
# LOGOSCREEN *nt*

## Paperless Recorder with TFT display and CompactFlash card

B 70.6580.2.0

Interface Description

07.07/00472204

# Contents

# Contents

## 1.1 Preface

Please read this interface description before commissioning the interfaces for the instrument. Keep the interface description in a place which is accessible to all users at all times.

Please assist us to improve this interface description, where necessary.

Your comments will be appreciated.

Phone +49 661 6003-0
Fax +49 661 6003-607

All the necessary information for operating the interface is contained in this interface description. However, if any difficulties should still arise during start-up, please do not carry out any unauthorized manipulations. You could endanger your rights under the instrument warranty!

Please contact the nearest subsidiary or the head office in such a case.

---

When returning modules, assemblies or components, the regulations of EN 100 015 “Protection of electrostatically sensitive components” must be observed. Use only the appropriate **ESD** packaging for transport.

Please note that we cannot accept any liability for damage caused by ESD.

**ESD** = electrostatic discharge

---

## 1.2 Typographical conventions

### 1.2.1 Warning signs

The symbols for **Danger** and **Caution** are used in these operating instructions under the following conditions:

**Danger** This symbol is used when there may be **danger to personnel** if the instructions are ignored or not followed correctly!

**Caution** This symbol is used when there may be **damage to equipment or data** if the instructions are ignored or not followed correctly.

**Caution** This symbol is used where special care is required when handling components liable to damage through electrostatic discharge.

### 1.2.2 Note signs

**Note** This symbol is used when your **special attention** is drawn to a remark.

abc[1] **Footnote** Footnotes are remarks that refer to specific points in the text. Footnotes consist of two parts:

A marker in the text, and the footnote text.

The markers in the text are arranged as continuous superscript numbers.

### 1.2.3 Representation modes

0x0010 **Hexadecimal number** A hexadecimal number is identified by being preceded by an “0x” (here: 16 decimal).

## 2.1 Areas of application

The paperless recorder has several interfaces:

- as standard: a TTL setup connector on the front panel, and on the back (only for a Modbus slave)
- as standard: serial interface RS232 or RS485 (Modbus slave, Modbus master, bar code)
  for communication with the bus system or a PC. This can be used to read out measurements or other device and process data from the paperless recorder.
- as standard: Ethernet 10/100 Mbit/sec
  for communication with the instrument via a network (setup, browser, Modbus-TCP, e-mail alarm)
- optionally: serial interface RS232
  for the connection of a bar code reader (Modbus slave, Modbus master, bar code)

## 2.2 System requirements

The following items are required for operating the serial interface:

- Connecting cable:
  for RS232 / RS485 e.g.
  PC interface
  with TTL/RS232 converter and adapter — Part No. 70/00350260
  PC interface
  with USB/RS232 converter and adapter — Part No. 70/00456352

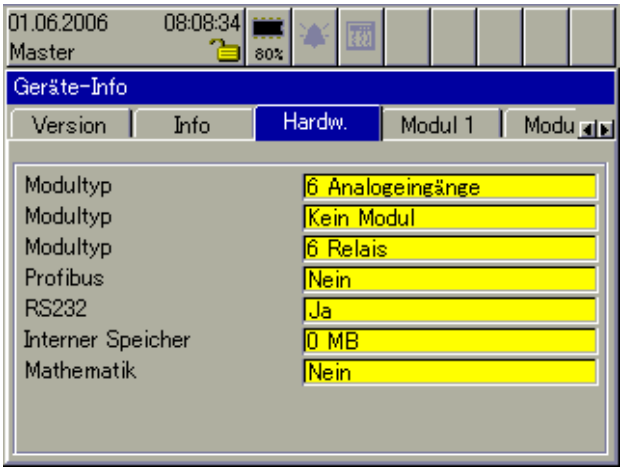
  for Ethernet interface, e.g.
  RJ 45 patch cable, CAT 5 or better (crossover)
- setup or evaluation program, e.g.
  setup program: — Part No. 70/00468991
  PC evaluation software PCA3000 — Part No. 70/00431882
  PCA communications software PCC — Part No. 70/00431879
  JUMO SVS-2000 visualization: — data sheet 70.0753
- PC or notebook

## 2.3 Identifying the interface

The interface for RS232 or RS485 and the Ethernet interface are available in every instrument.

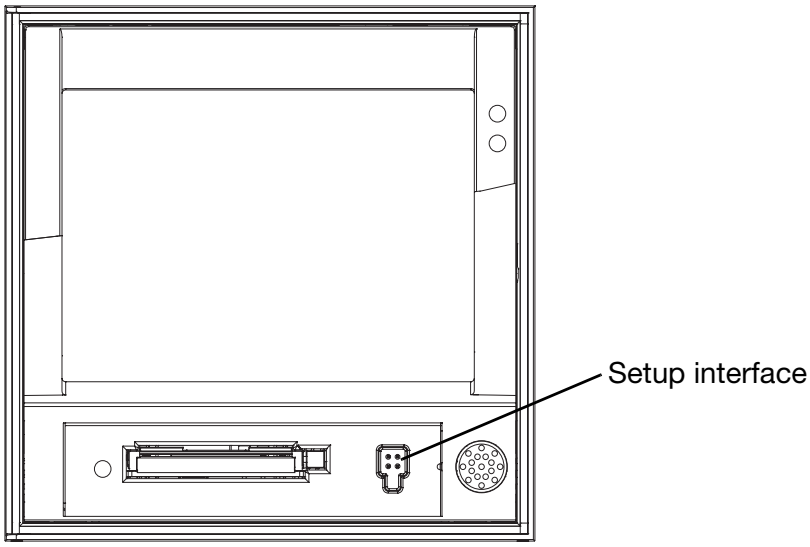
The RS232 serial interface for connecting a bar code reader and the Profibus-DP interface are options.

To see whether the instrument already has an additional option interface, use the menu *Device info* ➔ *Hardware* to check.



## 3.1 Position of the connection

**Front panel of the paperless recorder**




**Connection diagram**

Use either the setup interface on the front panel **or** the setup interface on the back!

The connection on the front panel can only be made through the setup interface with the connecting cable for “PC interface with TTL/RS232 converter and adapter”, Part No. 70/00350260.

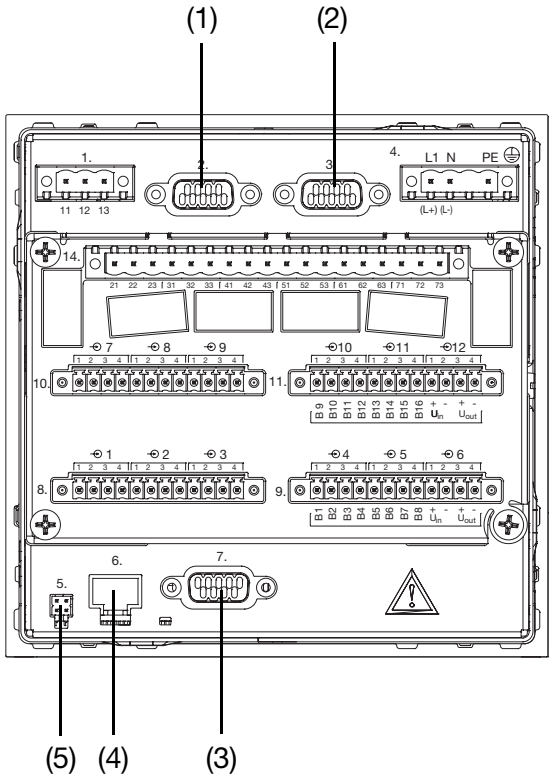
If the PC or notebook does not have a serial interface, then the connecting cable “PC interface with USB/RS232 converter and adapter”, is also required (Part No. 70/00456352).

The transmission parameters for the interface are fixed in the instrument, and cannot be altered.

- baud rate = 9600 bps
- data format = 8 data bits, 1 stop bit, no parity

# 3 Connecting the interface

**Rear view of the paperless recorder**




(1) RS232 interface for bar code reader (option)
(2) Profibus-DP interface (option)
(3) RS232/RS485 interface (standard)
(4) Ethernet interface (standard)
(5) Setup interface (standard)

**Connection diagram RS232/RS485**

| RS232 (connector (1) + (3)) | | RS485 (connector (3)) | |
|---|---|---|---|
| 1 ○ | | 1 ○ | |
| 2 ○ | RxD | 2 ○ | |
| 3 ○ | TxD | 3 ○ | TxD+/RxD+ |
| 4 ○ | | 4 ○ | |
| 5 ○ | GND | 5 ○ | GND |
| 6 ○ | | 6 ○ | |
| 7 ○ | | 7 ○ | |
| 8 ○ | | 8 ○ | TxD-/RxD- |
| 9 ○ | | 9 ○ | |

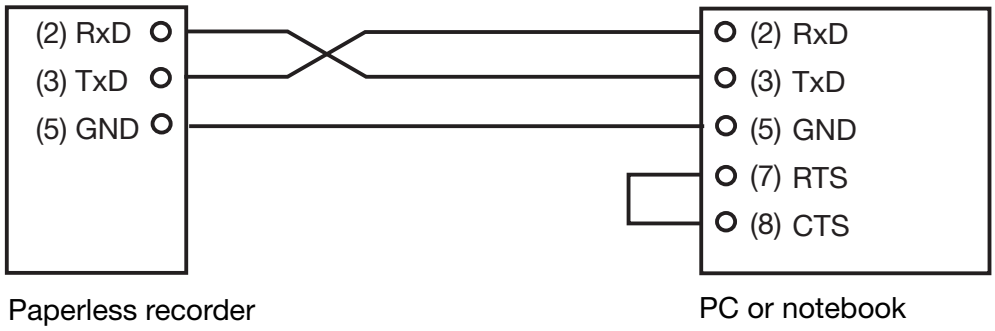
☞ We recommend using a twisted-pair connecting cable with shielding!
Only the signal lines shown above are to be connected, otherwise errors will occur.

## 3.2 RS232

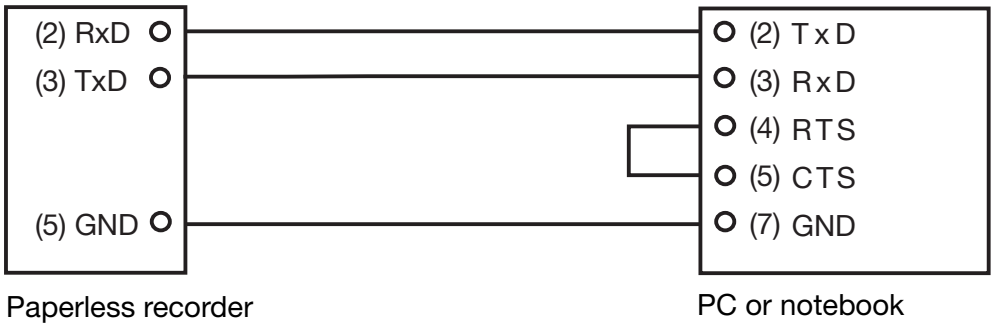
In the case of the RS232 interface, the handshake lines (RTS, CTS) are not used. The RTS line from the master (PC or notebook), which is the CTS line for the paperless recorder, will be ignored. The response is sent back immediately by the paperless recorder. The CTS line of the master (RTS on the recorder) remains open.

If the program that is used evaluates the handshake lines, then they must be bridged in the cable.

**PC COM interface with 9-pin Sub-D socket**




**PC COM interface with 25-pin Sub-D socket**




## 3.3 Switching between RS232 and RS485

The changeover between the RS232 and RS485 interfaces is made through the paperless recorder parameter
*Configuration* ➔ *Interfaces* ➔ *Interface1* ➔ *General* ➔ *Type*
or, using the setup program
*Edit* ➔ *serial interface* ➔ *RS232/RS485* ➔ *Type*

# 3 Connecting the interface

## 3.4 Configuration of the serial interfaces

**Configuration on the paperless recorder**

✱ On the recorder, select *Configuration* ➔ *Interface* ➔ *Interface 1/2* ➔ *General*.
The parameters for the configuration of the interface will now be available.

