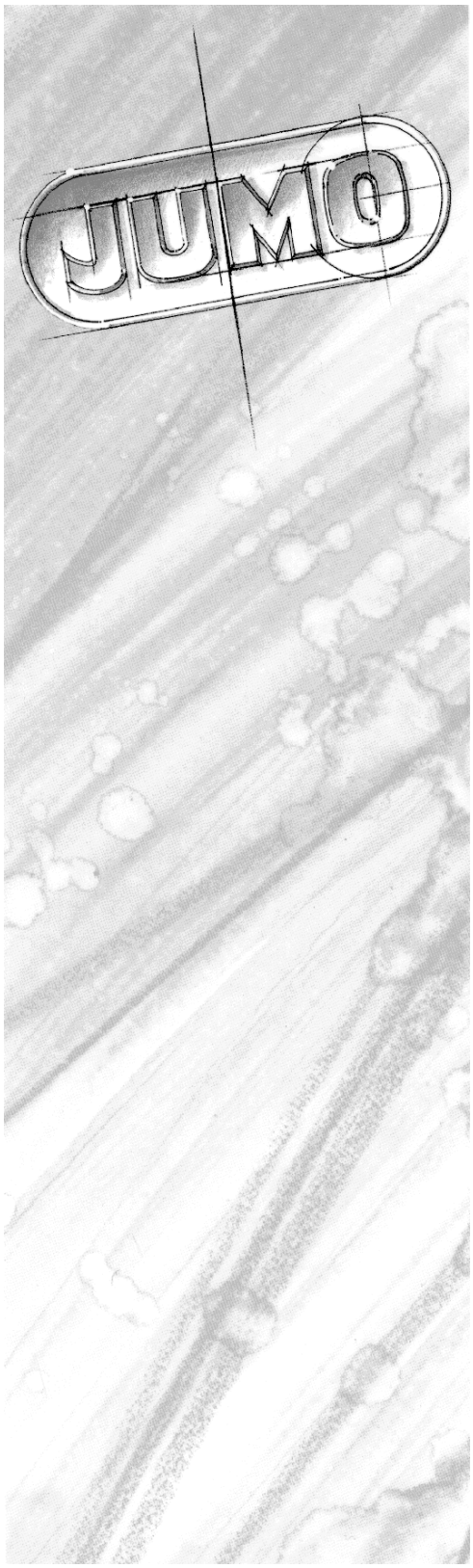




# JUMO DICON 1000

Universal Process Controller

# JUMO DICON 1001

Universal Profile Controller

B 70.3560.2

Interface Description

3.99/00316584

# Contents

# Contents

# 1 Introduction

## 1.1 Preliminary remarks

Please read this Manual before starting up the interface. Keep this Manual in a place which is at all times accessible to all users.

Please assist us to improve this Manual where necessary.

We are always grateful for your suggestions.

Phone Germany (06 61)60 03-7 27
abroad (+49) 661 60 03-0

Fax Germany (06 61)60 03-5 08
abroad (+49) 661 60 03-607

All necessary information for operating the interface are described in this Manual. If any problems should arise during start-up do not carry out any manipulations which are not permitted. You could endanger your rights under the warranty!

Please contact the nearest JUMO office or the main factory.

When returning chassis, modules or components the rules of EN 100 015 “Protection of electrostatic sensitive devices” have to be observed. Use only the appropriate **ESD** packaging material for transport.

Please note that we can not be held liable for any damage caused by **ESD** (electrostatic discharges).

## 1.2 Typographical conventions

### 1.2.1 Warning signs

The signs for **Danger** and **Warning** are used in this Manual under the following conditions:

**Danger** This mark is used when there may be **danger to personnel** if the instructions are disregarded or not followed accurately.

**Warning** This mark is used when there may be **damage to equipment or data** if the instructions are disregarded or not followed accurately.

**Warning** This mark is used where precautions have to be observed in handling components endangered by electrostatic discharges.

### 1.2.2 Note signs

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

**Reference** This symbol refers to additional information in other handbooks, chapters or sections.

abc[1] **Footnote** Footnotes are notes which refer to certain points in the text. Footnotes consist of 2 parts:

the text marking and the footnote text.

The text markings are arranged as continuous raised numbers.

The footnote text (in smaller typerface) is placed at the bottom of the page and starts with a number and a full stop.

