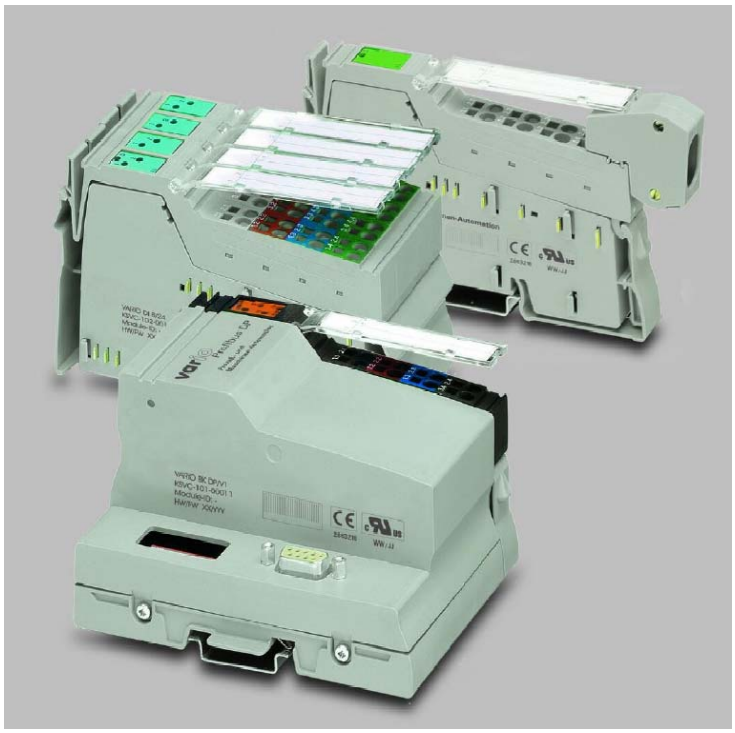
CD AUTOMATION SRL

CD VAR-IO ETHERNET BUSCOUPLER



Interface description

Ethernet 9499 040 69811

07/2004

Content

1. General

Modular controller system vario permits connection of various fieldbus interfaces. For this purpose, the relevant bus coupler is used as a head station for a controller system.
Via one of these bus couplers, the ETHERNET (Modbus/TCP protocol) is supported by means of a front-panel interface (PJ45 connector). Hereby, transmission of all process, parameter and configuration data is possible.
This communication interface permits communication with supervisory systems, visualization tools, etc.
Another standard interface is provided on the KS vario controller modules. This full RS232 interface is used for connection of the 'Control' tool, which runs on a PC.

Transfer rate The Ethernet coupler works as a ModbusTCP server with a maximum transfer rate of 100Mbit.

Clients The Ethernet bus coupler permits communication with up to 4 clients via the TCP/IP protocol.

2. Hints for operation

2.1. Connecting the interface

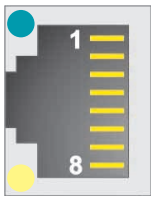
Connect the Ethernet to the front-panel RJ45 interface of the bus coupler.
10BaseT or 100BaseT are used as physical layer.

Physical connection is via Ethernet using twisted two-wire cable (CAT5 cable, 8-pole in RJ-45 connection technology).

Pin allocation RJ-45 Connection is via an RJ-45 socket, with 2 integrated LEDs.

Green LED on: connected to Ethernet
Yellow LED on: traffic on Ethernet

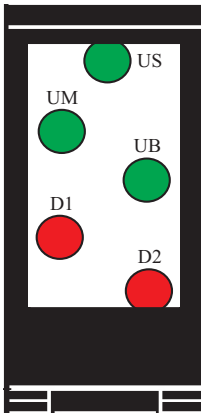
Contact	Signal	Description
1	TD +	Transmit +
2	TD -	Transmit -
3	RD +	Receive +
4	-	unused
5	-	unused
6	RD -	Receive -
7	-	unused
8	-	unused



2.2. Signification of indicator LEDs on the bus coupler

LEDs

LED no.	LED colour	Function
US	green	24 V segment voltage provided
UM	green	24V main supply provided (presently not used)
UB	green	24V coupler voltage provided
D1	red	ON: no connection to client
D2	red	BLINKING: faulty communication OFF: correct communication



2.3. Forcing

Inputs

All physical inputs can be overwritten (configurable) via ETHERNET. Thus e.g. process value measurement via remote I/O (e.g. VARIO I/O system) and entry via the bus are possible.