**Configuration through setup program**

Configuration with the aid of the setup software is made through the menu item *Edit* ➔ *Interface*.

| | Parameter | Value/Selection | Description |
|---|---|---|---|
| **Interface type** | ➔Type | **RS232**,<br>RS485 | Can only be edited for interface 1.<br>See Chapter 3.3 Switching between RS232 and RS485, page 11 |
| **Protocol** | ➔Protocol | **Modbus slave**,<br>Modbus master,<br>bar code | see Chapter 5 Serial protocol types, page 33. |
| **Baud rate** | ➔ Baud rate | 9**600 bps**,<br>19200 bps,<br>38400 bps | see Chapter 4.3 Timing sequence for communication, page 16. |
| **Transmission mode (RTU)** | ➔ Data format | **8-1-none**,<br>8-1-odd,<br>8-1-even | see Chapter 4.2 Transmission mode (RTU), page 15. |
| **Device address** | ➔ Device address | **1** — 254 | see Chapter 4.5 Device address, page 18. |
| **Min. response time** | ➔ Min. response time | **0** — 500 msec | see Chapter 4.3 Timing sequence for communication, page 16. |

☞ For communication via the **back panel** RS232 interface and the setup interface, the device address must be observed, even though it is not a bus interface!

The configuration of the setup interface cannot be edited, but is fixed to
- Modbus slave,
- 9600 bps,
- 8-1-none,
- device address 1

The three serial interfaces can be operated simultaneously. There are merely some interlocks built in at the protocol level (e.g. only one setup can be written at a time).

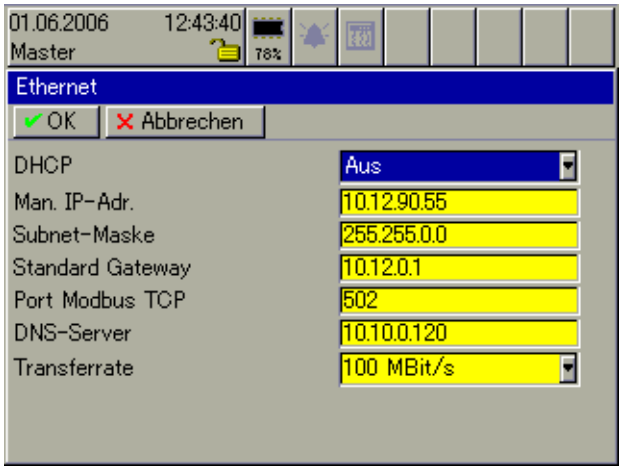
## 3.5 Configuration of the Ethernet interface

**Configuration on the paperless recorder**

is carried out via the menu item *Configuration* ➔ *Interface* ➔ *Ethernet*.

**Configuration via setup program**

is carried out via the menu item *Extras* ➔ *Ethernet interface.*




### 3.5.1 Parameters

**DHCP**

DHCP (Dynamic Host Configuration Protocol) is used so that the paperless recorder can automatically receive an IP address and other communication parameters from a DHCP server.

| | |
|---|---|
| On | DHCP is switched on, the recorder receives its IP address from the DHCP server |

The other communication parameters that the paperless recorder usually receives from the DHCP server include the subnet mask, gateway address, and the lease time.

When the lease time (user time) has expired, the IP address loses its validity. In order that the paperless recorder may always have a valid IP address, it sends a query to its DHCP server whether the address is still valid when 50 of the lease time has expired. If the DHCP server is not available, the recorder repeats the query until 87.5% of the lease time has expired. After that, the recorder sends the query not only to the DHCP server, but to the entire network. If the lease time expires without the IP address being confirmed, the recorder declares the address as invalid and is no longer accessible in the network.

An assigned address can be altered by the DHCP server. For example, if the PCA communications software is used for the automatic collection of data, then the address must be altered within the software.

The automatically assigned IP address can be read in the menu *Device manager* ➔ *Device info* ➔ *Eth. Info*.

**Manual IP address**

If automatic IP address assignment is not used (DHCP = OFF), then the IP address for the paperless recorder is set up here.

**Subnet mask**

If automatic IP address assignment is not used (DHCP = OFF), then the subnet mask is set up here.

The subnet mask is used to gather devices (PC, recorders etc.) together to form subnets. All devices that have an IP address that is AND linked to the subnet mask belong to one subnet and can communicate with one another.

If devices outside the subnet need to be accessed, then the communication must be handled by a gateway (standard gateway).

**Standard gateway**

If automatic IP address assignment is not used (DHCP = OFF), then the IP address for the standard gateway is set up here.

The standard gateway is used for communication with devices that do not belong to the subnet.

**Port Modbus TCP**

The port address must be set if the paperless recorder needs to be accessed by visualization software, and the Modbus TCP protocol (Modbus tunneling: outer frame = Ethernet, inner frame = Modbus) is to be used.

Any change to this parameter will only becomes effective after the recorder has been restarted!

**DNS server**

This is used to set up the IP address for a DNS server that is installed in the network. The DNS server is required to resolve the name for e-mail transfer via Ethernet.

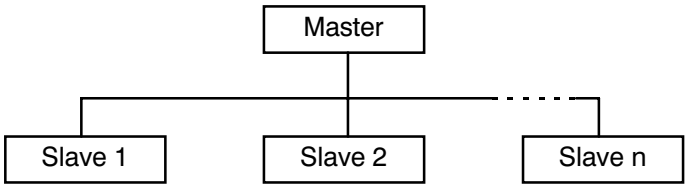
Any change to this parameter will only become effective after the recorder has been restarted!

**Transfer rate**

Here you can configure the transfer rate that is used for communication between the paperless recorder and the DHCP server or other computers.

## 4.1 Master-slave principle

When using Modbus/Jbus, the communication between a master (PC or notebook) and a slave device (paperless recorder), takes place according to the master-slave principle, in the form of a data request/instruction - response.




The master controls the data exchange, the slaves only have a response function. They are identified by their device addresses.

The paperless recorder can be operated either as a Modbus slave, see Chapter 5.1 Modbus slave, page 33, or as a Modbus master, see Chapter 5.2 Modbus master, page 35.

In a Modbus network, only **one** device is allowed to take on the master function!

## 4.2 Transmission mode (RTU)

The transmission mode used is the RTU mode (Remote Terminal Unit). Data are transmitted in binary format (hexadecimal) with 8 bits, as 16-bit integer values, or as 32-bit float values.

**Data format**

The data format describes the structure of a transmitted byte.

| Data word | Parity bit | Stop bit | Number of bits |
|---|---|---|---|
| 8 bits | none (no) | 1 | 9 |
| 8 bits | even | 1 | 10 |
| 8 bits | odd | 1 | 10 |

The data format that is to be used can be selected, see Chapter 3.4 Configuration of the serial interfaces, page 12.

# 4 Modbus protocol description

## 4.3 Timing sequence for communication

**Character transmission time**

The start and end of a data block are marked by transmission pauses. The character transmission time (the time taken to transmit one character) depends on the baud rate and the data format that is used.

For a data format with 8 data bits, no parity bit and one stop bit, this is:

**character transmission time [msec] = 1000 * 9 bits / (baud rate)**

For the other data formats it is:

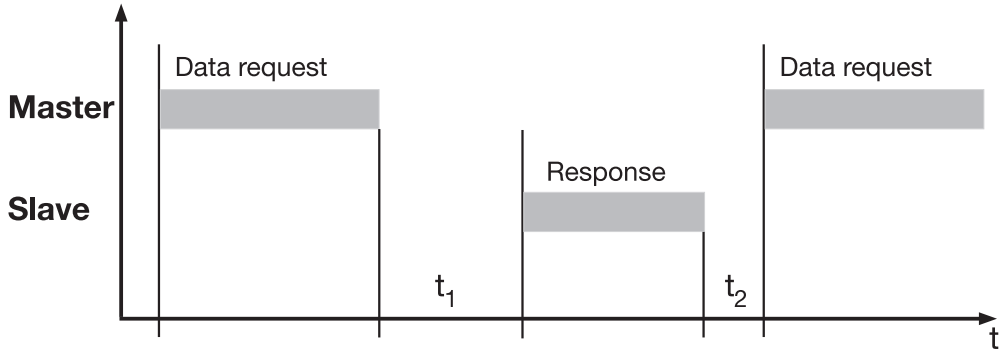
**character transmission time [msec] = 1000 * 10 bits / (baud rate)**

**Example**

| **Baud rate** [bps] | **Data format** [bit] | **Character transmission time** [msec] |
|---|---|---|
| 38400 | 10 | 0.260 |
| | 9 | 0.234 |
| 19200 | 10 | 0.521 |
| | 9 | 0.469 |
| 9600 | 10 | 1.042 |
| | 9 | 0.938 |

**Timing sequence**

A data request runs according to the following timing sequence:




$t_1$ Internal waiting time of the paperless recorder before checking the data request, and the internal processing time.

min.: 5 msec
typical: 5 — 45 msec
max.: 60 msec, or the preset “minimum response time”

A minimum response time can be set in the controller, in the menu item *Configuration* ➔ *Interface*. This setting is the minimum waiting time that must elapse before an answer is transmitted (0 — 500 msec). If a smaller value is set, then the response time may be longer than the preset value (because the internal processing time is longer), the controller will then answer as soon as the internal processing is completed. A preset time of 0 msec means that the controller answers with the maximum possible speed.

The minimum response time which can be set is required by the RS485 interface in the master, in order to switch over the interface driver from transmit to receive. This parameter is not required for the RS232 interface.

$t_2$ This is the waiting time which the master has to observe before initiating a new data request.

| | |
|---|---|
| for RS232 | at least 3.5 x the transmission time for one character (this time depends on the baud rate) |
| for RS485 | 60 msec |

While $t_1$ and $t_2$ are running, the master must not present any further data requests, since the paperless recorder will either ignore them or declare them to be invalid.

## 4.4 Structure of a Modbus telegram

**Data structure**

All telegrams have the same structure:

| Slave address | Function code | Data field | Checksum CRC16 |
|---|---|---|---|
| 1 byte | 1 byte | x byte(s) | 2 bytes |

Each telegram contains four fields:

**Slave address** the device address of a specific paperless recorder

**Function code** function selection (read/write a word)

**Data field** contains the information (depending on the function code):
- word address / bit address
- word number / bit number
- word value /bit values

**Checksum** detection of transmission errors

# 4 Modbus protocol description

## 4.5 Device address

The device address for the paperless recorder can be set between 1 and 254 (decimal), see Chapter 3.4 Configuration of the serial interfaces, page 12.

A maximum of 31 paperless recorders can be addressed via the RS485 interface.

Device address 0 is reserved as the Modbus broadcast address. An instruction from the master for address 0 will be carried out by all the slaves, but none of them will answer (because this would lead to a data collision).

If only **one** recorder is connected to the PC or notebook, then it can also be accessed through device address 255 (even if a different address has been configured). The paperless recorder will always respond to instructions for device address 255.

In the transmission protocol, the address is given in binary format (hexadecimal).

## 4.6 Function codes

**Function summary**

The functions described below can be used to read out measurements and other device and process data from the paperless recorder.

| Function number | Function | Restriction |
|---|---|---|
| 0x01 or 0x02 | read n bits | max. 256 bits (16 bytes) |
| 0x03 or 0x04 | read n words | max. 127 words (254 bytes) |
| 0x05 | write one bit | max. 1 bit |
| 0x06 | write one word | max. 1 word (2 bytes) |
| 0x10 | write n words | max. 127 words (254 bytes) |

If the recorder does not respond to one of these functions, or reacts by generating an error code, then please refer to Chapter 4.9 Error messages, page 25.

### 4.6.1 Read n bits

This function reads n bits, starting from a defined address.