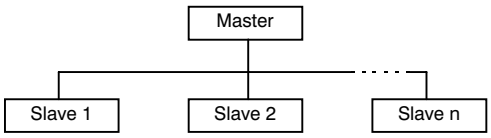
### 1.2.3 Presentation

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

# 2 Protocol description

## 2.1 Master-slave principle

The communication between a PC (master) and an instrument (slave) using MODbus/Jbus takes place according to the master-slave principle in the form of data request/instruction - response.




The master controls the data exchange, the slaves only have a response function. They are identified by their instrument address. A maximum of 255 slaves can be accessed.

## 2.2 Transmission mode (RTU)

The transmission mode used is the RTU mode (Remote Terminal Unit). Data are transmitted in binary form (hexadecimal) with 8 bits, 16 bits for integers, and 32 bits for floating values. The LSB (least significant bit) is transmitted first. The ASCII operating mode is not supported.

**Data format**

The data format describes the arrangement of a byte transmitted. The data format can be as follows:

| Data word | Parity bit | Stop bit 1/2 bit | Bit number |
|---|---|---|---|
| 8 bit | — | 1 | 9 |
| 8 bit | — | 2 | 10 |
| 8 bit | even | 1 | 10 |
| 8 bit | odd | 1 | 10 |

## 2.3 Instrument address

The address of the slaves can be set between 1 and 255. Address 0 is reserved.

☞ A maximum of 31 slaves can be accessed via the RS422/RS485 interface.

**Possibilities of data exchange**

**Query** Data request/instruction from the master to a slave via the appropriate address.
The slave accessed then responds.

**Broadcast** Instruction from the master to all slaves via address 0.
The slaves in the system do not respond.
A data request is not appropriate in this case.

For example, a certain setpoint can be transmitted to all slaves. The correct acceptance of the value by the slaves should in that case be checked by a subsequent read-out of the setpoint.

## 2.4 Timing of the communication

Start and end of a data block are identified by transmission pauses. The maximum permitted interval between two consecutive characters is three times the time for transmitting a character.

The character transmission time (time for transmitting a character) depends on the baud rate and on the data format used.

If the data format has 8 data bits, no parity bit and one stop bit, then:

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

For other data formats:

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

**Sequence**

| |
|---|
| **Data request from master**<br>transmission time = n characters * 1000 * x bits/baud rate |
| Marker for data request end<br>3 characters * 1000 * x bits/baud rate |
| Processing the data request by the slave (250 msec max. ) |
| **Response of the slave**<br>transmisssion time = n characters * 1000 * x bits/baud rate |
| Marker for response end<br>3 characters * 1000 * x bits/baud rate |

**Example**

Marker for end of data request or end of response for 10/9 bit data format

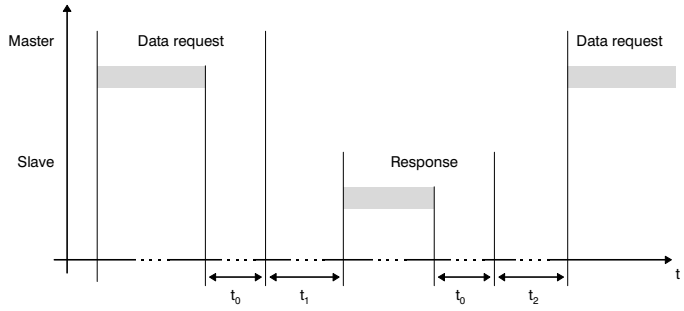
Waiting time = 3 characters * 1000 * 10 bits/baud rate

| **Baud rate** [baud] | **Data format [**bit] | **Waiting time** [msec] |
|---|---|---|
| 187k | 10 | 0.160 |
| | 9 | 0.144 |
| 125k | 10 | 0.240 |
| | 9 | 0.216 |
| 38400 | 10 | 0.781 |
| | 9 | 0.703 |
| 19200 | 10 | 1.563 |
| | 9 | 1.406 |
| 9600 | 10 | 3.125 |
| | 9 | 2.813 |
| 4800 | 10 | 6.250 |
| | 9 | 5.625 |
| 2400 | 10 | 12.500 |
| | 9 | 11.250 |
| 1200 | 10 | 25.000 |
| | 9 | 22.500 |
| 600 | 10 | 50.000 |
| | 9 | 45.000 |
| 300 | 10 | 100.000 |
| | 9 | 90.000 |
| 150 | 10 | 200.000 |
| | 9 | 180.000 |

### 2.4.1 Timing of a data request

**Timing scheme**

A data request runs according to the following timing scheme:




$t_0$ end marker = 3 characters
(the time depends on the baud rate)

$t_1$ this time depends on internal processing.
The maximum processing time is of the order of 250 msec.

