11 Communications with Modbus RTU

Contents

11.1 Protocol description

The Modbus protocol is used worldwide. The **MODICON Modbus® RTU** protocol is implemented in the AC500 CPU.

Numerous automation devices, such as PLC installations, displays, variable-frequency inverters or monitoring systems have a Modbus® RTU interface by default or as an option and can therefore communicate with AC500 basic units without any problems.

Modbus® is a master-slave protocol. The master sends a request to the slave and receives its response.

Modbus master

In operating mode MODBUS master, the telegram traffic with the slave(s) is handled via the function block MODMAST. The function block MODMAST sends Modbus request telegrams to the slave via the set interface and receives Modbus response telegrams from the slave via this interface.

For Modbus on TCP/IP, the function block ETH_MOD_MAST is used and for serial interfaces the function block COM_MODMAST (link to function blocks: ETH_MODMAST in library Ethernet_AC500_Vxx.lib and COM_MODMAST in library Modbus_AC500_Vxx.lib).

The Modbus® blocks transferred by the master contain the following information:

- Modbus® address of the interrogated slave (1 byte)
- Function code that defines the request of the master (1 byte)
- Data to be exchanged (n bytes)
- CRC16 control code (2 bytes)

Modbus slave

In operating mode MODBUS slave, no function block is required for Modbus communication. Sending and receiving Modbus telegrams is performed automatically.

The AC500 CPUs process only the following Modbus® operation codes:

Function code		**Description**
DEC	**HEX**	
01 or 02	01 or 02	read n bits
03 or 04	03 or 04	read n words
05	05	write one bit
06	06	write one word
07	07	fast reading the status byte of the CPU

15	0F	write n bits
16	10	write n words

The following restrictions apply to the length of the data to be sent:

Function code		Max. length	
DEC	**HEX**	**Serial**	**Modbus on TCP/IP**
01 or 02	01 or 02	2000 bits	255 bits (up to coupler FW V01.033) 1800 bits (as of coupler FW V01.041)
03 or 04	03 or 04	125 words / 62 double words	100 words / 50 double words
05	05	1 bit	1 bit
06	06	1 word	1 word
07	07	8 bits	8 bits
15	0F	1968 bits	255 bits (up to coupler FW V01.033) 1800 bits (as of coupler FW V01.041)
16	10	123 words / 61 double words	100 words / 50 double words

11.2 Modbus RTU with the serial interfaces COM1 and COM2

11.2.1 Modbus operating modes of the serial interfaces

Both serial interfaces of the AC500 CPUs can be operated simultaneously as Modbus interfaces and can operate as Modbus master as well as Modbus slave.

The Modbus operating mode and the interface parameters are set in the PLC Configuration (Link to Controller configuration / Modbus).

Description of the Modbus® protocol:

Supported standard	EIA RS-232 / RS-485 (PM55x and PM56x only support RS-485)
Number of connection points	1 master max. 1 slave with RS 232 interface

	max. 31 slaves with RS 485
Protocol	Modbus® (Master/Slave)
Data transmission control	CRC16
Data transmission speed	up to 187500 baud
Encoding	1 start bit 8 data bits 1 parity bit, even or odd (optional) 1 or 2 stop bits
Max. cable length	for RS 485: 1200 m at 19200 baud

11.3 Modbus on TCP/IP via Ethernet

Modbus on TCP/IP is described in the chapter System Technology Coupler / The Ethernet coupler (Link to System Technology Ethernet Coupler / Modbus on TCP/IP).

11.4 Modbus addresses

11.4.1 Modbus address table

A range of 128 kbytes is allowed for the access via Modbus, i.e., the segments line 0 and line 1 of the addressable flag area (%M area) can be accessed. Thus, the complete address range 0000_{hex} up to $FFFF_{hex}$ is available for Modbus.

The availability of the segments depends on the CPU. The size of the %M area can be found in the technical data of the CPUs (see Technical data of the CPUs) and in the target system settings (see Target Support Package).

Inputs and outputs cannot be directly accessed using Modbus.