**Data request**

| Slave address | Function 0x01 or 0x02 | Address first bit | Number of bits | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Response**

| Slave address | Function 0x01 or 0x02 | Number of bits read | bit value(s) | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | x byte(s) | 2 bytes |

The response is always made in complete 8-bit bytes.
The byte values that are not required are filled with the value 0.

**Example**

Read one bit, starting at address 0x0340 (word address 0x0034).

For addresses, see Chapter 7.1 Modbus addresses for important device and process data, page 43.

Data request:

| | | | | |
|---|---|---|---|---|
| 01 | 02 | 0340 | 0001 | B85A |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 02 | 01 | 01 | 6048 |
| | | | bit value | |

### 4.6.2 Read n words

This function reads n words, starting from a defined address.

**Data request**

| Slave address | Function 0x03 or 0x04 | Address of first word | Number of words | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 byte | 2 bytes | 2 bytes |

**Response**

| Slave address | Function 0x03 or 0x04 | Number of bytes read | Word value(s) | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | x byte(s) | 2 bytes |

**Example**

Read the first 3 analog inputs (i.e. the first 6 words starting at Modbus address 0x1257).

For the addresses of the analog inputs, see Chapter 7.1 Modbus addresses for important device and process data, page 43.

Data request:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 1257 | 0006 | 7160 |

Response:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 01 | 03 | 0C | 1999 | 4348 | 4CCC | 4348 | 2666 | 4396 | 8548 |
| | | | Measurement 1 200.1 | | Measurement 2 200.3 | | Measurement 3 300.3 | | |

### 4.6.3 Write one bit

For the “Write one bit” function, the data blocks for instruction and response are identical.

**Instruction**

| Slave address | Function 0x05 | Bit address | Bit value xx00 | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes<br>xx = 00 ➔ bit is set to 0<br>xx = FF ➔ bit is set to 1 | 2 bytes |

**Response**

| Slave address | Function 0x05 | Bit address | Bit value | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Example**

Write one bit, at bit address 0x0340 (i.e. bit 0 of the word address 0x0034).

For addresses, see Chapter 7.1 Modbus addresses for important device and process data, page 43.

Instruction:

| | | | | |
|---|---|---|---|---|
| 01 | 05 | 0340 | FF00 | 8DAA |

Response (as instruction):

| | | | | |
|---|---|---|---|---|
| 01 | 05 | 0340 | FF00 | 8DAA |

### 4.6.4 Write one word

For the “Write one word” function, the data blocks for instruction and response are identical.

**Instruction**

| Slave address | Function 0x06 | Word address | Word value | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Response**

| Slave address | Function 0x06 | Word address | Word value | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Example**

Set the variable “External binary In 1” (Modbus address 0x1638) to 1.

For addresses, see Chapter 7.1 Modbus addresses for important device and process data, page 43.

Instruction:

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 1638 | 0001 | CD8F |

Response (as instruction):

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 1638 | 0001 | CD8F |

### 4.6.5 Write n words

**Instruction**

| Slave address | Funct-ion 0x10 | Address of first word | Number of words | Number of bytes | Word value(s) | Check-sum CRC16 |
|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | x byte(s) | 2 bytes |

**Response**

| Slave address | Function 0x10 | Address of first word | Number of words | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Example**

Write the word “Test” (ASCII coding: 0x54 0x65 0x73 0x74 0x00) to the address 0x148 ff, so that this text is entered in the event list for group 1:

Instruction:

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 10 | 148A | 0003 | 06 | 54 65 73 74 00 00 | 9BFA |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 10 | 148A | 0003 | A412 |

# 4 Modbus protocol description

## 4.7 Transmission format (integer, float, double and text values)

**Integer values**

When using Modbus, integer values are transmitted in the following format: first the HIGH byte, then the LOW byte.

**Example**

Request the integer value from address 0x1017, when this address contains the value “4” (word value: 0x0004).

Request: 01031017000130CE (CRC16 = CE30)
Response: 010302**0004**B987 (CRC16 = 87B9)

**Float values**

When using Modbus, float values are processed in the IEEE 754 standard format (32-bit), but with the difference that bytes 1 and 2 are swapped with bytes 3 and 4.

**Single float format (32-bit) according to standard IEEE 754**

| SEEEEEEE | EMMMMMMM | MMMMMMMM | MMMMMMMM |
|---|---|---|---|
| Byte 1 | Byte 2 | Byte 3 | Byte 4 |

S = sign bit
E = exponent (complement to base 2)
M = 23 bit normalized mantissa

**Mdobus float format**

| Modbus address x | | Modbus address x+1 | |
|---|---|---|---|
| MMMMMMMM | MMMMMMMM | SEEEEEEE | EMMMMMMM |
| Byte 3 | Byte 4 | Byte 1 | Byte 2 |

**Example**

Request the float value from address 0x0035, when this address contains the value “550.0” (0x44098000 in IEEE 754 format).

Request: 140300350002D6C0 (CRC16 = C0D6)
Response: 140304**80004409**6434 (CRC16 = 3464)

After the transmission from the device, the bytes of the float value must be swapped accordingly.

Many compilers (e. g. Microsoft Visual C++) store the float values in the following order:

**Float value**

| Address x | Address x+1 | Address x+2 | Address x+3 |
|---|---|---|---|
| MMMMMMMM | MMMMMMMM | EMMMMMMM | SEEEEEEE |
| Byte 4 | Byte 3 | Byte 2 | Byte 1 |

Please check how float values are stored in your application. If necessary, the bytes will have to be swapped accordingly in your interface program, after they have been fetched from the paperless recorder.

**Double values**

When using Modbus, double values are also processed in the IEEE-754 standard format (32 bits). Unlike float values, no bytes are swapped for double values.

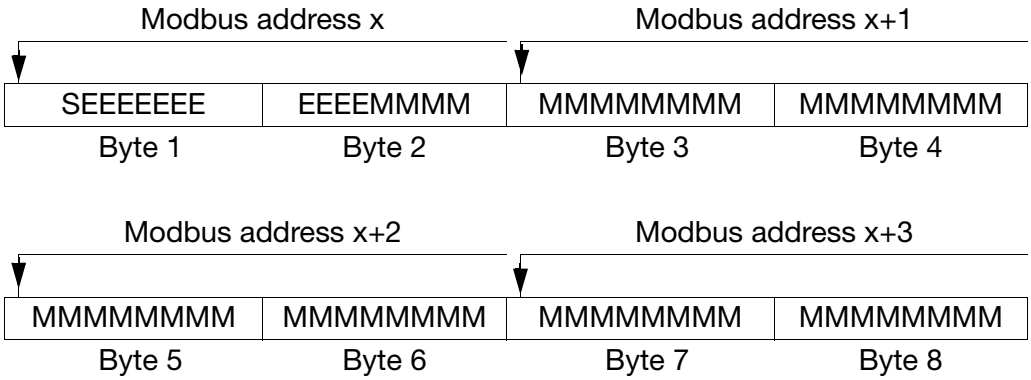
**Double float format (32-bit) according to standard IEEE 754**

| SEEEEEEE | EEEEMMMM | MMMMMMMM | MMMMMMMM |
|---|---|---|---|
| Byte 1 | Byte 2 | Byte 3 | Byte 4 |

| MMMMMMMM | MMMMMMMM | MMMMMMMM | MMMMMMMM |
|---|---|---|---|
| Byte 5 | Byte 6 | Byte 7 | Byte 8 |

S = sign bit
E = exponent (complement to base 2)
M = 52 bit normalized mantissa

**Modbus double format**




**Example**

Request the double value from address 0x0066, when this address contains the value “1234567.89” (0x4132D687E3D70A3D in IEEE 754 format).

Request: 140300660004A6D3 (CRC16 = D3A6)
Response: 140308**4132D687E3D70A3D**E1C1 (CRC16 = C1E1)

Please check how double values are stored in your application. If necessary, the bytes will have to be swapped accordingly in your program, after they have been fetched from the paperless recorder.

# 4 Modbus protocol description

**Character strings (texts)**

Character strings are transmitted in ASCII format.

An “0” (ASCII code 0x00) must always be transmitted as the last character, to mark the end of the string. Any following characters are meaningless.

Since text transmission is made as words (16-bit), if there is an odd number of characters (incl.“0”), an additional 0x00 will be added on.

The maximum lengths given in the address tables (See “Address tables” on page 43. ff) for character strings include the terminating “0”. I.e. the text for “char 11” can have a maximum length of 10 readable characters.

**Example**

Request the text from address 0x1000, if this address contains the character string “**LS NT**”
(ASCII code: **0x4C**, **0x53**, **0x20**, **0x4E**, **0x54**, 0x00).

Request: 01031000000440C9
Response: 010308**4C53204E54**0000AA0D96

Instead of “AA” in front of the CRC sum, there could be any value – since it comes after the “0”, it will be ignored.

## 4.8 Checksum (CRC16)

**Calculation method**

The checksum (CRC16) is used to recognize transmission errors. If an error is detected during evaluation, the corresponding device will not respond.

| | | | |
|---|---|---|---|
| CRC = 0xFFFF | | | |
| | CRC = CRC XOR ByteOfMessage | | |
| | For (1 to 8) | | |
| | | CRC = SHR(CRC) | |
| | | if (flag shifted right = 1) | |
| | | then<br>CRC = CRC XOR 0xA001 | else |
| while (not all ByteOfMessage processed); | | | |

The low byte of the checksum is transmitted first.

**Example 1**

Request status of relay output 1.

Instruction: read one word from address 0x1631

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 1631 | 0001 | D18D |

Response (CRC = 0x8479):

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0001 | 7984 |
| | | | Word 1 | |

Word 1 = 1 indicates that relay 1 is active.

## 4.9 Error messages

### 4.9.1 Modbus error codes

**No response from the recorder**

The slave will not respond if one of the following errors occurs:

- the baud rate or data format for the master (PC or notebook) does not match the slave (paperless recorder)
- the device address for the recorder does not match the address contained in the protocol
- the checksum (CRC16) is not correct
- the instruction from the master is incomplete or over-defined
- the number of words to be read is zero

In these cases, when the time-out of 2 seconds has expired, the data will have to be retransmitted.

**Error codes**

If the data request from the master is received by the paperless recorder without any transmission error, but cannot be processed, then the recorder will answer with an error code.

The following error codes may appear:

- 01 invalid error code
- 02 invalid address or too many words or bits to be read or written
- 03 value is outside the permitted range
- 08 write-protected value

**Response in the event of an error**

| Slave address | Function XX OR 80h | Error code | Checksum CRC16 |
|---|---|---|---|
| 1 byte | 1 byte | 1 byte | 2 bytes |

The function code is OR-ed with 0x80, which means that the MSB (most significant bit) is set to 1.

**Example**

Data request:

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 1257 | 0001 | FCA2 |

Response:

| | | | |
|---|---|---|---|
| 01 | 86 | 08 | 43A6 |

Response is error code 08, because address 0x1257 is write-protected.

### 4.9.2 Error messages for invalid values

For measurements in float format, the error number is shown in the value itself, i.e. the error number is entered instead of the measurement.

| Error code for float values | Errors |
|---|---|
| $1.0 \times 10^{37}$ | underrange |
| $2.0 \times 10^{37}$ | overrange |
| $3.0 \times 10^{37}$ | no valid input value |
| $4.0 \times 10^{37}$ | division by zero |
| $5.0 \times 10^{37}$ | math error |
| $6.0 \times 10^{37}$ | invalid terminal temperature for TE |
| $7.0 \times 10^{37}$ | invalid float value |
| $8.0 \times 10^{37}$ | integrator or statistics destroyed |

**Example**

Data request:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 1259 | 0002 | 1160 |

Response:

| | | | | | |
|---|---|---|---|---|---|
| 01 | 03 | 04 | 8E52 | 7DB4 | 51ED |

The measurement 0x7DB48E52 (= 3.0 x $10^{37}$) delivered by analog input 2 (Modbus address 0x1259) shows that the input value is invalid.