☞ A minimum response time can be set on the controller under the menu item “Interface”. This setting is used as the minimum value before a response is transmitted (0—500 msec). If a small value is selected there is a possibility that the response time is longer than the setting (the internal processing time is longer); the controller then responds immediately after internal processing has been completed. A time setting of 0 msec means that the controller responds at the maximum possible speed.

The minimum response time setting is required by the master in the case of the RS485 interface in order to switch the interface driver from transmitting to receiving. In the case of the RS422 interface this parameter is not required.

$t_2$ This is the time required by the controller to switch from transmitting back to receiving. This time has to be kept by the master before making a fresh data request. It must always be kept, even when the new data request is addressed to a different instrument.

$t_2$ = 1 msec for RS422 interface
$t_2$ = 10 msec for RS485 interface

#### 2.4.2 Communication during the internal processing time of the slave

The master must not make any data requests during the internal processing time of the slave. Any data requests during this period are ignored by the slave.

#### 2.4.3 Communication during the response time of the slave

The master must not make any data requests during the response time of the slave. Any data requests during this period cause all data currently on the bus to become invalid.

## 2.5 Arrangement of the data blocks

All data blocks have the same structure:

**Data structure**

| slave address | function code | data field | checksum CRC16 |
|---|---|---|---|
| 1 byte | 1 byte | x byte(s) | 2 bytes |

Each data block consists of four fields:

**Slave address** instrument address of a particular slave

**Function code** function selection (read, write, bit, word)

**Data field** contains the information:

-bit address (word address)

-bit number (word number)

-bit value (word value)

**Checksum** recognition of transmission errors

## 2.6 Error treatment

**Error codes**

There are five error codes:

1 invalid function

2 invalid parameter address

3 parameter value outside range of values

4 slave not ready

8 write access to parameter denied

**Response in case of error**

| slave address | function XX OR 80h | error code | checksum CRC16 |
|---|---|---|---|
| 1 byte | 1 byte | 1 byte | 2 bytes |

The function code is linked by OR with 0x80, i.e. the MSB (most significant bit) is set to 1.

**Example**

Data request:

| 01 | 02 | 00 | 00 | 00 | 00 | CRC16 |
|---|---|---|---|---|---|---|

Response:

| 01 | 82 | 01 | CRC16 |
|---|---|---|---|

**Special cases**

The slave does not respond to the following errors:

- the checksum (CRC16) is incorrect
- the instruction of the master is incomplete or over-defined
- the number of the words or bits to be read is zero

Error code 4 (slave not ready) is not implemented in the controller since this always responds within 250 msec to a valid data request.

## 2.7 Distinction MODbus/Jbus

The MODbus protocol is compatible with the Jbus protocol. The structure of the data blocks is identical.

The difference between MODbus and Jbus is that the absolute addresses of the data are different. The addresses of MODbus are shifted by one against those of Jbus.

| Absolute address | Jbus address | MODbus address |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 1 |
| 3 | 3 | 2 |
| ... | ... | ... |

## 2.8 Checksum (CRC16)

The checksum (CRC16) serves to recognise transmission errors. If an error is identified during processing, the appropriate instrument does not respond.

**Calculation scheme**

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

☞ The low byte of the checksum is transmitted first.

**Example 1** Data request: reading two words from address 1 (CRC16 = 0x0E97)

| 14 | 03 | 00 | 01 | 00 | 02 | 97 | 0E |
|---|---|---|---|---|---|---|---|

Response: (CRC16 = 0x953E)

| 14 | 03 | 04 | 03 | E8 | 01 | F4 | 3E | 95 |
|---|---|---|---|---|---|---|---|---|
| | | | Word 1 | | Word 2 | | | |

