXIT FOR
              IF n = 9 THEN EXIT DO
          LOOP
          IF n < 9 THEN
              y& = y& XOR 40961
              n = n + 1
          END IF
          IF n = 9 THEN
              IF (i = MaxI) THEN EXIT FOR
              EXIT DO
          END IF
     LOOP
     x& = y&
NEXT
PRINT "CRC ="; HEX$(y&); " Hex"
PRINT " must be entered in the order "; HEX2$(y& MOD 256); " "; HEX2$(y& \ 256);
PRINT " into the telegram !"

FUNCTION HEX2$ (x)
HEX2$ = RIGHT$("00" + HEX$(x), 2)
END FUNCTION
```

ABB Automation Products GmbH
Hoeseler Platz 2
D-42579 Heiligenhaus
phone +49(0)20 56 12 - 5181
Fax +49(0)20 56 12 - 5081
http://www.abb.com

Subject to technical changes
Printed in the Fed. Rep. of Germany
42/62-50040 EN Rev. 04
Edition 11.01