The address assignment for word and double word accesses is done according to the following table:

Modbus address		**Byte BYTE**	**Bit (byte-oriented) BOOL**	**Word WORD**	**Double word DWORD**
HEX	**DEC**				
Line 0					
0000	0	%MB0.0	%MX0.0.0...%MX0.0.7	**%MW0.0**	**%MD0.0**
		%MB0.1	%MX0.1.0...%MX0.1.7		
0001	1	%MB0.2	%MX0.2.0...%MX0.2.7	**%MW0.1**	

		%MB0.3	%MX0.3.0...%MX0.3.7		
0002	2	%MB0.4	%MX0.4.0...%MX0.4.7	**%MW0.2**	**%MD0.1**
		%MB0.5	%MX0.5.0...%MX0.5.7		
0003	3	%MB0.6	%MX0.6.0...%MX0.6.7	**%MW0.3**	
		%MB0.7	%MX0.7.0...%MX0.7.7		
...					
7FFE	32766	%MB0.65532	%MX0.65532.0...%MX0.65532.7	**%MW0.32766**	**%MD0.16383**
		%MB0.65533	%MX0.65533.0...%MX0.65533.7		
7FFF	32767	%MB0.65534	%MX0.65534.0...%MX0.65534.7	**%MW0.32767**	
		%MB0.65535	%MX0.65535.0...%MX0.65535.7		
Line 1					
8000	32768	%MB1.0	%MX1.0.0...%MX1.0.7	**%MW1.0**	**%MD1.0**
		%MB1.1	%MX1.1.0...%MX1.1.7		
8001	32769	%MB1.2	%MX1.2.0...%MX1.2.7	**%MW1.1**	
		%MB1.3	%MX1.3.0...%MX1.3.7		
8002	32770	%MB1.4	%MX1.4.0...%MX1.4.7	**%MW1.2**	**%MD1.1**
		%MB1.5	%MX1.5.0...%MX1.5.7		

8003	32771	%MB1.6	%MX1.6.0...%MX1.6.7	**%MW1.3**	
		%MB1.7	%MX1.7.0...%MX1.7.7		
...					
FFFE	65534	%MB1.65532	%MX1.65532.0 ...%MX1.65532.7	**%MW1.32766**	**%MD1.16383**
		%MB1.65533	%MX1.65533.0 ...%MX1.65533.7		
FFFF	65535	%MB1.65534	%MX1.65534.0 ...%MX1.65534.7	**%MW1.32767**	
		%MB1.65535	%MX1.65535.0 ...%MX1.65535.7		

The **address assignment for bit accesses** is done according to the following table:

Modbus address		**Byte BYTE**	**Bit (byte-oriented) BOOL**	**Word WORD**	**Double word DWORD**
HEX	**DEC**				
Line 0					
0000	0	%MB0.0	**%MX0.0.0**	%MW0.0	%MD0.0
0001	1		**%MX0.0.1**		
0002	2		**%MX0.0.2**		
0003	3		**%MX0.0.3**		
0004	4		**%MX0.0.4**		
0005	5		**%MX0.0.5**		
0006	6		**%MX0.0.6**		
0007	7		**%MX0.0.7**		
0008	8	%MB0.1	**%MX0.1.0**		
0009	9		**%MX0.1.1**		
000A	10		**%MX0.1.2**		

000B	11		**%MX0.1.3**		
000C	12		**%MX0.1.4**		
000D	13		**%MX0.1.5**		
000E	14		**%MX0.1.6**		
000F	15		**%MX0.1.7**		
0010	16	%MB0.2	**%MX0.2.0**	%MW0.1	
0011	17		**%MX0.2.1**		
0012	18		**%MX0.2.2**		
0013	19		**%MX0.2.3**		
0014	20		**%MX0.2.4**		
0015	21		**%MX0.2.5**		
0016	22		**%MX0.2.6**		
0017	23		**%MX0.2.7**		
0018	24	%MB0.3	**%MX0.3.0**		
0019	25		**%MX0.3.1**		
001A	26		**%MX0.3.2**		
001B	27		**%MX0.3.3**		
001C	28		**%MX0.3.4**		
001D	29		**%MX0.3.5**		
001E	30		**%MX0.3.6**		
001F	31		**%MX0.3.7**		
0020	32	%MB0.4	**%MX0.4.0**	%MW0.2	%MD0.1
0021	33		**%MX0.4.1**		
0022	34		**%MX0.4.2**		