**Example 2** Instruction: Set bit on address 24 (CRC16 = 0xF80E)

| 14 | 05 | 00 | 18 | FF | 00 | 0E | F8 |
|---|---|---|---|---|---|---|---|

Response (as instruction):

| 14 | 05 | 00 | 18 | FF | 00 | 0E | F8 |
|---|---|---|---|---|---|---|---|

## 2.9 Interface

| C2 | Protocol type | | SETUP |
|---|---|---|---|
| **Interface** | | | MODBUS |
| | | | JBUS |
| | Data format | Parity | no parity |
| | | | odd parity |
| **Explanation** | | | even parity |
| * = input a value | | | zero parity |
| ■ = factory setting | | Stop bit | 1 stop bit |
| | | | 2 stop bits |
| | | Baud rate | 9600 baud |
| | | | 4800 baud |
| | | | 2400 baud |
| | | | 1200 baud |
| | | | 600 baud |
| | | | 300 baud |
| | | | 150 baud |
| | | | 19200 baud |
| | | | 38400 baud |
| | | | 125000 baud |
| | | | 187500 baud |
| | Instrument address | | Value *_____ |
| | | | value range: 0 — 255 |
| | | | factory-set: 0 |
| | Minimum response time | | Value *_____ |
| | | | value range: 0 — 500 msec |
| | | | factory-set: 0 msec |

# 3 Functions

The following functions are available to the controller:

| Function number | Function | |
|---|---|---|
| 0x01/0x02 | Reading n bits | (max. 256 bits) |
| 0x03/0x04 | Reading n words | (max. 80 words) |
| 0x05 | Writing one bit | |
| 0x06 | Writing one word | |
| 0x0F | Writing n bits | (max. 256 bits) |
| 0x10 | Writing n words | (max. 80 words) |

There are no separate areas for bit and word for the system variables.The bit and word areas overlap and can be read and written both as bit area and as word area.

The bit address is calculated as follows:

**bit address = word address * 16 + bit number**

## 3.1 Reading n bits

This function reads n bits 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 values | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | x byte(s) | 2 bytes |

**Example**

Reading the status of the 5 logic inputs (process data),
⇨ Section 6.1.2

Bit address = (base address + process data address) * 16
= (0x0034 + 0x0036) * 0x10 = 0x06A0

**Data request**

| 0A | 01 | 06 | A0 | 00 | 05 | CRC16 |
|---|---|---|---|---|---|---|

**Response**

| 0A | 01 | 01 | 1F | CRC16 |
|---|---|---|---|---|

☞ In every case at least 8 bits (1 byte) are read irrespective of the number of bits to be read, since the response is made in bytes.

In the example above this means that the bits 0x06A0 — 0x06A7 are being read.

| 0x06A7 | 0x06A6 | 0x06A5 | 0x06A4 | 0x06A3 | 0x06A2 | 0x06A1 | 0x06A0 |
|---|---|---|---|---|---|---|---|

8 bits = 1 byte

For all non-relevant bits (0x06A5 — 0x06A7) the response is the value 0.

## 3.2 Reading n words

This function reads n words from a defined address.

**Data request**

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

Reading the four signal inputs

⇨ Section 6.2.8

Word address= base address + signal address
= 0x0034 + 0x0003 = 0x0037

Data request:

| 14 | 03 | 00 | 37 | 00 | 08 | CRC16 |
|---|---|---|---|---|---|---|

Response:

| 14 | 03 | 10 | 1999 | 4348 | 4CCC | 4348 | 2666 | 4396 | F333 | 43CA | CRC16 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Signal 1 200.1 | | Signal 2 200.3 | | Signal 3 300.3 | | Signal 4 405.9 | | |

## 3.3 Writing one bit

In the “writing bit” function the data blocks for instruction and response are identical.

**Instruction**

| Slave address | Function 0x05 | Bit address | Bit value XX 00 | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Response**

| Slave address | Function 0x05 | Bit address | Bit value XX 00 | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Example**

Set status bit 0 of data block parameter set 1 on controller 1
⇨ Section 6.3.3

Bit address = (base address + address status of data structure) * 16 + bit number
= (0x076E + 0x0) * 0x10 + 0x0
= 0x76E0

Instruction:

| 14 | 05 | 76 | E0 | FF | 00 | CRC16 |
|---|---|---|---|---|---|---|

Response (as instruction):

| 14 | 05 | 76 | E0 | FF | 00 | CRC16 |
|---|---|---|---|---|---|---|

## 3.4 Writing one word

In the “writing 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**

Write output 1: signal on OutOfRange (= 100 = 0x0064)
⇨ Section 6.2.9

Word address= base address + address output 1: signal on OutOfRange
= 0x0161 + 0x0007 = 0x0168