Outputs

With output forcing, the fail-safe function setting must be taken into account. If "zero" fail-safe behaviour was adjusted, all outputs are set to zero in case of bus error or master stop, otherwise, their old value remains unchanged. .

2.4. Fail-safe

User parameter setting 'fail-safe' determines the instrument behaviour in case of master bus failure or 'bus stop' .
Bus failure
In case of bus failure, the instrument operates according to the following rules.

Fail-safe	Reaction in case of bus failure or master stop
Last value	Continue operation with the values sent last
	Forced analog inputs are set to FAIL
zero	Forced analog inputs are set to FAIL 1) .
	Forced digital inputs are set to zero
	Forced outputs are set to zero

3. Communication via Ethernet

3.1. Physical Layer

10BaseT or 100BaseT is used as physical layer.

3.2. Data link layer Ethernet / MAC-ID

Ethernet transports Ethernet packets from a sender to one or several receivers without acknowledgement and without repetition of lost packets.
Senders and receivers of Ethernet packets are addressed via the MAC-ID. The MAC-ID is a 6-byte ident code, which is unambiguous, i.e. worldwide different for each unit connected to Ethernet. The MAC-ID consists of two parts. The first part, i.e. the first 3 bytes, are a manufacturer identification.
The identification is 00 0E 0D. The next 3 bytes are determined by the manufacturer and correspond to a series number. They are unique. The MAC-ID can be used, for instance, for the BOOTP protocol for adjustment of the TCP/IP number.
For this purpose, a telegram including the information such as name or TCP/IP number is sent to the relevant node.

3.3. Network layer IP

Basis of data communication is the Internet protocol (IP). IP transports data telegrams between communicating units connected in the same or a different network and provides address management (finding and assignment of MAC-Ids), segmenting and routing.

3.4. Transport layer

The IP-based transmission control protocol (TCP) enables two hosts to establish a connection and exchange streams of data. It includes treatment mechanisms. Lost telegrams are repeated.

UDP is a connection-less transport protocol. It does not include control mechanisms during data exchange between sender and receiver.
This results in a higher processing speed than e.g. with TCP. Checking, whether the telegram has arrived must be done by the superordinate protocol.

3.5. Application layer Modbus/TCP

Modbus/TCP is a Modbus communication based on the TCP/IP transmission protocols. Ethernet is used as transmission standard. ModbusTCP follows the client-server model, whereby the ModbusTCP server provides services for clients. Communication is started by a 'request' made by a Modbus TCP client. The server replies this 'request' with an 'indication'. When processing in the client is finished, the server sends a 'response' to the client, which is replied with a 'confirmation'.
Communication via bridges, routers or gateways is possible.
In the Ethernet bus coupler, the Modbus protocol in RTU (Remote Terminal Unit) mode is used, i.e. each transmitted message byte contains two hexadecimal characters (0..9, A..F).
Further information is given in the "Modicon Modbus Protocol Reference Guide" of Modicon, Inc., Industrial Automation Systems. See also chapter 5: "Information on the Modbus protocol".

3.6. IP address of the KS vario system via BootP protocol

IP-address and sub-net mask can be requested with the BootP protocol. After power-on, the BootP protocol is always sent, unless an own IP addr. is known. Known means that it was determined using the "BlueControl" engineering tool .

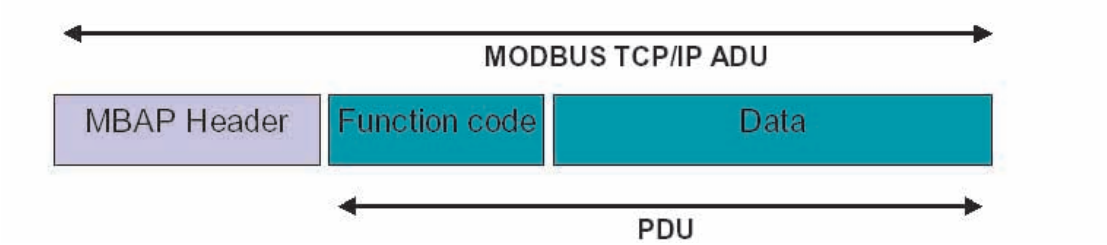
If an IP addr. is known, a change by the client is made possible nevertheless by sending 4 BootP protocols.