...	...	...	**...**	...	...
0FFF	4095	%MB0.511	**%MX0.511.7**	%MW0.255	%MD0.127
1000	4096	%MB0.512	**%MX0.512.0**	%MW0.256	%MD0.128
...	...	...	**...**	...	...
7FFF	32767	%MB0.4095	**%MX0.4095.7**	%MW0.2047	%MD0.1023
8000	32768	%MB0.4096	**%MX0.4096.0**	%MW0.2048	%MD0.1024
...	...	...	**...**	...	...
FFFF	65535	%MB0.8191	**%MX0.8191.7**	%MW0.4095	%MD0.2047

Calculation of the bit variable from the hexadecimal address:

Formula:			
	Bit variable (BOOL) := %MX0.BYTE.BIT		
where:	DEC	Decimal address	
	BYTE	DEC / 8	
	BIT	DEC mod 8	(Modulo division)

Examples:

Address hexadecimal = 16#2002
DEC := HEX2DEC(16#2002) := 8194
BYTE := 8194 / 8 := 1024
BIT := 8194 mod 8 := 2
Bit variable: %MX0.1024.2

Address hexadecimal = 16#3016
DEC := HEX2DEC(16#3016) := 12310
BYTE := 12310 / 8 := 1538,75 -> 1538
BIT := 12310 mod 8 := 6
Bit variable: %MX0.1538.6

Address hexadecimal = 16#55AA
DEC := HEX2DEC(16#55AA) := 21930
BYTE := 21930 / 8 := 2741,25 -> 2741
BIT := 21930 mod 8 := 2
Bit variable: %MX0.2741.2

Calculation of the hexadecimal address from the bit variable:

Formula:

Address hexadecimal := DEC2HEX(BYTE * 8 + BIT)

Examples:

Bit variable := %MX0.515.4
Address hex := DEC2HEX(515 * 8 + 4) := DEC2HEX(4124) := 16#101C

Bit variable := %MX0.3.3
Address hex := DEC2HEX(3 * 8 + 3) := DEC2HEX(27) := 16#001B

Bit variable := %MX0.6666.2
Address hex := DEC2HEX(6666 * 8 + 2) := DEC2HEX(53330) := 16#D052

11.4.2 Peculiarities for accessing Modbus addresses

Peculiarities for bit access:

- As you can see in the address table, a WORD in the %M area is assigned to each Modbus address 0000hex .. FFFFhex
- Bit addresses 0000hex .. FFFFhex are contained in the word range %MW0.0 .. %MW0.4095

Write/read-protected areas for the Modbus slave:

As described in the PLC configuration, one write-protected and one read-protected area can be defined for each segment line 0 and line 1. (Link to Controller configuration / The setting 'COMx - Modbus'). If you try to write to a write-protected area or to read from a read-protected area, an error message is generated.

Segment exceedance for line 0 and line 1:

A write- or read-protected area that lies in both segments, line 0 and line 1, cannot be accessed with a write/read operation. In case of a segment exceedance, an error message is generated.

Example:
Read 10 words beginning at address := 7FFE$_{hex}$
This includes the addresses: 7FFE$_{hex}$...8007$_{hex}$ with the operands %MW0.32766...%MW1.7.
Because line 0 is exceeded in this case, an error message is generated.
Due to this, two telegrams have to be generated here:
1. Read 2 words beginning at address := 7FFE$_{hex}$ and
2. Read 8 words beginning at address := 8000$_{hex}$.

Valid data areas for reading/writing the Modbus master:

If the AC500 control system operates as Modbus master, the data exchange with the Modbus slaves is controlled using a MODMAST block (ETH_MOD_MAST for Modbus on TCP/IP and COM_MOD_MAST for serial interfaces). (Link to blocks: ETH_MODMAST in library Ethernet_AC500_Vxx.lib and COM_MODMAST in library Modbus_AC500_Vxx.lib).

The address of the area from which data are to be read or to which data are to be written is specified at block input "Data" via the ADR operator.