### 4.9.3 JUMO error codes as integer return values

For some lengthy sequences, such as the transfer of a program to a JUMO controller or e-mail delivery, an error code is entered in an event field or the event list at the end. The error codes are the same for all JUMO instruments.

**Error codes**

| Error code | Description |
|---|---|
| **Error list: Program memory management** | |
| 1 | program cannot be created |
| 2 | program not available |
| 3 | program cannot be deleted |
| 4 | segment cannot be deleted |
| 5 | checksum cannot be saved |
| 6 | checksum cannot be read |
| 7 | program cannot be copied |
| 8 | segment cannot be copied |
| 9 | program checksum error |
| 10 | program pointer tab. checksum error |
| 11 | program memory end |
| 12 | segment not available |
| 13 | repeat jump mark cannot be corrected |

| Error code | Description |
|---|---|
| **Error list: General input and output** | |
| 14 | please confirm with the ENTER key |
| 15 | invalid number of places |
| 16 | the entry contains invalid characters |
| 17 | value is outside the limits |
| 18 | segment not programmed correctly |
| 19 | password error |
| **Error list: Profibus job processing** | |
| 20 | busy flag from master not reset |
| 21 | impermissible job |
| 22 | error on data acceptance |
| 23 | no cyclical data available |
| 24 | structure length is not permitted |
| 25 | header ID is not permitted |
| **Error list: Keypad and program lock** | |
| 26 | keypad is locked |
| 27 | programming is inhibited |
| 28 | error in ser. EEprom (calib) |
| 29 | hardware error: HAND + AUTO locked |
| 30 | editing not permitted when program is active |
| 31 | copying not permitted when program is active |
| 32 | HAND is not permitted during AUTO delay time |
| 33 | segment change! Screen must be generated |
| 34 | no DB number for screen generation from PLC |
| 35 | no DB number for process values from PLC |
| 36 | printer busy or not ready |
| 37 | setpoint 1 was not programmed |
| 38 | set up the printer (config. / interface) |
| 39 | only possible when device is in HAND mode |
| 40 | self-optimization already running |
| 41 | time axis expired or not programmed |

| Error code | Description |
|---|---|
| 42 | time axis cannot be copied |
| 43 | time axis not available |
| 44 | program alteration is inhibited |
| 45 | HAND mode is locked |
| 46 | program start is locked |
| **Error list: Interface processing** | |
| 47 | wrong response length |
| 48 | time-out error (no response) |
| 49 | error reported in telegram protocol |
| 50 | checksum error |
| 51 | parity error |
| 52 | frame error |
| 53 | interface buffer full |
| 54 | address error (for instance: address not available) |
| 55 | wrong or unexpected command |
| **Error list: Event processing** | |
| 60 | event could not be created |
| 61 | event setting failed |
| 62 | event clear failed |
| 63 | event wait failed |
| 64 | event close failed |
| 65 | event open failed |
| 66 | sync error between group and data manager |
| **Error list: Message processing** | |
| 70 | no queue memory available |
| 71 | message queue cannot be opened |
| 72 | message pool cannot be created |
| 73 | memory from message pool cannot be requested |
| 74 | message cannot be sent |

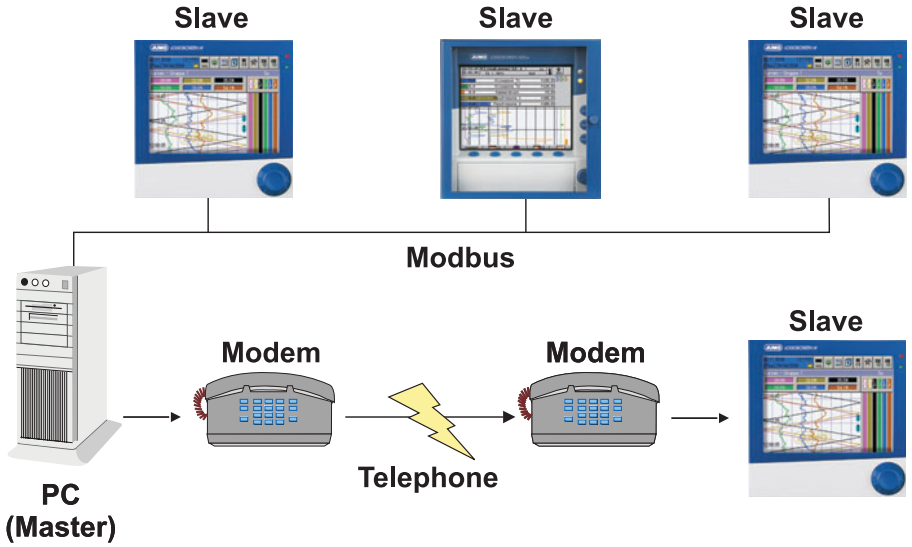
| Error code | Description |
|---|---|
| **Error list: Handling of MQX functions** | |
| 80 | task creation failed |
| 81 | hardware timer not created |
| **Error list: Flash processing** | |
| 90 | write error for data flash |
| **Error list: Other errors** | |
| 100 | undefined error |
| 101 | division by zero |
| 102 | cannot find RAM |
| 103 | RTC run-time overrun |
| 104 | ID does not exist |
| 105 | index too large (overflow) |
| 106 | data not valid |
| 107 | invalid pointer |
| 109 | string without zero characters |
| 110 | time-out (overrun) during initialization |
| 111 | value write forbidden |
| 112 | log entry with error bits that initiate debug mode |
| **Error list: E-mail dispatch via modem and Ethernet** | |
| 120 | step error in status automatic |
| 121 | invalid response length |
| 122 | no CONNECT from modem |
| 123 | FCS checksum error |
| 124 | unexpected value or response |
| 125 | CONF request not accepted |
| 126 | no CONF request from other end |
| 127 | no CHAP request from other end |
| 128 | response time-out |
| 129 | unknown modem response |
| 130 | unexpected OK from modem |
| 131 | unexpected CONNECT from modem |

| Error code | Description |
|---|---|
| 132 | unknown frame received |
| 133 | unexpected PROTOCOL from modem |
| 134 | unexpected COMPRESS from modem |
| 135 | invalid PPP packet received |
| 136 | unexpected BUSY from modem |
| 137 | unknown authentication protocol |
| 138 | ignored LCP option |
| 139 | unexpected DELAYED from modem |
| 140 | unexpected NODIALTONE |
| 141 | unknown PPP protocol |
| 142 | unknown PAP code |
| 143 | ignored IPCP option |
| 144 | ignored IPCP code |
| 145 | unknown CHAP code |
| 146 | IP checksum error |
| 147 | unknown IP protocol |
| 148 | unknown ICMP type |
| 149 | unknown LCP type |
| 150 | received as client DNS request |
| 151 | unknown DNS error |
| 152 | DNS response is divided |
| 153 | no IP received by DNS |
| 154 | unknown Udp port |
| 155 | TCP checksum error |
| 156 | TCP port error |
| 157 | unknown TCP-SYN option |
| 158 | unused TCP port |
| 159 | unknown POP3 response |
| 160 | unknown SMTP response |
| 161 | unknown DNS name |
| 162 | no MD5 requested for CHAP |

| Error code | Description |
|---|---|
| 163 | authentication error |
| 164 | cancel from other end |
| 165 | error on creating TCP socket |
| 166 | error on binding TCP socket |
| 167 | error on TCP CONNECT |
| 168 | error in TCP telegram transmission |
| 169 | error on closing TCP socket |
| 170 | error on TCP listing |
| 171 | reset on TCP ACCEPT |
| 172 | error on TCP ACCEPT |
| 173 | SMTP server reports syntax error |
| **Error list: File system processing** | |
| 200 | error during installation of partition manager |
| 201 | error during installation of MFS file system |
| 202 | error during de-installation of partition manager |
| 203 | error during de-installation of MFS file system |

## 5.1 Modbus slave

If the JUMO Logoscreen nt has been configured as a slave (see Chapter 3.4 Configuration of the serial interfaces, page 12) it will respond to a Modbus request from the master in the network.
The master controls the data exchange, the slaves only have a response function. They are identified by their device addresses.
The master is usually a PC with a setup program or a visualization program. The master can requests all the device variables of the JUMO Logoscreen nt (see Chapter 7 Address tables, page 43).



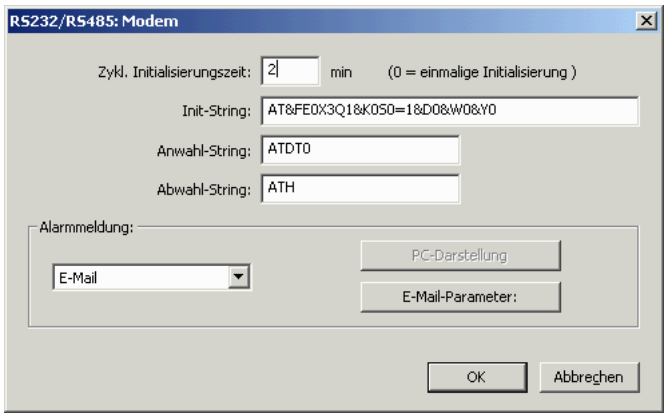

In a Modbus network, only **one** device is allowed to take on the master function!

**Modem operation**

- A Logoscreen nt in slave mode can be controlled by the master via a modem over a telephone line, see the picture above.
- The paperless recorder can initialize a modem itself (also cyclically, for the situation where the modem is switched on after the instrument).
- The recorder can use the INIT string (entry with the setup program in the mask “RS232/RS485: Modem”) to configure the modem so that it will automatically accept (“pick up”) an incoming call. The recorder can then be remotely interrogated by the master, using Modbus commands, or send a signal (e.g. an alarm) or an e-mail after active dial-up.
- Using a dial-up/hang-up string, the paperless recorder (slave) can send an alarm to a PC (master) with the JUMO SVS-2000 software. The JUMO SVS-2000 software detects incoming modem calls.
- Using a dial-up/hang-up string, the paperless recorder (slave) can also call an Internet provider and send an e-mail.

When modem operation is active, the following parameters can be edited (only through the setup program):

## Modem parameters



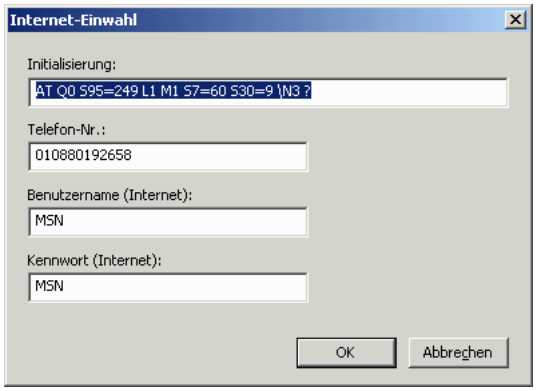

The following INIT string is required for operation as a Modbus slave via modem:

AT&FE0X3Q1&K0S0=1&D0&W0&Y0

AT&F = load present manufacturer profile
E0 = switch off character echo
X3 = switch off fixed dial tone selection, switch on busy tone selection
Q1 = switch off command responses
&K0 = switch off data flow control
S0=1 = automatic pick-up after first ring
&D0 = ignore DTR signal
&W0 = save present configuration as profile 0
&Y0 = use user profile 0 after switch-on

The dial-up/hang-up strings are only required if one of the modem alarms is activated.

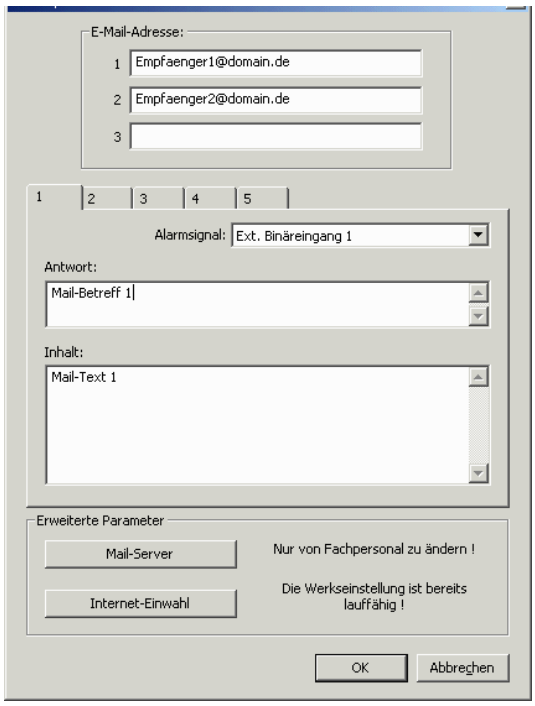
## Internet dial-up



For active dial-up to the Internet for alarms by e-mail, the device modem must be switched into a different mode, using another INIT string.