3.7. IP address of the KS vario system via "BlueControl"engineering tool

IP address (4 bytes) and sub-net mask (4 bytes) can be entered into KS vario by the BlueControl tool. KS vario transmits this information to the bus coupler during transmission. If the IP address is determined as "0" via the tool, the bus coupler detects this address as invalid and address determination via Boot P becomes relevant.

3.8. Modbus TCP message format (Application Data Unit)

The ADU (Application Data Unit) structure is shown below:



ADU structure

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the Client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the Client	Recopied by the server from the received request
Length	2 Bytes	Number of following Bytes	Initialized by the Client (request)	Initialized by the Server(response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the Client	Recopied by the server from the received request
Function-Code	1 Byte	Modbus RTU Function-Code		
Data	x Bytes	Data		

3.9. Function codes

The following function codes are realized in KS vario:

Function code	Signification
0x03	Read process data, parameters or configuration data
0x04	Read process data, parameters or configuration data
0x06	Write a single datum (process data, parameters or configuration)
0x10	Write several data (process data, parameters or configuration)
0x17	Read/write register Write outputs wordwisely with start address and number of outputs Read inputs wordwisely with start address and number of inputs
0x2B	Read Device Identification Read manufacturer name, product code and software version. For details, see chapter 5 “Information on the Modbus protocol”.

Note: No difference between function codes 03 and 04 is made, because it is not possible to make sure that all masters support both function codes. Function codes 0x17 and 0x2B are supported only by the bus coupler. Thus access to the process data caches is possible. Free access to any other KS vario data is not possible with this function code.

4. Addressing in the Ethernet bus coupler

In addition to the access to all parameters and data via Modbus addresses, access to selected data (max. 1080 per direction) of a cache via the BlueControl tool is possible.

Addressing is done via the unit identifier data field of the MBAP header.

Unit Identifier = 0 or 1 --> access to data of the cache memory in the bus coupler
Unit Identifier = 2..255 --> access to KS vario data

4.1. Access to data of the cache memory in the bus coupler

Access using Unit Identifier =0 or 1.

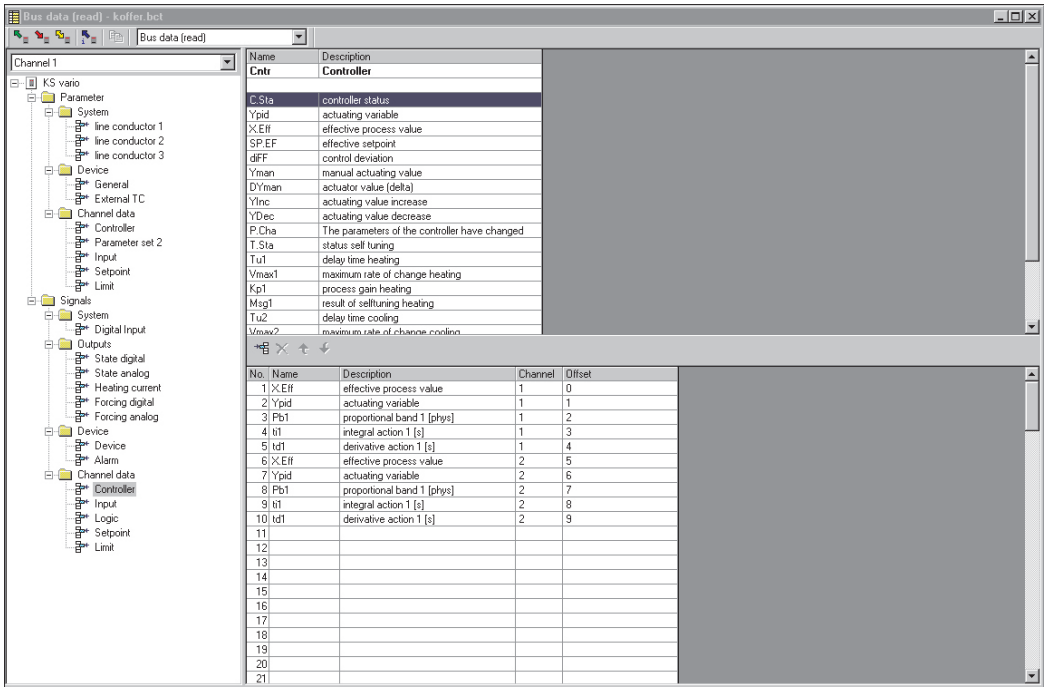
Any process data and parameters can be selected in KS vario using the BlueControl tool. This data is updated continuously in the bus coupler cache memory.