For the AC500, the following areas can be accessed using the ADR operator:

- Inputs area (%I area)

- Outputs area (%Q area)
- Area of non-buffered variables (VAR .. END_VAR or VAR_GLOBAL END_VAR)
- Addressable flag area (also protected areas for %M area)
- Area of buffered variables (VAR RETAIN .. END_VAR or VAR_GLOBAL RETAIN .. END_VAR)

11.4.3 Comparison between AC500 and AC31/S90 Modbus addresses

The following table shows the addresses for AC500 controllers and its predecessor AC31 / S90:

Address HEX	**FCT HEX**	**AC1131 operand**	**FCT HEX**	**AC500 operand**
Bit accesses				
0000...0FFF	01, 02	%IX0.0...%IX255.15	01, 02, 05, 07, 0F	%MX0.0.0...%MX0.511.7
0000		%IX0.0		%MX0.0.0
0001		%IX0.1		%MX0.0.1
0002		%IX0.2		%MX0.0.2
...		...		...
0010		%IX1.0		%MX0.2.0
...		...		...
0FFF		%IX255.15		%MX0.511.7
1000...1FFF	01, 02, 05, 0F	%QX0.0...%QX255.15	01, 02, 05, 07, 0F	%MX0.512.0...%MX0.1023.7
1000		%QX0.0		%MX0.512.0
1001		%QX0.1		%MX0.512.1
1002		%QX0.2		%MX0.512.2
...		...		...
1010		%QX1.0		%MX0.514.0
...		...		...
1FFF		%QX255.15		%MX0.1023.7
2000...2FFF	01, 02, 05, 07, 0F	%MX0.0...%MX255.15	01, 02, 05, 07, 0F	%MX0.1024.0...%MX0.1535.7
2000		%MX0.0		%MX0.1024.0
2001		%MX0.1		%MX0.1024.1
2002		%MX0.2		%MX0.1024.2
...		...		...
2010		%MX1.0		%MX0.1026.0

...		...		...
2FFF		%MX255.15		%MX0.1535.7
3000...3FFF	01, 02, 05, 07, 0F	%MX5000.0...%MX5255.15	01, 02, 05, 07, 0F	%MX0.1536.0...%MX0.2047.7
3000		%MX5000.0		%MX0.1536.0
3001		%MX5000.1		%MX0.1536.1
3002		%MX5000.2		%MX0.1536.2
...		...		...
3010		%MX5001.0		%MX0.1538.0
...		...		...
3FFF		%MX5255.15		%MX0.2047.7
4000...FFFF		No access	01, 02, 05, 07, 0F	%MX0.2048.0...%MX0.8191.7
Word accesses				
0000...0CFF	03, 04	%IW1000.0...%IW1207.15	03, 04, 06, 10	%MW0.0...%MW0.3327
0D00...0FFF	03, 04	No access	03, 04, 06, 10	%MW0.3328...%MW0.4095
1000...1CFF	03, 04, 06, 10	%QW1000.0...%QW1207.15	03, 04, 06, 10	%MW0.4096...%MW0.7423
1D00...1FFF		No access	03, 04, 06, 10	%MW0.7424...%MW0.8191
2000...2FFF	03, 04, 06, 10	%MW1000.0...%MW1255.15	03, 04, 06, 10	%MW0.8192...%MW0.12287
3000...359F	03, 04, 06, 10	%MW3000.0...%MW3089.15	03, 04, 06, 10	%MW0.12288...%MW0.13727
35A0...3FFF		No access	03, 04, 06, 10	%MW0.13728...%MW0.16383
4000...47FF		%MW2000.0.0...%MW2063.15.1 No access	03, 04, 06, 10	%MW0.16384...%MW18431
4800...4FFF		No access	03, 04, 06, 10	%MW0.18432...%MW0.20479
5000...517F		%MW4000.0.0...%MW4023.15.1 No access	03, 04, 06, 10	%MW0.20480...%MW0.21247
5180...FFFF		No access	03, 04, 06, 10	%MW0.21248...%MW1.32767
Double word accesses				
0000...3FFF		No access	03, 04, 06, 10	%MD0.0...%MD0.8191
4000...47FF	03, 04, 06, 10	%MD2000.0...%MD2063.15	03, 04, 06, 10	%MD0.8192...%MD0.9215
4800...4FFF		No access	03, 04, 06, 10	%MD0.9216...%MD0.10239