The telephone number, user name and password must be entered so that they match the details for the selected Internet provider.
When the dial-up procedure to the Internet has been completed, the modem will automatically be reset to the initial state, using the INIT string that was entered in “Modem parameters”.

**E-mail parameters**

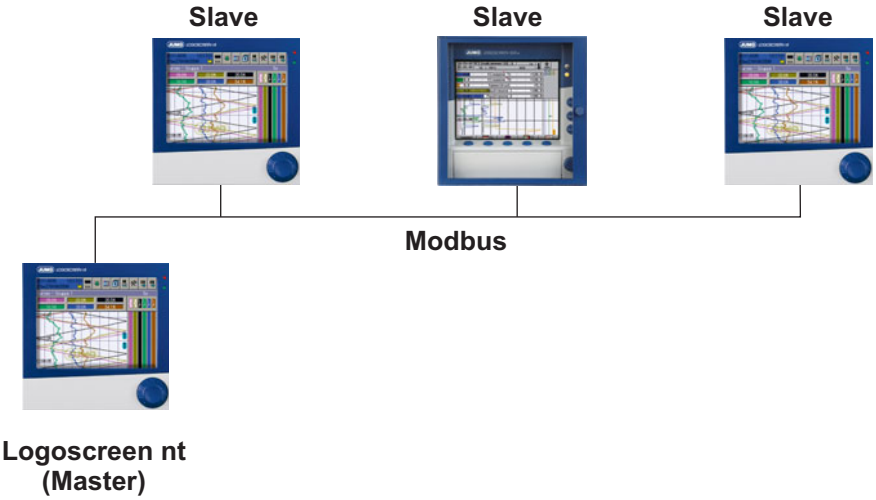



## 5.2 Modbus master

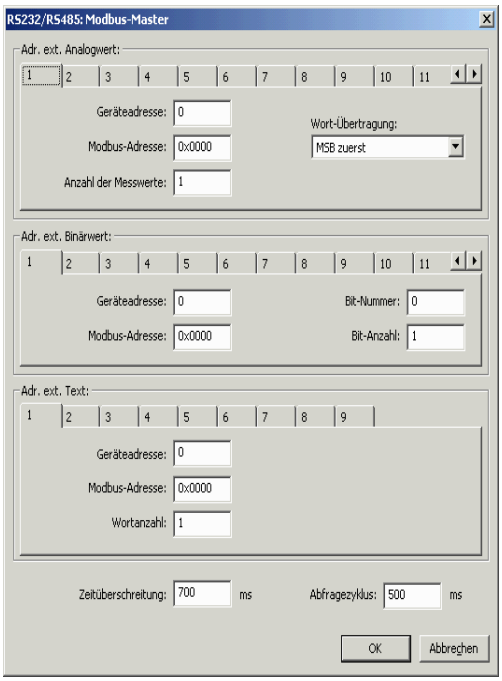
If the JUMO Logoscreen nt has been configured as a master, (see Chapter 3.4 Configuration of the serial interfaces, page 12) it can send Modbus requests to the slaves in the network (e.g. other recorders). The values that are requested will be written to the external device variables of the Logoscreen nt:

- external analog values 1 to 24 in the analog selector
- external binary values 1 to 24 in the binary selector

- external texts 1 to 9 (e.g. in order to combine batch texts in the instrument)




## Modbus master




These parameters can be edited in the setup program and on the Logoscreen nt.

For each target variable, you can enter the device address and the Modbus address from which the value will be requested.

Every programmed request can be switched to inactive by entering the device address 0 (e.g. if “external analog value 5” is no longer to be written by the Modbus master, but from the Profibus).

Double writing of a target variable will lead to undefined states, and must be avoided!

For analog and binary values, the entry of "Number of measurements" or "Number of bits" greater than 1 allows one command to read several variables, one after another. On saving, the following target variables will automatically be utilized.

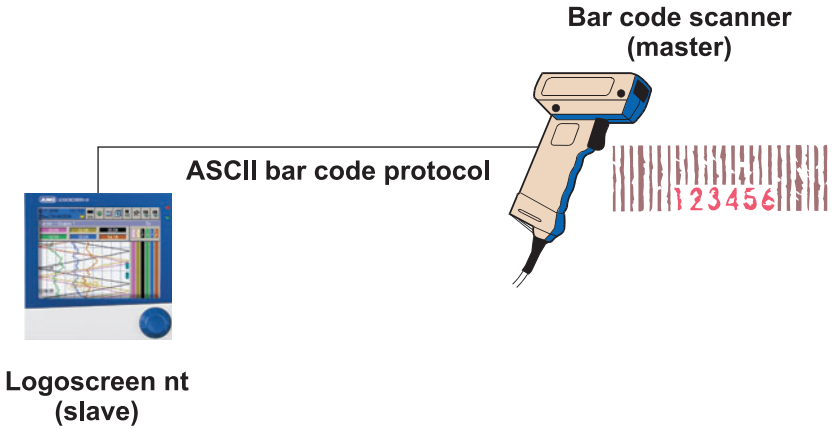
**Time-out** defines the maximum waiting time for a response after the transmission of each command before the next command is implemented.

**Polling cycle** defines the time interval between a read-in of the variables and the next read.

## 5.3 Bar code

In "Bar code mode" (see Chapter 3.4 Configuration of the serial interfaces, page 12), the JUMO Logoscreen nt appears as a Slave. It waits for the ASCII string that is transmitted by the bar code reader (master).
This interface mode merely requires the setting of the configuration parameters "Baud rate" and "Data format", see Chapter 3.4 Configuration of the serial interfaces, page 12 – no other parameters are needed in this case.

These strings can be used by the paperless recorder for the step-by-step control of a batch sequence (see Operating Manual 70.6580.0.0, Chapter 4.9).



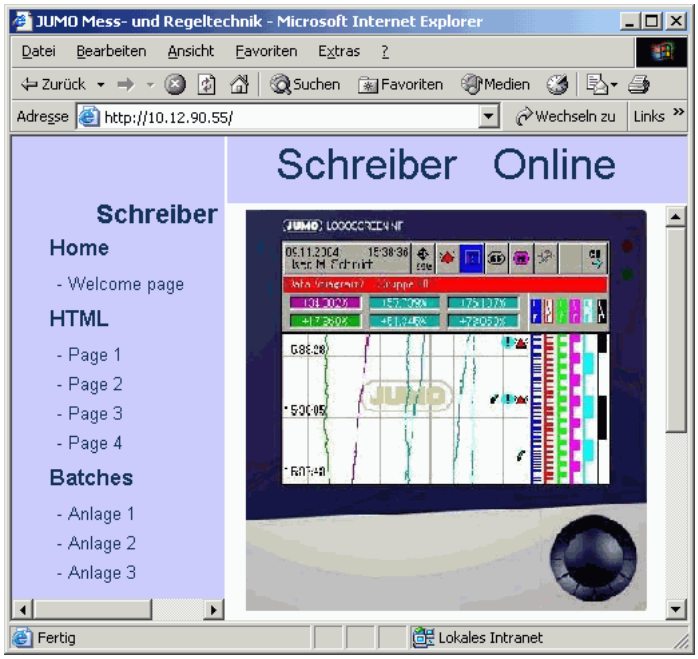
The advantage of an Ethernet connection compared with a serial connection lies in the higher transmission rate and the company-wide accessibility.

## 6.1 HTTP

In this case, the Logoscreen nt is configured as a slave, and handles the incoming queries as a server through port 80. These queries could, for instance, come from a PC with the setup software, PC evaluation software (PCA) or PCA communications software (PCC).

## 6.2 Browser connection

The Logoscreen nt can also be accessed by a browser, using the HTTP protocol. The URL required for this purpose is the IP address of the recorder.




This is used to access the HTML home page “index.htm”, and from here it can branch to the other HTML pages.

Using the setup program *Edit* ➔ *Web server directories* ➔ *Web import*, the start page “index.htm” and other HTML pages can be downloaded to the Logoscreen nt. 512 kB of memory are available for this purpose.

Three HTML process diagrams and three HTML batch pages are stored as templates in the ex-factory instrument.

Device variables can be accessed by special tags in the HTML pages. Support is provided in a Help window for the setup program, in which the device variable can be selected and the corresponding HTML tag copied to the clipboard.

## 6.3 Modbus-TCP

In this case, the Logoscreen nt is configured as a slave, and handles the incoming queries as a server through port 502. The port can also be changed, see Chapter 3.5 Configuration of the Ethernet interface, page 13.

Modbus-TCP is a standardized procedure for packing a Modbus telegram inside a TCP frame (tunneling) and transmitting it via Ethernet.
The Modbus telegram (without CRC) is transmitted with an additional, 6 or 7 byte long MBAP header. The seventh byte corresponds to the first serial byte, but has a different designation in this case.

**Structure of a Modbus-TCP telegram**

| MBAP header | | | | Modbus telegram |
|---|---|---|---|---|
| 2 bytes trans-action ID | 2 bytes protocol ID | 2 bytes length | 1 byte unit ID | other bytes as below, but without CRC |
| Identical for request and response | Must be 0 for Modbus | Length of query or response in bytes from (incl.) unit ID | Corresponds to device address. For TCP, must be 0xFF or 0 (0 = broadcast) | |

As a comparison: the normal Modbus telegram, see Chapter 4.4 Structure of a Modbus telegram, page 17:

| Slave address | Function code | Data field | CRC16 |
|---|---|---|---|
| 1 byte | 1 byte | x bytes | 2 bytes |

This protocol can be used, for instance, by the JUMO SVS 2000 visualization program (or similar programs) to read from and write to the Logoscreen nt over a company-wide Ethernet network. It is possible to access all the device variables in the Modbus address tables (see Chapter 7 Address tables, page 43).

Only one Modbus master (client) at a time can access a Logoscreen nt by means of the Modbus-TCP.

A connection that has been opened by a client will be closed after 30 seconds of inactivity!

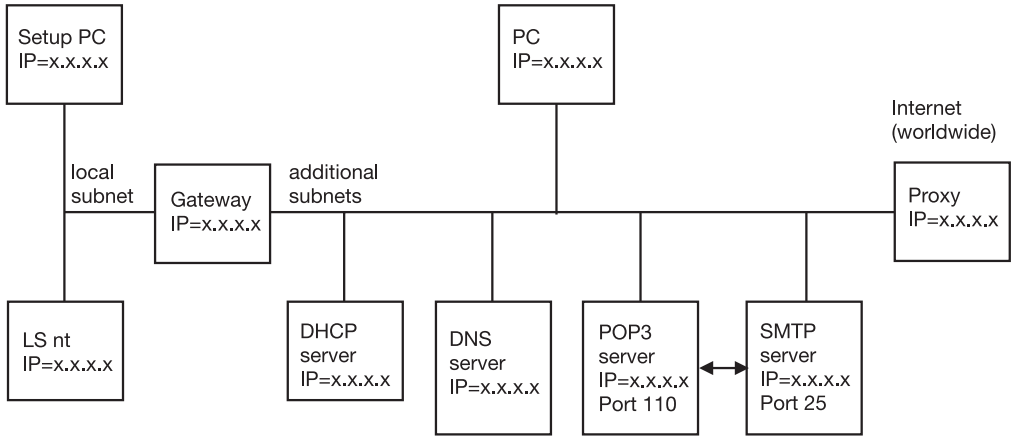
A closed Modbus-TCP port (from the Logoscreen nt or the other end) can only be opened again after 10 seconds.