Instruction:

| | | | | | | |
|---|---|---|---|---|---|---|
| 14 | 06 | 01 | 68 | 00 | 64 | CRC16 |

Response (as instruction):

| | | | | | | |
|---|---|---|---|---|---|---|
| 14 | 06 | 01 | 68 | 00 | 64 | CRC16 |

## 3.5 Writing n bits

**Instruction**

| Slave address | Function 0x0F | Address first bit | Number of bits | Number of bytes | Bit value(s) | Checksum CRC16 |
|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | x byte(s) | 2 bytes |

**Response**

| Slave address | Function 0x0F | Address first bit | Number of bits | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

**Example**

Set the status bit 0 and bit 1 of the data block parameter set 1 on controller 1

status bit 0 = 1, status bit 1 = 0
⇨ Section 6.3.4

Bit address = (base address + address status of the data structure) * 16 + bit number
= (0x076E + 0x0) * 0x10 + 0x0 = 0x76E0

Instruction:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 14 | 0F | 76 | E0 | 00 | 02 | 01 | 01 | CRC16 |

Response:

| | | | | | | |
|---|---|---|---|---|---|---|
| 14 | 0F | 76 | E0 | 00 | 02 | CRC16 |

## 3.6 Writing n words

**Instruction**

| Slave address | Function 0x10 | Address first word | Number of words | Number of bytes | Word value(s) | Checksum CRC16 |
|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | x byte(s) | 2 bytes |

**Response**

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

**Example**

Write analogue input 2: process value correction offset = 0.66
(2 words: 3F28 F5C2)
⇨ Section 6.2.8

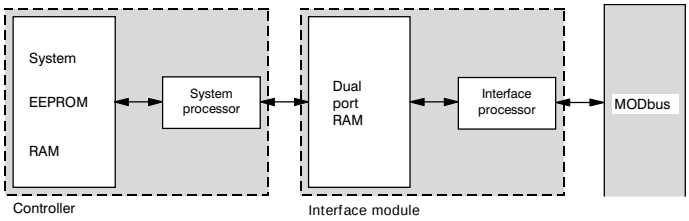
Word address= base address + analogue input 2: process value correction offset
= 0x00E8 + 0x0027 = 0x010F

Instruction:

| 14 | 10 | 01 | 0F | 00 | 02 | 04 | F5 | C2 | 3F | 28 | CRC16 |
|---|---|---|---|---|---|---|---|---|---|---|---|

Response:

| 14 | 10 | 01 | 0F | 00 | 02 | CRC16 |
|---|---|---|---|---|---|---|




For data transmission to the MODbus the process values are placed by the system processor in a dual port RAM. Not all the system variables present in the controller are updated cyclically in the dual port RAM. The dual port RAM is divided into two areas:

**System variables**

These variables can be read and written directly by the MODbus driver (cyclic data).

These data are updated cyclically in the dual port RAM (within the sampling time).

**Data after data request**

This area is not updated cyclically by the system processor (non-cyclic data).

Variables in this data area must be requested by the MODbus driver.

They are available only after processing by the system processor.

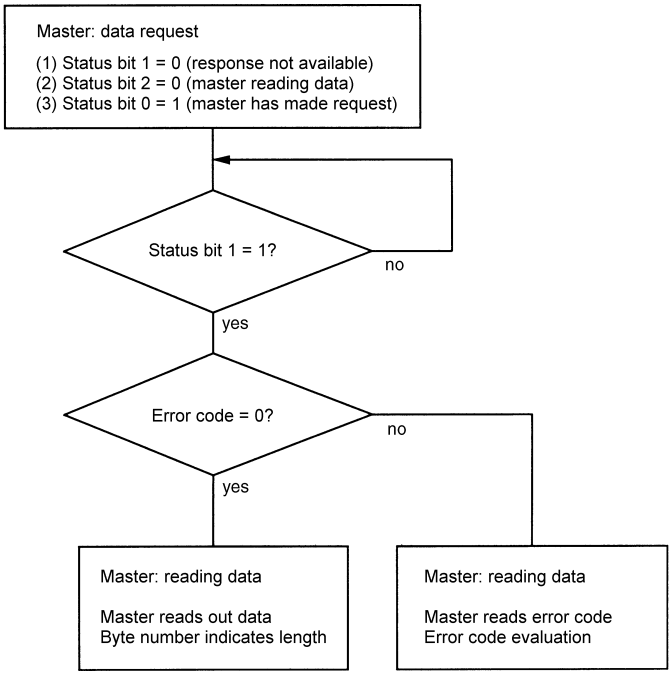
☞ The dual port RAM is erased after a reset. This is followed by the system processor updating the cyclic data. The area of non-cyclic data has to be updated by the user.