The process data cover a data range of 1080 word data in the write cache and read cache.

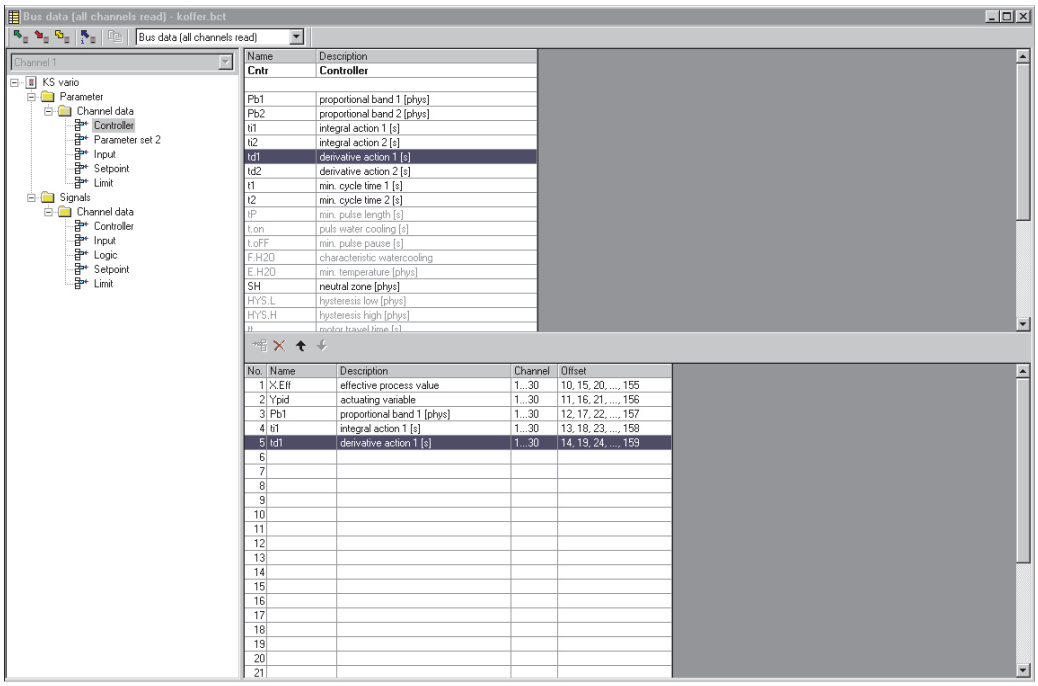
4.1.1 Definition of transmitted values in the “BlueControl” engineering tool

BlueControl provides two methods for selecting the data to be read (analogous for write direction):

- Any max. 120 parameters and process data from different channels for writing and max. 120 for reading. The position determines the order of transmission.



- Additionally or as an alternative, any max. 32 parameters and process data can be selected in common for all channels. Thus e.g. the process values from all channels (max. 30) can be transmitted by selecting a datum. In total,. max. 960 write and read data can be defined (32 data x 30 channels).



These selected data (max. 1080 write and 1080 read data) are available in the bus coupler asa cache memory in the order defined in BlueControl. The indexes or offsets of the individual data are displayed via the BlueControl tool or can be printed out.

4.1.2 Structure of and access to the data cache in the Ethernet bus coupler

The process data cover a data area of 1080 word data in the write cache and in the read cache. Access is using the following addresses:

Modbus address	Data area	Access mode
1...1080	Read cache	Read
2001...3080	Write cache	Write

Index read cache	**Content**
1 **max.120**	Any data from any channels
max. 121 **max. 1080**	Selected data (identical for all channels): All channel 1 data All channel 2 data ... All channel 30 data

Index write cache	**Content**
1 **max.120**	Any data from any channels
max. 121 **max. 1080**	Selected data (identical for all channels): All channel 1 data All channel 2 data ... All channel 30 data

4.2. Free access to any KS vario data

Access using Unit Identifier = 2..255

The address is coded in 2 bytes. The 2 most significant bits (D15, D14) are used to define the format in which the data are written or read.