# 6 Ethernet protocols

## 6.4 E-mail (SMTP and POP3)

The Logoscreen nt can send out e-mails (e.g. alarms). In this case, it is the master (client) and can access an SMTP server through the standard port (25) as well as a POP3 server through the standard port (110).

**Typical networking in a company network**




**Function of the individual nodes**

Gateway
separates local subnets from one another, and thus provides a filtering of the packets. Not every packet is received in every subnet. Packets from outside the local subnet must be addressed to the gateway.

DHCP server
can automatically assign the IP address, subnet mask and gateway address for other nodes at switch-on. These parameters can also be entered manually, and then a DHCP server is not necessary.

DNS server
converts symbolic names into IP addresses, e.g. the query: “www.name.de” will generate the response “www.name.de has IP=10.12.32.45”.

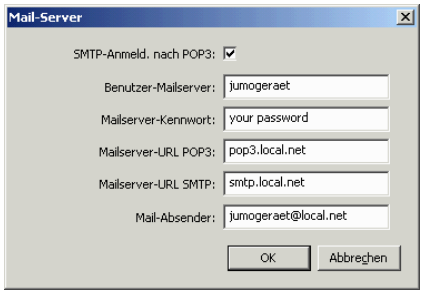
POP3 server
is used to read out received e-mails for a mail account. The POP3 mail account can be accessed after logging in with a user name and a password. A successful log-in procedure will often also enable the transmission authorization for an attached SMTP server.

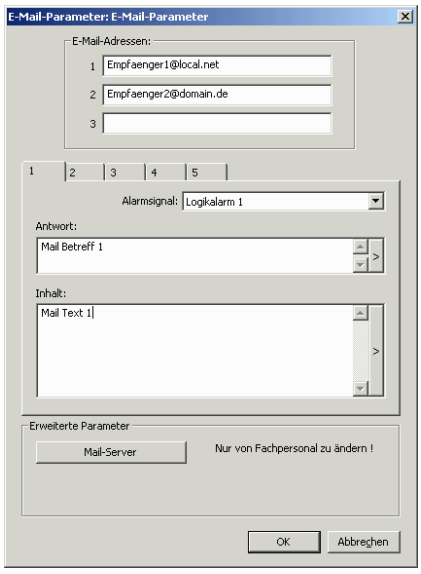
SMTP server
is used for sending e-mails. In many networks, the authorization for transmission via a mail account must first of all be enabled by a log-in to the corresponding POP3 server.

Proxy
serves as a gateway between the local company network and the Internet. This is also the place where the local IP addresses (as used in the company network) are converted into unique IP addresses (as used on the Internet).
The Logoscreen nt software cannot address a proxy!

**Parameters for the mail server and e-mail**

These parameters can only be edited though the setup program:
*Edit* ➔ *E-mail parameters*

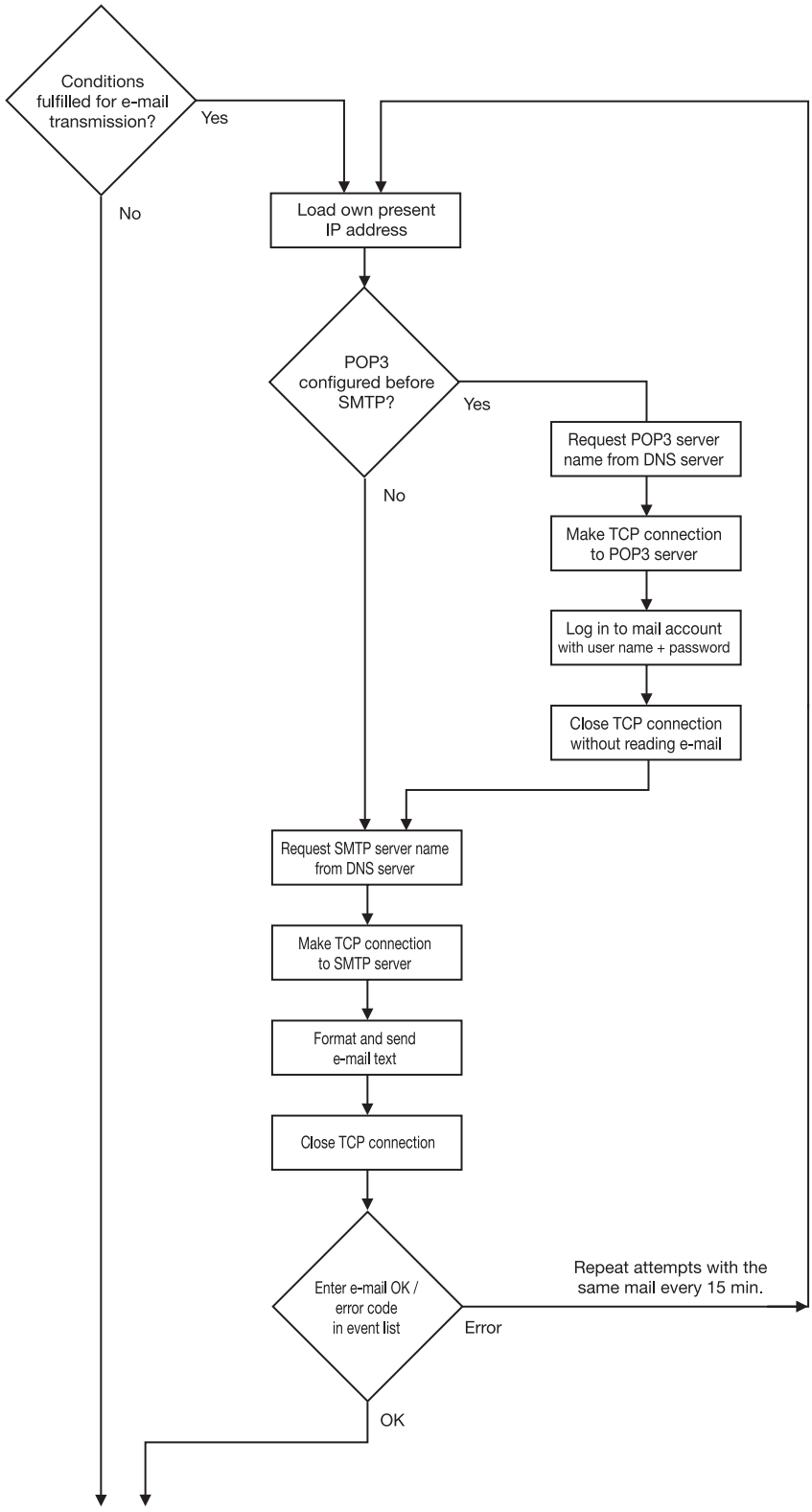



Here you must enter a mail server that is located within the company network (not on the Internet)! This mail server should also have the capability of passing on e-mails to the Internet.

**Sending an e-mail via the Internet**

Many of the steps here depend on the device parameters that have been configured. From the error code for the event entry (see Chapter 4.9.3 JUMO error codes as integer return values, page 27), particularly error codes 120 to 173, it is possible to deduce that a parameter has been set incorrectly. For instance, an incorrectly entered IP for the DNS server produces the error code 153 = “no IP received by DNS”.



All process values (variables) together with their addresses, data types and access modes are described below.

References are as follows:

| | |
|---|---|
| R | read access only |
| W | write access only |
| R/W | read and write access |
| char | ASCII character (8 bits) |
| byte | byte (8 bits) |
| int | integer (16 bits) |
| char xx | character string of length xx;<br>xx = length, including the string termination character “0” |
| bit x | bit No. x |
| float | float value (4 bytes) |
| double | double value (8 bytes) |
| bool | boolean variable |

The process values are divided into logical groups.

In the following address tables, bit 0 is always the least significant bit.

## 7.1 Modbus addresses for important device and process data

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x0009 | R | char 12 | software version |
| 0x1000 | R | char 46 | device name |
| 0x1017 | R/W | byte | display brightness |
| | | | |
| 0x120F | R | long | hardware count 1 |
| 0x1211 | R | long | hardware count 2 |
| 0x1213 | R | long | hardware count 3 |
| 0x1215 | R | long | hardware count 4 |
| 0x1217 | R | long | hardware count 5 |
| 0x1219 | R | long | hardware count 6 |
| | | | |
| 0x121B | R | char 11 | software version, card 1 |
| 0x1221 | R | char 11 | software version, card 2 |
| 0x1227 | R | char 11 | software version, card 3 |
| 0x122D | R | char 11 | software version, Profibus |