☞ The length information for char data types as given below always covers the character chain length including the chain end character /0.

☞ Every alteration of a process value which is stored in EEPROM results in updating of the data in the EEPROM. Please note that the EEPROM can be re-written about 10 000 times.

# 4 Data flow

## 4.1 Receive data from the controller



**Example**

Reading the fuzzy intensity parameter set 1 (= 30 decimal) of controller 1
⇨ Section 5.6

**Step 1:** The data structure parameter set 1 of controller 1 is requested.

Set status bit 0 = 1, status bit 1 = 0, and status bit 2 = 0

MODbus instruction: write 1 word

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 06 | 07 | 6E | 00 | 01 | CRC16 |

Response:

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 06 | 07 | 6E | 00 | 01 | CRC16 |

**Step 2:** Cyclic polling whether the corresponding data structure is available.

Status bit 1 read

MODbus instruction: read 1 word

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 03 | 07 | 6E | 00 | 01 | CRC16 |

Response:

| | | | | | |
|---|---|---|---|---|---|
| 01 | 03 | 02 | 00 | 00 | CRC16 |

Status bit 1 = 0 (data structure is not yet available)

| | | | | | |
|---|---|---|---|---|---|
| 01 | 03 | 02 | 00 | 02 | CRC16 |

Status bit 1 = 1 (data structure is available)

**Step 3:** Read error code of the structure requested

MODbus instruction: read 1 word

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 03 | 07 | 6F | 00 | 01 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0000 | CRC16 |

**Step 4:** Read out fuzzy intensity

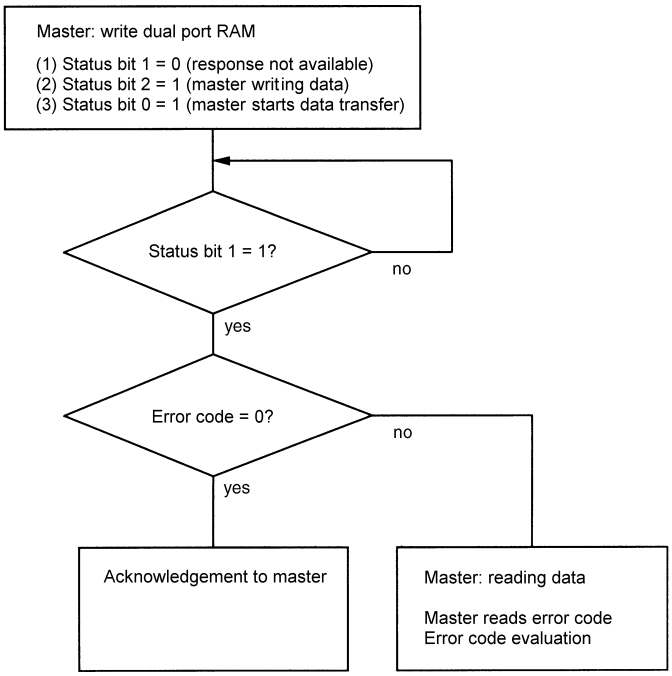
MODbus instruction: read fuzzy intensity (2 words)

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 03 | 07 | 95 | 00 | 02 | CRC16 |

Response:

| | | | | | |
|---|---|---|---|---|---|
| 01 | 03 | 04 | 0000 | 41F0 | CRC16 |
| | | | = 30 decimal | | |

## 4.2 Transmit data to the controller




The non-cyclic data of the controller are requested by the interface module through the setup commands.

**Example**

Writing setpoint 2 of controller 1

**Step 1:** The data structure setpoints is requested.

Set status: bit 0=1, bit 1 = 0, bit 2 = 0

MODbus instruction: write 1 word

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 09F6 | 0001 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 09F6 | 0001 | CRC16 |

**Step 2:** Cyclic polling whether the appropriate data structure is available.

Status bit 1 read

MODbus instruction: read 1 word

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 09F6 | 0001 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0000 | CRC16 |

Status bit 1 = 0 (data structure is not yet available)

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0002 | CRC16 |

Status bit 1 = 1 (data structure is available)

**Step 3:** Read error code of the structure requested

MODbus instruction: read 1 word

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 09F7 | 0001 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0000 | CRC16 |