5000...537F	03, 04, 06, 10	%MD4000.0...%MD4023.15	03, 04, 06, 10	%MD0.1240...%MD0.10815
5480...FFFF		No access	03, 04, 06, 10	%MD0.10816...%MD1.16383

11.5 Modbus telegrams

The send and receive telegrams shown in this section are not visible in the PLC. However, the complete telegrams can be made visible using a serial data analyzer connected to the connection line between master and slave, if required.

The amount of user data depends on the properties of the master and slave.

For the following examples, it is assumed that an AC500 Modbus module is used as slave. There may be different properties if modules of other manufacturers are used.

FCT 1 or 2: Read n bits

Master request

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low

Slave response

Slave address	Function code	Number of bytes	...Data...	CRC	
				High	Low

Example :	Modbus interface of the master:	COM1
	Master reads from:	Slave 1
	Data:	%MX0.1026.4 = FALSE; %MX0.1026.5 = TRUE %MX0.1026.6 = FALSE
	Source address at slave:	%MX0.1026.4 : 2014HEX = 8212DEC
	Target address at master:	abReadBit : ARRAY[0..2] OF BOOL;
	The values of the flags %MX0.1026.4..%MX0.1026.6 on the slave are written to the ARRAY abReadBool on the master.	

Modbus request of the master

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low

01HEX	01HEX	20HEX	14HEX	00HEX	03HEX	37HEX	CFHEX

Modbus response of the slave

Slave address	Function code	Number of bytes	Data	CRC	
				High	Low
01HEX	01HEX	01HEX	02HEX	D0HEX	49HEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	1	Application-specific	8212	3	ADR (abReadBool[0])

FCT 3 or 4: Read n words

Master request

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

Slave response

Slave address	Function code	Number of bytes	...Data...	CRC	
				High	Low

Example :	Modbus interface of the master:	COM1
	Master reads from:	Slave 1

	Data:	%MW0.8196 = 4;
		%MW0.8197 = 5;
		%MW0.8198 = 6

	Source address at slave:	%MW0.8196 : 2004HEX = 8196DEC
	Target address at master:	awReadWord : ARRAY[0..2] OF WORD;
	The values of the flag words %MW0.8196..%MW0.8198 on the slave are written to the ARRAY awReadWord on the master.	

Modbus request of the master

Slave	Function	Slave operand address	Number of words	CRC

address	code	High	Low	High	Low	High	Low
01HEX	03HEX	20HEX	04HEX	00HEX	03HEX	4FHEX	CAHEX

Modbus response of the slave

Slave address	Function code	Number of bytes	Data	Data	Data	CRC	
			High / Low	High / Low	High / Low	High	Low
01HEX	03HEX	06HEX	00HEX /04HEX	00HEX /05HEX	00HEX /06HEX	40HEX	B6HEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	3	Application-specific	8196	3	ADR (awReadWord[0])

FCT 3 or 4: Read n double words

The function code "read double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer).

Master request

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

Slave response

Slave address	Function code	Number of bytes	...Data...	CRC	
				High	Low

Example :	Modbus interface of the master:	COM1
	Master reads from:	Slave 1
	Data:	%MD0.8193 = 32DEC = 00000020HEX; %MD0.8194 = 80000DEC = 00013880HEX
	Source address at slave:	%MD0.8193: 4002HEX = 16386DEC
	Target address at master:	adwReadDWord : ARRAY[0..1] OF DWORD
	The values of the flag double words %MD0.8193..%MD0.8194 on the slave are written to the ARRAY adwReadDWord on the master.	

Modbus request of the master

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	03HEX	40HEX	02HEX	00HEX	04HEX	F0HEX	09HEX

Modbus response of the slave

Slave address	Function code	Number of bytes	Data	Data	Data	Data	CRC	
			High / Low	High / Low	High / Low	High / Low	High	Low
01HEX	03HEX	08HEX	00HEX / 00HEX	00HEX / 20HEX	00HEX / 01HEX	38HEX / 80HEX	57HEX	B0HEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	31	Application-specific	16386	4	ADR (adwReadDWord[0])

FCT 5: Write 1 bit

For the function code "write 1 bit", the value of the bit to be written is encoded in one word.