# 7 Address tables

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x1257 | R | float | filtered analog value 1 |
| 0x1259 | R | float | filtered analog value 2 |
| 0x125B | R | float | filtered analog value 3 |
| 0x125D | R | float | filtered analog value 4 |
| 0x125F | R | float | filtered analog value 5 |
| 0x1261 | R | float | filtered analog value 6 |
| 0x1263 | R | float | filtered analog value 7 |
| 0x1265 | R | float | filtered analog value 8 |
| 0x1267 | R | float | filtered analog value 9 |
| 0x1269 | R | float | filtered analog value 10 |
| 0x126B | R | float | filtered analog value 11 |
| 0x126D | R | float | filtered analog value 12 |
| 0x126F | R | float | filtered analog value 13 |
| 0x1271 | R | float | filtered analog value 14 |
| 0x1273 | R | float | filtered analog value 15 |
| 0x1275 | R | float | filtered analog value 16 |
| 0x1277 | R | float | filtered analog value 17 |
| 0x1279 | R | float | filtered analog value 18 |
| | | | |
| 0x127B | R | bool | analog alarm 1, channel 1 |
| 0x127C | R | bool | analog alarm 1, channel 2 |
| 0x127D | R | bool | analog alarm 1, channel 3 |
| 0x127E | R | bool | analog alarm 1, channel 4 |
| 0x127F | R | bool | analog alarm 1, channel 5 |
| 0x1280 | R | bool | analog alarm 1, channel 6 |
| 0x1281 | R | bool | analog alarm 1, channel 7 |
| 0x1282 | R | bool | analog alarm 1, channel 8 |
| 0x1283 | R | bool | analog alarm 1, channel 9 |
| 0x1284 | R | bool | analog alarm 1, channel 10 |
| 0x1285 | R | bool | analog alarm 1, channel 11 |
| 0x1286 | R | bool | analog alarm 1, channel 12 |
| 0x1287 | R | bool | analog alarm 1, channel 13 |
| 0x1288 | R | bool | analog alarm 1, channel 14 |
| 0x1289 | R | bool | analog alarm 1, channel 15 |
| 0x128A | R | bool | analog alarm 1, channel 16 |
| 0x128B | R | bool | analog alarm 1, channel 17 |
| 0x128C | R | bool | analog alarm 1, channel 18 |
| | | | |
| 0x128D | R | bool | analog alarm 2, channel 1 |
| 0x128E | R | bool | analog alarm 2, channel 2 |
| 0x128F | R | bool | analog alarm 2, channel 3 |
| 0x1290 | R | bool | analog alarm 2, channel 4 |
| 0x1291 | R | bool | analog alarm 2, channel 5 |
| 0x1292 | R | bool | analog alarm 2, channel 6 |
| 0x1293 | R | bool | analog alarm 2, channel 7 |
| 0x1294 | R | bool | analog alarm 2, channel 8 |
| 0x1295 | R | bool | analog alarm 2, channel 9 |
| 0x1296 | R | bool | analog alarm 2, channel 10 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x1297 | R | bool | analog alarm 2, channel 11 |
| 0x1298 | R | bool | analog alarm 2, channel 12 |
| 0x1299 | R | bool | analog alarm 2, channel 13 |
| 0x129A | R | bool | analog alarm 2, channel 14 |
| 0x129B | R | bool | analog alarm 2, channel 15 |
| 0x129C | R | bool | analog alarm 2, channel 16 |
| 0x129D | R | bool | analog alarm 2, channel 17 |
| 0x129E | R | bool | analog alarm 2, channel 18 |
| | | | |
| 0x12AA | R | float | external, limit-checked, analog value 1 |
| 0x12AC | R | float | external, limit-checked, analog value 2 |
| 0x12AE | R | float | external, limit-checked, analog value 3 |
| 0x12B0 | R | float | external, limit-checked, analog value 4 |
| 0x12B2 | R | float | external, limit-checked, analog value 5 |
| 0x12B4 | R | float | external, limit-checked, analog value 6 |
| 0x12B6 | R | float | external, limit-checked, analog value 7 |
| 0x12B8 | R | float | external, limit-checked, analog value 8 |
| 0x12BA | R | float | external, limit-checked, analog value 9 |
| 0x12BC | R | float | external, limit-checked, analog value 10 |
| 0x12BE | R | float | external, limit-checked, analog value 11 |
| 0x12C0 | R | float | external, limit-checked, analog value 12 |
| 0x12C2 | R | float | external, limit-checked, analog value 13 |
| 0x12C4 | R | float | external, limit-checked, analog value 14 |
| 0x12C6 | R | float | external, limit-checked, analog value 15 |
| 0x12C8 | R | float | external, limit-checked, analog value 16 |
| 0x12CA | R | float | external, limit-checked, analog value 17 |
| 0x12CC | R | float | external, limit-checked, analog value 18 |
| 0x12CE | R | float | external, limit-checked, analog value 19 |
| 0x12D0 | R | float | external, limit-checked, analog value 20 |
| 0x12D2 | R | float | external, limit-checked, analog value 21 |
| 0x12D4 | R | float | external, limit-checked, analog value 22 |
| 0x12D6 | R | float | external, limit-checked, analog value 23 |
| 0x12D8 | R | float | external, limit-checked, analog value 24 |
| | | | |
| 0x12DA | R/W | float | external analog value, interface 1 |
| 0x12DC | R/W | float | external analog value, interface 2 |
| 0x12DE | R/W | float | external analog value, interface 3 |
| 0x12E0 | R/W | float | external analog value, interface 4 |
| 0x12E2 | R/W | float | external analog value, interface 5 |
| 0x12E4 | R/W | float | external analog value, interface 6 |
| 0x12E6 | R/W | float | external analog value, interface 7 |
| 0x12E8 | R/W | float | external analog value, interface 8 |
| 0x12EA | R/W | float | external analog value, interface 9 |
| 0x12EC | R/W | float | external analog value, interface 10 |
| 0x12EE | R/W | float | external analog value, interface 11 |
| 0x12F0 | R/W | float | external analog value, interface 12 |
| 0x12F2 | R/W | float | external analog value, interface 13 |
| 0x12F4 | R/W | float | external analog value, interface 14 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x12F6 | R/W | float | external analog value, interface 15 |
| 0x12F8 | R/W | float | external analog value, interface 16 |
| 0x12FA | R/W | float | external analog value, interface 17 |
| 0x12FC | R/W | float | external analog value, interface 18 |
| 0x12FE | R/W | float | external analog value, interface 19 |
| 0x1300 | R/W | float | external analog value, interface 20 |
| 0x1302 | R/W | float | external analog value, interface 21 |
| 0x1304 | R/W | float | external analog value, interface 22 |
| 0x1306 | R/W | float | external analog value, interface 23 |
| 0x1308 | R/W | float | external analog value, interface 24 |
| | | | |
| 0x130A | R | bool | external analog alarm 1, channel 1 |
| 0x130B | R | bool | external analog alarm 1, channel 2 |
| 0x130C | R | bool | external analog alarm 1, channel 3 |
| 0x130D | R | bool | external analog alarm 1, channel 4 |
| 0x130E | R | bool | external analog alarm 1, channel 5 |
| 0x130F | R | bool | external analog alarm 1, channel 6 |
| 0x1310 | R | bool | external analog alarm 1, channel 7 |
| 0x1311 | R | bool | external analog alarm 1, channel 8 |
| 0x1312 | R | bool | external analog alarm 1, channel 9 |
| 0x1313 | R | bool | external analog alarm 1, channel 10 |
| 0x1314 | R | bool | external analog alarm 1, channel 11 |
| 0x1315 | R | bool | external analog alarm 1, channel 12 |
| 0x1316 | R | bool | external analog alarm 1, channel 13 |
| 0x1317 | R | bool | external analog alarm 1, channel 14 |
| 0x1318 | R | bool | external analog alarm 1, channel 15 |
| 0x1319 | R | bool | external analog alarm 1, channel 16 |
| 0x131A | R | bool | external analog alarm 1, channel 17 |
| 0x131B | R | bool | external analog alarm 1, channel 18 |
| 0x131C | R | bool | external analog alarm 1, channel 19 |
| 0x131D | R | bool | external analog alarm 1, channel 20 |
| 0x131E | R | bool | external analog alarm 1, channel 21 |
| 0x131F | R | bool | external analog alarm 1, channel 22 |
| 0x1320 | R | bool | external analog alarm 1, channel 23 |
| 0x1321 | R | bool | external analog alarm 1, channel 24 |
| | | | |
| 0x1322 | R | bool | external analog alarm 2, channel 1 |
| 0x1323 | R | bool | external analog alarm 2, channel 2 |
| 0x1324 | R | bool | external analog alarm 2, channel 3 |
| 0x1325 | R | bool | external analog alarm 2, channel 4 |
| 0x1326 | R | bool | external analog alarm 2, channel 5 |
| 0x1327 | R | bool | external analog alarm 2, channel 6 |
| 0x1328 | R | bool | external analog alarm 2, channel 7 |
| 0x1329 | R | bool | external analog alarm 2, channel 8 |
| 0x132A | R | bool | external analog alarm 2, channel 9 |
| 0x132B | R | bool | external analog alarm 2, channel 10 |
| 0x132C | R | bool | external analog alarm 2, channel 11 |
| 0x132D | R | bool | external analog alarm 2, channel 12 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x132E | R | bool | external analog alarm 2, channel 13 |
| 0x132F | R | bool | external analog alarm 2, channel 14 |
| 0x1330 | R | bool | external analog alarm 2, channel 15 |
| 0x1331 | R | bool | external analog alarm 2, channel 16 |
| 0x1332 | R | bool | external analog alarm 2, channel 17 |
| 0x1333 | R | bool | external analog alarm 2, channel 18 |
| 0x1334 | R | bool | external analog alarm 2, channel 19 |
| 0x1335 | R | bool | external analog alarm 2, channel 20 |
| 0x1336 | R | bool | external analog alarm 2, channel 21 |
| 0x1337 | R | bool | external analog alarm 2, channel 22 |
| 0x1338 | R | bool | external analog alarm 2, channel 23 |
| 0x1339 | R | bool | external analog alarm 2, channel 24 |
| | | | |
| 0x133A | R | bool | binary inputs/outputs, status 1 |
| 0x133B | R | bool | binary inputs/outputs, status 2 |
| 0x133C | R | bool | binary inputs/outputs, status 3 |
| 0x133D | R | bool | binary inputs/outputs, status 4 |
| 0x133E | R | bool | binary inputs/outputs, status 5 |
| 0x133F | R | bool | binary inputs/outputs, status 6 |
| 0x1340 | R | bool | binary inputs/outputs, status 7 |
| 0x1341 | R | bool | binary inputs/outputs, status 8 |
| 0x1342 | R | bool | binary inputs/outputs, status 9 |
| 0x1343 | R | bool | binary inputs/outputs, status 10 |
| 0x1344 | R | bool | binary inputs/outputs, status 11 |
| 0x1345 | R | bool | binary inputs/outputs, status 12 |
| 0x1346 | R | bool | binary inputs/outputs, status 13 |
| 0x1347 | R | bool | binary inputs/outputs, status 14 |
| 0x1348 | R | bool | binary inputs/outputs, status 15 |
| 0x1349 | R | bool | binary inputs/outputs, status 16 |
| 0x134A | R | bool | binary inputs/outputs, status 17 |
| 0x134B | R | bool | binary inputs/outputs, status 18 |
| 0x134C | R | bool | binary inputs/outputs, status 19 |
| 0x134D | R | bool | binary inputs/outputs, status 20 |
| 0x134E | R | bool | binary inputs/outputs, status 21 |
| 0x134F | R | bool | binary inputs/outputs, status 22 |
| 0x1350 | R | bool | binary inputs/outputs, status 23 |
| 0x1351 | R | bool | binary inputs/outputs, status 24 |
| | | | |
| 0x1352 | R | bool | binary inputs/outputs, alarm 1 |
| 0x1353 | R | bool | binary inputs/outputs, alarm 2 |
| 0x1354 | R | bool | binary inputs/outputs, alarm 3 |
| 0x1355 | R | bool | binary inputs/outputs, alarm 4 |
| 0x1356 | R | bool | binary inputs/outputs, alarm 5 |
| 0x1357 | R | bool | binary inputs/outputs, alarm 6 |
| 0x1358 | R | bool | binary inputs/outputs, alarm 7 |
| 0x1359 | R | bool | binary inputs/outputs, alarm 8 |
| 0x135A | R | bool | binary inputs/outputs, alarm 9 |
| 0x135B | R | bool | binary inputs/outputs, alarm 10 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x135C | R | bool | binary inputs/outputs, alarm 11 |
| 0x135D | R | bool | binary inputs/outputs, alarm 12 |
| 0x135E | R | bool | binary inputs/outputs, alarm 13 |
| 0x135F | R | bool | binary inputs/outputs, alarm 14 |
| 0x1360 | R | bool | binary inputs/outputs, alarm 15 |
| 0x1361 | R | bool | binary inputs/outputs, alarm 16 |
| 0x1362 | R | bool | binary inputs/outputs, alarm 17 |
| 0x1363 | R | bool | binary inputs/outputs, alarm 18 |
| 0x1364 | R | bool | binary inputs/outputs, alarm 19 |
| 0x1365 | R | bool | binary inputs/outputs, alarm 20 |
| 0x1366 | R | bool | binary inputs/outputs, alarm 21 |
| 0x1367 | R | bool | binary inputs/outputs, alarm 22 |
| 0x1368 | R | bool | binary inputs/outputs, alarm 23 |
| 0x1369 | R | bool | binary inputs/outputs, alarm 24 |
| | | | |
| 0x136A | R/W | char 64 | external batch text 1 |
| 0x138A | R/W | char 64 | external batch text 2 |
| 0x13AA | R/W | char 64 | external batch text 3 |
| 0x13CA | R/W | char 64 | external batch text 4 |
| 0x13EA | R/W | char 64 | external batch text 5 |
| 0x140A | R/W | char 64 | external batch text 6 |
| 0x142A | R/W | char 64 | external batch text 7 |
| 0x144A | R/W | char 64 | external batch text 8 |
| 0x146A | R/W | char 64 | external batch text 9 |
| | | | |
| 0x148A | R/W | char 94 | external event text, group 1 |
| 0x14B9 | R/W | char 94 | external event text, group 2 |
| 0x14E8 | R/W | char 94 | external event text, group 3 |
| 0x1517 | R/W | char 94 | external event text, group 4 |
| 0x1546 | R/W | char 94 | external event text, group 5 |
| 0x1575 | R/W | char 94 | external event text, group 6 |
| 0x15A4 | R/W | char 94 | external event text, group 7 |
| 0x15D3 | R/W | char 94 | external event text, group 8 |
| 0x1602 | R/W | char 94 | external event text, group 9 |
| | | | |
| 0x1631 | R | bool | relay output 1 |
| 0x1632 | R | bool | relay output 2 |
| 0x1633 | R | bool | relay output 3 |
| 0x1634 | R | bool | relay output 4 |
| 0x1635 | R | bool | relay output 5 |
| 0x1636 | R | bool | relay output 6 |
| 0x1637 | R | bool | relay output 7 |
| | | | |
| 0x1638 | R/W | bool | external binary input, status 1 |
| 0x1639 | R/W | bool | external binary input, status 2 |
| 0x163A | R/W | bool | external binary input, status 3 |
| 0x163B | R/W | bool | external binary input, status 4 |
| 0x163C | R/W | bool | external binary input, status 5 |
| 0x163D | R/W | bool | external binary input, status 6 |
| 0x163E | R/W | bool | external binary input, status 7 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x163F | R/W | bool | external binary input, status 8 |
| 0x1640 | R/W | bool | external binary input, status 9 |
| 0x1641 | R/W | bool | external binary input, status 10 |
| 0x1642 | R/W | bool | external binary input, status 11 |
| 0x1643 | R/W | bool | external binary input, status 12 |
| 0x1644 | R/W | bool | external binary input, status 13 |
| 0x1645 | R/W | bool | external binary input, status 14 |
| 0x1646 | R/W | bool | external binary input, status 15 |
| 0x1647 | R/W | bool | external binary input, status 16 |
| 0x1648 | R/W | bool | external binary input, status 17 |
| 0x1649 | R/W | bool | external binary input, status 18 |
| 0x164A | R/W | bool | external binary input, status 19 |
| 0x164B | R/W | bool | external binary input, status 20 |
| 0x164C | R/W | bool | external binary input, status 21 |
| 0x164D | R/W | bool | external binary input, status 22 |
| 0x164E | R/W | bool | external binary input, status 23 |
| 0x164F | R/W | bool | external binary input, status 24 |
| | | | |
| 0x1650 | R | bool | external binary input, alarm 1 |
| 0x1651 | R | bool | external binary input, alarm 2 |
| 0x1652 | R | bool | external binary input, alarm 3 |
| 0x1653 | R | bool | external binary input, alarm 4 |
| 0x1654 | R | bool | external binary input, alarm 5 |
| 0x1655 | R | bool | external binary input, alarm 6 |
| 0x1656 | R | bool | external binary input, alarm 7 |
| 0x1657 | R | bool | external binary input, alarm 8 |
| 0x1658 | R | bool | external binary input, alarm 9 |
| 0x1659 | R | bool | external binary input, alarm 10 |
| 0x165A | R | bool | external binary input, alarm 11 |
| 0x165B | R | bool | external binary input, alarm 12 |
| 0x165C | R | bool | external binary input, alarm 13 |
| 0x165D | R | bool | external binary input, alarm 14 |
| 0x165E | R | bool | external binary input, alarm 15 |
| 0x165F | R | bool | external binary input, alarm 16 |
| 0x1660 | R | bool | external binary input, alarm 17 |
| 0x1661 | R | bool | external binary input, alarm 18 |
| 0x1662 | R | bool | external binary input, alarm 19 |
| 0x1663 | R | bool | external binary input, alarm 20 |
| 0x1664 | R | bool | external binary input, alarm 21 |
| 0x1665 | R | bool | external binary input, alarm 22 |
| 0x1666 | R | bool | external binary input, alarm 23 |
| 0x1667 | R | bool | external binary input, alarm 24 |
| | | | |
| 0x1668 | R | float | math result 1 |
| 0x166A | R | float | math result 2 |
| 0x166C | R | float | math result 3 |
| 0x166E | R | float | math result 4 |
| 0x1670 | R | float | math result 5 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x1672 | R | float | math result 6 |
| 0x1674 | R | float | math result 7 |
| 0x1676 | R | float | math result 8 |
| 0x1678 | R | float | math result 9 |
| | | | |
| 0x167A | R | bool | math alarm 1, 1 |
| 0x167B | R | bool | math alarm 1, 2 |
| 0x167C | R | bool | math alarm 1, 3 |
| 0x167D | R | bool | math alarm 1, 4 |
| 0x167E | R | bool | math alarm 1, 5 |
| 0x167F | R | bool | math alarm 1, 6 |
| 0x1680 | R | bool | math alarm 1, 7 |
| 0x1681 | R | bool | math alarm 1, 8 |
| 0x1682 | R | bool | math alarm 1, 9 |
| | | | |
| 0x1683 | R | bool | math alarm 2, 1 |
| 0x1684 | R | bool | math alarm 2, 2 |
| 0x1685 | R | bool | math alarm 2, 3 |
| 0x1686 | R | bool | math alarm 2, 4 |
| 0x1687 | R | bool | math alarm 2, 5 |
| 0x1688 | R | bool | math alarm 2, 6 |
| 0x1689 | R | bool | math alarm 2, 7 |
| 0x168A | R | bool | math alarm 2, 8 |
| 0x168B | R | bool | math alarm 2, 9 |
| | | | |
| 0x168C | R | bool | logic result 1 |
| 0x168D | R | bool | logic result 2 |
| 0x168E | R | bool | logic result 3 |
| 0x168F | R | bool | logic result 4 |
| 0x1690 | R | bool | logic result 5 |
| 0x1691 | R | bool | logic result 6 |
| 0x1692 | R | bool | logic result 7 |
| 0x1693 | R | bool | logic result 8 |
| 0x1694 | R | bool | logic result 9 |
| | | | |
| 0x1695 | R | bool | logic alarm 1 |
| 0x1696 | R | bool | logic alarm 2 |
| 0x1697 | R | bool | logic alarm 3 |
| 0x1698 | R | bool | logic alarm 4 |
| 0x1699 | R | bool | logic alarm 5 |
| 0x169A | R | bool | logic alarm 6 |
| 0x169B | R | bool | logic alarm 7 |
| 0x169C | R | bool | logic alarm 8 |
| 0x169D | R | bool | logic alarm 9 |
| | | | |
| 0x169E | R | bool | limit comparator status 1 |
| 0x169F | R | bool | limit comparator status 2 |
| 0x16A0 | R | bool | limit comparator status 3 |
| 0x16A1 | R | bool | limit comparator status 4 |
| 0x16A2 | R | bool | limit comparator status 5 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x16A3 | R | bool | limit comparator status 6 |
| 0x16A4 | R | bool | limit comparator status 7 |
| 0x16A5 | R | bool | limit comparator status 8 |
| 0x16A6 | R | bool | limit comparator status 9 |
| | | | |
| 0x16A7 | R | bool | limit comparator alarm 1 |
| 0x16A8 | R | bool | limit comparator alarm 2 |
| 0x16A9 | R | bool | limit comparator alarm 3 |
| 0x16AA | R | bool | limit comparator alarm 4 |
| 0x16AB | R | bool | limit comparator alarm 5 |
| 0x16AC | R | bool | limit comparator alarm 6 |
| 0x16AD | R | bool | limit comparator alarm 7 |
| 0x16AE | R | bool | limit comparator alarm 8 |
| 0x16AF | R | bool | limit comparator alarm 9 |
| | | | |
| 0x16B0 | R | bool | counter alarm 1, 1 |
| 0x16B1 | R | bool | counter alarm 1, 2 |
| 0x16B2 | R | bool | counter alarm 1, 3 |
| 0x16B3 | R | bool | counter alarm 1, 4 |
| 0x16B4 | R | bool | counter alarm 1, 5 |
| 0x16B5 | R | bool | counter alarm 1, 6 |
| 0x16B6 | R | bool | counter alarm 1, 7 |
| 0x16B7 | R | bool | counter alarm 1, 8 |
| 0x16B8 | R | bool | counter alarm 1, 9 |
| 0x16B9 | R | bool | counter alarm 1, 10 |
| 0x16BA | R | bool | counter alarm 1, 11 |
| 0x16BB | R | bool | counter alarm 1, 12 |
| 0x16BC | R | bool | counter alarm 1, 13 |
| 0x16BD | R | bool | counter alarm 1, 14 |
| 0x16BE | R | bool | counter alarm 1, 15 |
| 0x16BF | R | bool | counter alarm 1, 16 |
| 0x16C0 | R | bool | counter alarm 1, 17 |
| 0x16C1 | R | bool | counter alarm 1, 18 |
| 0x16C2 | R | bool | counter alarm 1, 19 |
| 0x16C3 | R | bool | counter alarm 1, 20 |
| 0x16C4 | R | bool | counter alarm 1, 21 |
| 0x16C5 | R | bool | counter alarm 1, 22 |
| 0x16C6 | R | bool | counter alarm 1, 23 |
| 0x16C7 | R | bool | counter alarm 1, 24 |
| 0x16C8 | R | bool | counter alarm 1, 25 |
| 0x16C9 | R | bool | counter alarm 1, 26 |
| 0x16CA | R | bool | counter alarm 1, 27 |
| | | | |
| 0x16CB | R | bool | counter alarm 2, 1 |
| 0x16CC | R | bool | counter alarm 2, 2 |
| 0x16CD | R | bool | counter alarm 2, 3 |
| 0x16CE | R | bool | counter alarm 2, 4 |
| 0x16CF | R | bool | counter alarm 2, 5 |
| 0x16D0 | R | bool | counter alarm 2, 6 |