**Step 4:** Write setpoint 2 of controller 1
(Setpoint 20.32 is IEEE format 41 A2 8F 5C)

MODbus instruction: write 2 words

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 01 | 10 | 09FA | 0002 | 04 | 8F5C | 41A2 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 09FA | 0002 | CRC16 |

**Step 5:** The data structure setpoints is transmitted

Set status: bit 0 = 1, bit 1 = 0, bit 2 = 1

MODbus instruction: write 1 word

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 09F6 | 0005 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 06 | 09F6 | 0005 | CRC16 |

**Step 6:** Cyclic polling whether the corresponding data structure has been transmitted.

Status bit 1 read

MODbus instruction: read 1 word

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 09F6 | 0001 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0000 | CRC16 |

Status bit 1 = 0 (data structure not yet transmitted)

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0002 | CRC16 |

Status bit 1 = 1 (data structure has been transmitted)

**Step 7:** Read error code of transmitted structure

MODbus instruction: read 1 word

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 09F7 | 0001 | CRC16 |

Response:

| | | | | |
|---|---|---|---|---|
| 01 | 03 | 02 | 0000 | CRC16 |

# 5 Error messages

The error numbers can be found under error code in the data blocks of the non-cyclic data.

| Error number | Error |
|---|---|
| 0x0000 | No error |
| **Profile program store** | |
| 0x0001 | Profile program can not be created |
| 0x0002 | Profile program is not available |
| 0x0007 | Profile program can not be copied |
| 0x0009 | Profile program checksum error |
| 0x0011 | Profile program store end |
| 0x0012 | Segment is not available |
| **Setup command processing** | |
| 0x0014 | Command busy flag not reset by the master |
| 0x0015 | Illegal command |
| 0x0016 | Error in data acceptance |
| 0x0017 | There are no cyclic data |
| 0x0018 | Illegal structure length |
| 0x0019 | Illegal header |
| 0x001C | Writing error into the serial EEPROM (Calib) |
| **Key and programming inhibit** | |
| 0x001A | Keys are inhibited |
| 0x001B | Programming is inhibited |

**Error messages for invalid values**

For setpoints, process values, and values calculated from them the convention applies that the error number is shown in the value itself.

| Error number | Error |
|---|---|
| 0x4843 5000 | Underrange |
| 0x4843 5040 | Overrange |
| 0x4843 5080 | Division by zero |
| 0x4843 50C8 | No valid input values |
| 0x4843 5100 | Value not programmed |
| 0x4843 5140 | Hardware range |
| 0x4843 5340 | Maths/Logics module |

# 6 Address tables

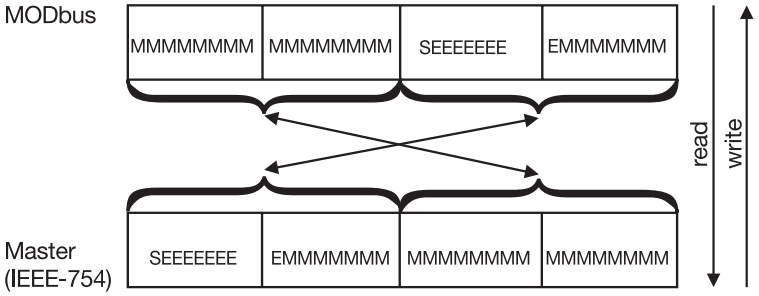
All process values (variables) together with their addresses, the data type and the access mode are described below.

References are as follows:

| | |
|---|---|
| R/O | access reading only |
| R/W | access reading and writing |
| char | ASCII character (8 bits) |
| byte | byte (8 bits) |
| int | integer (16 bits) |
| char xx | character chain of length xx;<br>xx = length including chain and character /0 |
| bit x | bit No. x |
| float | floating value (4 bytes) |

The explanations below apply on condition that the master operates in the IEEE-754 format. Before transmitting a value the byte sequence has to be changed so that it corresponds to the presentation for MODbus (see diagram).

M - 23 bit normalised mantissa
E - exponent (complement base 2)
S - sign bit; 1 = negative; 0 = positive




Example:
Transmitting the decimal value 550:
MODbus: 0x80, 0x00, 0x44, 0x09

| | |
|---|---|
| long | long integer (4 byte) |

The process values are divided into logical areas.

The absolute MODbus address is given by the base address of the appropriate area and the address offset.

In the address tables below, bit 0 is always the lowest value bit.

# 6 Address tables

☞ Memory contents which apply only to JUMO DICON 1001 are shown in *italics*.
Memory contents which apply only to JUMO DICON 1000 are shown **bold**.