The Modbus directory is divided into segments of equal size of 512 words each (bit D13...D09). Each of these segments can be used for access to all data for one control channel at a time (1...30 channels).

2 segments have a special status. In the lowest addresssegment (Modbus addr. 0..512), all device data are stored. The following segment (addr. 512...1023) contains the most important process data from all 30 channels additionally. This segment is intended for access by visualization facilities.

For a detailed address survey of all data, see document:
Parameter table for KS vario (9499-040-72918)

The signification of address bits is as given on the following page.

INTEGER/ FIX-Point Modbus addresses:

MSB		LSB
D15 - D14	**D13 - D09**	**D08 - D00**
Data format **00: Integer** **01: Fix Point 1** **1X: reserved for float**	**Device, visualization, channel X** **00000: device data** **00001: visualization data** **00010: channel 1 data** **00011: channel 2 data** **....** **11111: channel 30 data**	**Relevant datum**

Modbus directory (data format: integer):
For the Fix Point 1 area, 4000 hex must be added for the addresses.

Addresses	**Data**
0 511 (1FF hex)	**Device data**
512 (200 hex) 1023 (3FF hex)	**Visualization area Channel 1..30**
1024 (400 hex) 1535 (5FF hex)	**Channel 1 data**
1536 (600 hex) 2047 (7FF hex)	**Channel 2 data**
....	....
15872 (3E00 hex) 16383 (3FFF hex)	**Channel 30 data**

FLOAT Modbus addresses:

MSB		LSB
D15	**D14 - D10**	**D09 - D00**
Data format **0: reserved for integer and Fix Point 1** **1: Float**	**Device, visualization, channel X** **00000: device data** **00001: visualization data** **00010: channel 1 data** **00011: channel 2 data** **....** **11111: channel 30 data**	**relevant datum (offset 2)**

Modbus directory (data format: FLOAT):

Addresses	**Data**
32768 (8000 hex) 33791 (83FF hex)	**Device data**
33792 (8400 hex) 34815 (87FF hex)	**Visualization area Channel 1..30**
34816 (8800 hex) 35839 (8BFF hex)	**Channel 1 data**
35840 (8C00 hex) 36863 (8FFF hex)	**Channel 2 data**
....	....
64512 (FC00 hex) 65535 (FFFF hex)	**Channel 30 data**

These data are 4 bytes long each.

Values which can be transmitted:

Integer:	-30000 ... +32000	(resolution: +/-1)
Fix Point 1:	-3000.0 ...+3200.0	(resolution: +/- 0,1)
Float:	-1.0 E+037...+1.0 E+037	(resolution: +/- 1.4E-045)

The following special values are defined during transmission in **integer format**:

-31000	This datum is not defined. This value is returned by the controller, unless a datum within a block is defined with block reading.
-32000	The function is switched off.
-32768	Corresponds to 0x8000 hex. The value to be transmitted is out of the integer range which can be transmitted.

The following special values are defined for transmission in **float format**:

-1.5E37	This datum is not defined. This value is returned by the controller, if a datum within the block is not defined with block reading.

In the code tables (parameter tables for KS vario (9499-040-72918)), the addresses of each parameter for the relevant data format in decimal values are specified (addr. = integer without digits behind the decimal point; 1 dP = integer with 1 digit behind the decimal point; real = float (IEEE format)).

5. General information on the RTU Modbus protocol

5.1. General

The MODBUS protocol was defined for communication between a supervisory system and the Modicon control system.

ASCII and RTU protocols were defined. Instrument KS VARIO supports the RTU protocol.

The structure for transfer of a byte in the RTU protocol is:

Start bit	8 data bits	Parity/stop bit	Stop bit

Even or odd parity bit can be selected. Unless a parity bit is selected, an additional stop bit is transferred.

5.1.1 General message structure

The message is read in into a data buffer with a max. length of 250 bytes. Longer messages are not accepted. The instrument does not reply.