| Address | Access | Data type | Signal designation |
|---|---|---|---|
| 0x16D1 | R | bool | counter alarm 2, 7 |
| 0x16D2 | R | bool | counter alarm 2, 8 |
| 0x16D3 | R | bool | counter alarm 2, 9 |
| 0x16D4 | R | bool | counter alarm 2, 10 |
| 0x16D5 | R | bool | counter alarm 2, 11 |
| 0x16D6 | R | bool | counter alarm 2, 12 |
| 0x16D7 | R | bool | counter alarm 2, 13 |
| 0x16D8 | R | bool | counter alarm 2, 14 |
| 0x16D9 | R | bool | counter alarm 2, 15 |
| 0x16DA | R | bool | counter alarm 2, 16 |
| 0x16DB | R | bool | counter alarm 2, 17 |
| 0x16DC | R | bool | counter alarm 2, 18 |
| 0x16DD | R | bool | counter alarm 2, 19 |
| 0x16DE | R | bool | counter alarm 2, 20 |
| 0x16DF | R | bool | counter alarm 2, 21 |
| 0x16E0 | R | bool | counter alarm 2, 22 |
| 0x16E1 | R | bool | counter alarm 2, 23 |
| 0x16E2 | R | bool | counter alarm 2, 24 |
| 0x16E3 | R | bool | counter alarm 2, 25 |
| 0x16E4 | R | bool | counter alarm 2, 26 |
| 0x16E5 | R | bool | counter alarm 2, 27 |
| | | | |
| 0x17B0 | R | bool | combination alarm |
| | | | |
| 0x17B1 | R | byte | Ethernet IP address, byte 1 |
| 0x17B2 | R | byte | Ethernet IP address, byte 2 |
| 0x17B3 | R | byte | Ethernet IP address, byte 3 |
| 0x17B4 | R | byte | Ethernet IP address, byte 4 |
| | | | |
| 0x9000 | R/W | char 1204 | recipe for active batch 0 |
| 0x9400 | R/W | char 1204 | recipe for active batch 1 |
| 0x9800 | R/W | char 1204 | recipe for active batch 2 |
| | | | |

# A



# B



# C



# D



# E



# F



# H



# I



# M

## O



## P



## R



## S



## T



## U



## W

**JUMO GmbH & Co. KG**

Street address:
Moltkestraße 13 - 31
36039 Fulda, Germany
Delivery address:
Mackenrodtstraße 14
36039 Fulda, Germany
Postal address:
36035 Fulda, Germany
Phone: +49 661 6003-0
Fax: +49 661 6003-607
e-mail: mail@jumo.net
Internet: www.jumo.net

**JUMO Instrument Co. Ltd.**

JUMO House
Temple Bank, Riverway
Harlow, Essex CM20 2TT, UK
Phone: +44 1279 635533
Fax: +44 1279 635262
e-mail: sales@jumo.co.uk
Internet: www.jumo.co.uk

**JUMO Process Control, Inc.**

8 Technology Boulevard
Canastota, NY 13032, USA
Phone: 315-697-JUMO
1-800-554-JUMO
Fax: 315-697-5867
e-mail: info@jumo.us
Internet: www.jumo.us