BIT = TRUE -> Data word = FF 00 HEX

BIT = FALSE -> Data word = 00 00 HEX

Master request

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

Slave response

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

Example :	Modbus interface of the master:	COM1
	Master writes to:	Slave 1

	Data:	bBit := TRUE
	Source address at master:	bBit : BOOL;
	Target address at slave:	%MX0.1026.7 : 2017HEX = 8215DEC
	The value of the BOOL variable bBit on the master is written to %MX0.1026.7 on the slave.	

Modbus request of the master

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	05HEX	20HEX	17HEX	FFHEX	00HEX	37HEX	FEHEX

Modbus response of the slave (mirrored)

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	05HEX	20HEX	17HEX	FFHEX	00HEX	37HEX	FEHEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	5	Application-specific	8215	1	ADR (bBit)

FCT 6: Write 1 word

Master request

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

Slave response

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

Example :	Modbus interface of the master:	COM1
	Master writes to:	Slave 1
	Data:	wData := 7

	Source address at master:	wData : WORD;
	Target address at slave:	%MW0.8199 : 2007HEX = 8199DEC
	The value of the WORD variable bBit on the master is written to %MW0.8199 on the slave.	

Modbus request of the master

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	06HEX	20HEX	07HEX	00HEX	07HEX	72HEX	09HEX

Modbus response of the slave (mirrored)

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	06HEX	20HEX	07HEX	00HEX	07HEX	72HEX	09HEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	6	Application-specific	8215	1	ADR (wData)

FCT 7: Fast reading the status byte of the CPU

Master request

Slave address	Function code	CRC					
		High	Low				

Slave response

Slave address	Function code	Data byte	CRC			
			High	Low		

Example :	Modbus interface of the master:	COM1

	Master writes to:	Slave 1
	Data:	
	Source address at slave:	
	Target address at slave:	
	In version V1.x, this function always returns 0!	

Modbus request of the master

Slave address	Function code	CRC					
		High	Low				
01HEX	07HEX	41HEX	E2HEX				

Modbus response of the slave

Slave address	Function code	Data byte	CRC			
			High	Low		
01HEX	07HEX	00HEX	xxHEX	xxHEX		

Parameterization of the COM_MOD_MAST block inputs
NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	7	Application-specific	0	0	ADR (BoolVar)

Note: In version V1.x, function 7 always returns 0!

FCT 15: Write n bits

Master request

Slave address	Function code	Slave operand address		Number of bits		Number of bytes	...Data...	CRC	
		High	Low	High	Low			High	Low

Slave response

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low

Example :	Modbus interface of the master:	COM1
	Master writes to:	Slave 1

	Data:	abWriteBool[0] := TRUE;
		abWriteBool[1] := FALSE;
		abWriteBool[2] := TRUE

	Source address at master:	abWriteBool : ARRAY[0..2] OF BOOL;
	Target address at slave:	%MX0.1026.1 : 2011HEX = 8209DEC
	The values of the BOOL variables abWriteBool[0]..abWriteBool[2] on the master are written to %MX0.1026.1..%MX0.1026.3 on the slave.	

Modbus request of the master

Slave address	Function code	Slave operand address		Number of bits		Number of bytes	Data	CRC	
		High	Low	High	Low			High	Low
01HEX	0FHEX	20HEX	11HEX	00HEX	03HEX	01HEX	05HEX	B4HEX	37HEX

Modbus response of the slave

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low
01HEX	0FHEX	20HEX	11HEX	00HEX	03HEX	4EHEX	0FHEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	15	Application-specific	8209	3	ADR (abWriteBool[0])

FCT 16: Write n words

Master request

Slave address	Function code	Slave operand address		Number of words		Number of bytes	...Data...	CRC	
		High	Low	High	Low			High	Low

Slave response

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

Example :	Modbus interface of the master:	COM1
	Master writes to:	Slave 1

	Data:	awWriteWord[0] := 1;
		awWriteWord[1] := 2;
		awWriteWord[2] := 3

	Source address at master:	awWriteWord : ARRAY[0..2] OF WORD;
	Target address at slave:	%MW0.8193 : 2001HEX = 8193DEC
	The values of the WORD variables awWriteWord[0]..awWriteWord[2] on the master are written to %MW0.8193..%MW0.8195 on the slave.	