The message is composed of the following elements:

Instrument address	Function code	Data	CRC	End identifier

- Instrument address (Addr)
  The instrument address specifies the instrument. Instrument addresses within 1 - 247 can be defined. Instrument address 0 is used as a broadcast message. A broadcast message can be defined for write orders, which are handled by all instruments on the bus. As all instruments handle the order, no reply by the instruments is given.
- Function code
  The function code defines the type of a message. There are 17 defined messages. Which messages are supported is described in chapter "Data and function control".
- Data
  The data block comprises the further specification of the action defined with the function code. The data block length is dependent of function code. For further information, see chapter "MODBUS function format" (chapter 4). The internal data buffer includes 256 bytes. I.e., max. 120 integer or 60 real data of a message can be written or read out.
- CRC
  The CRC code ensures that transmission errors can be detected. For further information, see chapter "CRC".
- End identifier
  The end of a message is defined by a time of 3,5 characters without data transfer. For further information, see chapter "End identifier".

5.1.2 Parity (PrtY)

Even, odd or no parity can be selected.

The parity bit can be used for checking, if there was a single error within a byte during transmission.

With even parity, the parity bit is set so that the sum of set bits in the 8-data bits and the parity bit is an even number. This is applicable analogously to odd parity.

When detecting a parity error during reception, no reply message is generated.

Unless a parity is selected, 1 or 2 stop bits can be output (determination via configuration).

5.1.3 CRC

The CRC is a 16-bit value which is appended to the message. This value is used to determine, if the transmission of a message was detected correctly. In conjunction with parity checking, all possible transmission errors should be detected.

When detecting a parity error during transmission, no reply message is generated.

The algorithm for generating the CRC is:

- Load the CRC register with FFFF
- Exclusive OR function of the send/receive bytes with the high portion of the CRC register
- Shift the CRC register right by 1 bit
- If the shifted bit was 1, connect the CRC register with value A001 by an exclusive OR function.
- Repeat steps 3 and 4 for the other 7 data bits.
- Repeat steps 2 to 5 for all other send/receive bytes.
- Append the result of the CRC register to the message, starting with the high portion. When checking a receive message, the result in the CRC register is 0, if the message is handled inclusive of the CRC.

5.1.4 End identier

The end identifier of a message is specified as rest situation on the Modbus with a length of 3,5 characters. This time must elapse, before a slave may start its reply, or before a master can send another message.

The evaluation of a message may start, when the rest condition on the Modbus was met during more than 1,5 characters. However, a reply is started only after 3,5 characters.

5.1.5 Modbus function format

The signification of the data range is different dependent of function code. The Modbus protocol defines 17 different functions.

To permit reading and writing of process data, parameters and configuration data with a minimum number of accesses, the relevant ranges are grouped together, whereby process data can be defined several times in different groups.

Example for a transmission

Inquiry:

Field name	Value (hex)	Signification
Address	11	Address 17
Function	04	Read parameter/configuration
Start address high Start address low	03 EC	Start address 1004
Number of values	00 03	Number of values 03
CRC	CRC byte1 CRC byte2	

Reply:

Field name	Value	Signification
Address	11	Address 17
Function	04	Read parameter/configuration
Number of bytes	06	6 data bytes are sent
Value1	04 2A	Value1 = 1066
Value2	00 8C	Value2 = 140
Value3	10 3E	Value3 = 4158
CRC	CRC byte1 CRC byte2	

5.1.6 Modbus Functioncodes

Codetable see chapter 3.9

Details for functionscode 43 (0x2B), Read Device Identification:

Object ID	Object Name / Description	Type	M/O	category
0x00	VendorName	ASCII String	Mandatory	Basic
0x01	ProductCode	ASCII String	Mandatory	
0x02	MajorMinorRevision	ASCII String	Mandatory	
0x03	VendorUrl	ASCII String	Optional	Regular
0x04	ProductName	ASCII String	Optional	
0x05	ModelName	ASCII String	Optional	
0x06	UserApplicationName	ASCII String	Optional	
0x07 ... 0x7F	Reserved		Optional	
0x80 ... 0xFF	Private objects may be optionally defined. The range (0x080 - 0xFF) is product depandant	Device dependant	Optional	Extended

Accessses at the basic- and regular-objects are supported. Level 2 is supported as conformity level [regular identification (stream access only)].