Modbus request of the master

Slave address	Function code	Slave operand address	Number of words	Number of bytes	Data	Data	Data	CRC
		High / Low	High / Low		High / Low	High / Low	High / Low	High / Low
01HEX	10HEX	20HEX / 01HEX	00HEX / 03HEX	06HEX	00HEX / 01HEX	00HEX / 02HEX	00HEX / 03HEX	C0HEX / 84HEX

Modbus response of the slave

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	10HEX	20HEX	01HEX	00HEX	03HEX	DAHEX	08HEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	16	Application-specific	8193	3	ADR (awWriteWord[0])

FCT 16: Write n double words

The function code "write double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer).

Master request

Slave address	Function code	Slave operand address		Number of words		Number of bytes	...Data...	CRC	
		High	Low	High	Low			High	Low

Slave response

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

Example :	Modbus interface of the master:	COM1
	Master writes to:	Slave 1

	Data:	adwWriteDWord[0] := 18DEC = 00000012HEX;
		adwWriteDWord[1] := 65561DEC = 00010019HEX;

	Source address at master:	adwWriteDWord : ARRAY[0..1] OF WORD;
	Target address at slave:	%MD0.8192 : 4000HEX = 16384DEC
	The values of the Double WORD variables adwWriteDWord[0].. adwWriteDWord[1] on the master are written to %MD0.8192..%MD0.8193 on the slave.	

Modbus request of the master

Slave address	Function code	Slave operand address	Number of words	Number of bytes	Data	Data	Data	Data	CRC
		High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low
01HEX	10HEX	40HEX / 00HEX	00HEX / 04HEX	00HEX / 08HEX	00HEX / 00HEX	00HEX / 12HEX	00HEX / 01HEX	00HEX / 19HEX	60HEX / B3HEX

Modbus response of the slave

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	10HEX	40HEX	00HEX	00HEX	04HEX	DAHEX	0AHEX

Parameterization of the COM_MOD_MAST block inputs
NB = Number of words = 2 x Number of double words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	16	Application-specific	16384	4	ADR (adwWriteDWord[0])

Error telegram

In operating mode Modbus master, the AC500 does only send telegrams, if the parameters at the MODMAST inputs are logically correct. Nevertheless, it can happen that a slave cannot process the request of the master or that the slave cannot interpret the request due to transmission errors. In those cases, the slave returns an error telegram to the master. In order to identify this telegram as an error telegram, the function code returned by the slave is a logical OR interconnection of the function code received from the master and the value 80HEX.

Slave response

Slave address	Function code OR 80HEX	Error code	CRC	
			High	Low

Possible error codes of the slave

Code	Meaning
01DEC	The slave does not support the function requested by the master
02DEC	Invalid operand address in the slave
02DEC	Operand area exceeded
03DEC	At least one value is outside the permitted value range
12DEC	The amount of data is higher than the slave can process
13DEC	The telegram contains an odd number of words in case of double word access
10DEC	Length specifications in the telegram do not match
11DEC	The type of operand area and the function do not match

Example:

Modbus request of the master:

	Function code:	01	(Read n bits)
	Slave operand address:	$4000_{HEX} = 16384_{DEC}$	(Area for read access disabled in slave)
Modbus response of the slave:			
	Function code:	81_{HEX}	
	Error code:	03	

11.6 Function block COM_MOD_MAST

This function block is only required in the operating mode Modbus master. It handles the communication (transmission of telegrams to the slaves and reception of telegrams from the slaves). The function block can be used for the local interfaces COM1 and COM2 of the controller. A separate instance of the function block has to be used for each interface.

COM_MOD_MAST is contained in the library **Modbus_AC500_V1x.LIB (version V1.0 and later).**