REQUEST

Function Code	1 Byte	0x2B
MEI Type *	1 Byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Object ID	1 Byte	0x00 to 0xFF

MEI= Modbus Encapsulated Interface

RESPONSE

Function Code	1 Byte	0x2B
MEI Type	1 Byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Conformity level	1 Byte	
More Folows	1 Byte	00 / FF
Next Object Id	1 Byte	Object ID number
Number of objects	1 Byte	
List of		
Object ID	1 Byte	
Object length	1 Byte	
Object Value	Object length	Depending on the object ID

ERROR

Function Code	1 Byte	0xAB: Fc 0x2B + 0x80
MEI Type	1 Byte	14
Exception Code	1 Byte	01, 02, 03, 04

Object Contens:
Object: 0 = PMAutomation
Object: 1 = VARIO BK ETH
Object: 2 = V1.00 (aktuelle Softwareversion)
Object: 3 =
Object: 4 = vario
Object: 5 = Ethernet
Object: 6 = VC-101-00131

5.2. Examples

5.2.1 Read process data data, parameter or configuration data

The structure of a message is:

Inquiry:

Field name	Value	Signification
Address	11	Address 17
Function	03 or 04	Read process data data, parameter or configuration data
Start address high Start address low	04 98	Start address 0498 (ti1 / channel 1)
Number of values	00 02	2 data
CRC	CRC byte1 CRC byte2	

Reply:

Field name	Value	Signification
Address	11	Address 17
Function	03 or 04	Read process data data, parameter or configuration data
Number of bytes	04	4 data bytes are sent
Parameter1	00 B4	Process data data, parameter/configuration datum 0498= 180
Parameter ti2	01 4D	Process data data, parameter/configuration datum 0499= 333
CRC	CRC byte1 CRC byte2	

Broadcast is not possible.

Unless the 1st parameter / configuration datum was defined, error message "ILLEGAL DATA ADDRESS" is generated.

Unless other values after the 1st value in the area to be read out are defined, these are entered with value "NOT DEFINED VALUE". This can be used for reading out areas with gaps using a message

5.2.2 Write a single data (process data, parameter or configuration)

The structure of a message is:

Inquiry:

Field name	Value	Signification
Address	11	Address 17
Function	06	Write a single datum (process data, parameter or configuration)
Write address high Write address low	0D 57	Write address 15990 (SetpInterface of channel 30)
Value = 123	00 7B	
CRC	CRC byte1 CRC byte2	

Reply:

Field name	Value	Signification
Address	11	Address 17
Function	06	Write a single datum (process data, parameter or configuration)
Write addr. high Write addr. low	3E 76	Write address 15990 (SetpInterface of channel 30)
Value = 123	00 7B	
CRC	CRC byte1 CRC byte2	

The structure of a correct reply message is exactly as defined.
Broadcast is possible.

Entry in real data format is not possible, because only values of 2 bytes can be transmitted.

If the value is out of the adjustable range, error message "ILLEGAL DATA VALUE" is generated. The datum remains unchanged.
Unless the datum can be written (e.g. configuration datum and the instrument is on-line), error message "ILLEGAL DATA VALUE" is generated.

5.2.3 Writing several process data, parameter and configuration data

The structure of a message is:

Inquiry:

Field name	Value	Signification
Address	11	Address 17
Function	10	Write several process data data, parameter or configuration data
Start address high Start addresse low	0D 57	Write address 3415
Number of values	00 02	2 values
Number of bytes	04	4 data bytes are sent
Parameter/configuration datum 15	00 DE	Process data datum, parameter or configuration datum 3415 = 222
Parameter/configuration datum 16	01 4D	Process data datum, parameter or configuration datum 3416 = 333
CRC	CRC byte1 CRC byte2	

Reply:

Field name	Value	Signification
Address	11	Address 17
Function	10	Write several process data data, parameter or configuration data
Start address high Start address low	0D 57	Write address 3415
Number of values	00 02	2 process data data, parameter/configuration data
CRC	CRC byte1 CRC byte2	

Broadcast is possible.

Unless the 1st value was defined, an error message "ILLEGAL DATA ADDRESS" is generated.

Unless the 1st value can be written (configuration and instrument is on-line), an error message "ILLEGAL DATA VALUE" is generated.

Unless other values in the defined range after the 1st value are defined or can be written instantaneously, these values are overread. Data in these positions are not changed. The purpose is to change parts with gaps or which cannot be written instantaneously by means of a message. No error message is output.

If values are out of the adjustable limits, error message "ILLEGAL DATA VALUE" is generated. Evaluation of the following data is omitted. Data which were already stored correctly are active.

The Modbus does not provide information related to the error position in its error report. If this is required, a datum containing the position of the last error must be defined. In case of error, this datum can be read out by the master.

5.2.4 Read-out and specification of data in float format

Level-1 data, parameter and configuration data in float format can be read out and written. (Function codes 03, 04, 16)
Writing single data in float format with code 06 is not possible, since only 2 bytes for the value of the datum can be transmitted by means of this function.

If data in float format are required, the address of the required datum must be calculated as follows:
Address of the datum in integer format multiplied with factor 2
Addition of an offset of 8000H.

In "Number of values", a value twice as high as with a message for data in integer format is required.
Accordingly, the value in field "Number of data bytes" is twice as high.

All data are always converted into float values. This is also applicable to status or control words.
The data are transmitted in Motorola format (exponent followed by mantissa first).
The float format structure of a message as described in the previous chapter is:

Inquiry:

Field name	Value	Signification
Address	11	Address 17
Function	10	Write several process data data, parameter or configuration data
Start address high Start address low	9A AE	Write address 2 * 3415 + 8000H for float format
Number of values	00 04	2 values in float format
Number of bytes	08	8 data bytes are sent
Parameter/configuratio n datum 15	43 5E 00 00	Process data datum, parameter or configuration datum 3415 = 222
Parameter/configuratio n datum 16	43 A6 80 00	Process data datum, parameter or configuration datum 3416 = 333
CRC	CRC byte1 CRC byte2	

Reply:

Field name	Value	Signification
Address	11	Address 17
Function	10	Write several process data data, parameter or configuration data
Start address high Start address low	9A AE	Write address 2 * 3415 + 8000H for float format
Number of values	00 04	2 process data data, parameter or configuration data in float format
CRC	CRC byte1 CRC byte2	

5.3. Error report

The error report is generated, when interpretation or changing a datum are not possible, although the message was received correctly.

When detecting a transmission error, no reply is given. The master must resend the message.

Detected transmission errors are:

- Parity error
- Framing error (no stop bit received)
- Overrun error (receive buffer overflow, or data could not be fetched in time by the UART)
- CRC error

The data structure of the error report is:

Field name	Value	Signification
Address	11	Address 17
Function	90	Error report for message Write several parameter/configuration data
Error code	02	ILLEGAL DATA ADDRESS
CRC	CRC byte1 CRC byte2	

In field Function, the most significant bit is set.
The error code is transmitted in the following byte.

The following error codes are defined:

Code	Name	Signification
01	ILLEGAL FUNCTION	The received function code is not defined in the instrument.
02	ILLEGAL DATA ADDRESS	The received address is not defined in the instrument. When reading (function code 01, 03, 04) or writing (function code 0F, 10) several data simultaneously, this error is generated only, unless the first datum is defined.
03	ILLEGAL DATA VALUE	The received value is out of the adjusted limits or cannot be written instantaneously (instrument is not in configuration mode). When writing several data simultaneously (function code 0F, 10), this error is generated only, unless the first datum can be written.
06	SLAVE DEVICE BUSY	Will be sent back, if no communication channel is available. The buscoupler supports max. 16 communication channels
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Will be sent back, if no communication is possible with KSvario

The Modbus protocol includes further defined error codes, which, however, are presently not supported:

Code	Name	Signification
04	SLAVE DEVICE FAILURE	A non-reproducible error occurred during message processing.
05	ACKNOWLEDGE	The instrument has received an inquiry and handles it. As handling takes a very long time, this reply is output to prevent an interface timeout. The master can poll the diagnosis, to find out if handling is finished.
07	NEGATIVE ACKNOWLEDGE	The instrument cannot handle the requested order. This error message can be output for changing a configuration datum, although the instrument is not in configuration mode.
08	MEMORY PARITY ERROR	Parity error found when reading the memory.