**Status bits for non-cyclic data**

The following address tables (non-cyclic data) are read or written only following data request by the MODbus driver. The status of these address tables (non-cyclic data) is indicated in the status word. Status word and error code are always located in the data blocks of the non-cyclic data.

| Status | | | Interpretation for master |
|---|---|---|---|
| **bit 0** | **bit 1** | **bit 2** | |
| 0 | 0 | X | Master has made no data request |
| 1 | 0 | X | Master has made a data request to the instrument, the data request is being processed |
| 0 | 1 | X | Processing is completed, the response is ready for the master in the buffer |
| 1 | 1 | X | Invalid |
| | | 0 | Data transmission from instrument to dual port RAM |
| | | 1 | Data transmission from dual port RAM to instrument |
| Bit 3 to bit 15 not used | | | |

## 6.1 Cyclic data

### 6.1.1 Instrument data

Base address: 0x0000

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/O | Instrument group= 6 |
| 0x0001 | int | R/O | Instrument type = 0 |
| 0x0002 | char10 | R/O | Instrument name |
| 0x0007 | char12 | R/O | Software version 073.xx.xx / 84.xx.xx |
| 0x000D | char14 | R/O | VdN number |
| 0x0014 | char10 | R/O | Serial number |
| 0x0019 | char16 | R/O | Date/time last configuration alteration |
| 0x0021 | char16 | R/O | Date/time last parameter alteration |
| 0x0029 | char12 | R/O | Reserved |

Instrument data can only be read.

### 6.1.2 Process data

Base address: 0x0034

Cyclic process data can only be read.

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/O | System error 0=no system error |
| 0x0001 | int | R/O | Run time error 0=no run time error |
| 0x0002 | int | R/O | Control flags |
| | bit 0 | R/O | Controller 1: auto-tuning 0=OFF / 1=ON |
| | bit 1 | R/O | Controller 2: auto-tuning 0=OFF / 1=ON |
| | **bit 2** | **R/O** | **Controller 1: manual operation 0=OFF / 1=ON** |
| | **bit 3** | **R/O** | **Controller 2: manual operation 0=OFF / 1=ON** |
| | **bit 4** | **R/O** | **Controller 1: ramp off=1** |
| | **bit 5** | **R/O** | **Controller 2: ramp off=1** |
| | **bit 6** | **R/O** | **Controller 1: ramp stop 0=inactive / 1=active** |
| | **bit 7** | **R/O** | **Controller 2: ramp stop 0=inactive / 1=active** |
| | bit 8 | R/O | Keys inhibited=1 |
| | bit 9 | R/O | Parameter and configuration levels inhibited=1 |
| | bit 10 | R/O | Test and calibration mode=1 |
| | *bit 11* | *R/O* | *Profile program editor inhibit* |
| | bit 12 — 15 | R/O | Not used |
| | | | |
| | | | |
| 0x0003 | float | R/O | Signal input 1 |
| 0x0005 | float | R/O | Signal input 2 |
| 0x0007 | float | R/O | Signal input 3 |
| 0x0009 | float | R/O | Signal input 4 |
| 0x000B | float | R/O | Analogue output 1 |
| 0x000D | float | R/O | Not used |
| 0x000F | float | R/O | Analogue output 3 |
| 0x0011 | float | R/O | Analogue output 4 |
| 0x0013 | float | R/O | Controller 1: process value |
| 0x0015 | float | R/O | Controller 1: ramp setpoint |
| 0x0017 | float | R/O | Controller 1: setpoint |
| 0x0019 | float | R/O | Controller 1: deviation |
| 0x001B | float | R/O | Controller 1: output indication |
| 0x001D | float | R/O | Controller 1: heating output |
| 0x001F | float | R/O | Controller 1: cooling output |
| 0x0021 | int | R/O | Controller 1: relay status heating |
| 0x0022 | int | R/O | Controller 1: relay status cooling |
| 0x0023 | int | R/O | Controller 1: parameter set |
| 0x0024 | float | R/O | Controller 2: process value |
| 0x0026 | float | R/O | Controller 2: ramp setpoint |
| 0x0028 | float | R/O | Controller 2: setpoint |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x002A | float | R/O | Controller 2: deviation |
| 0x002C | float | R/O | Controller 2: output indication |
| 0x002E | float | R/O | Controller 2: heating output |
| 0x0030 | float | R/O | Controller 2: cooling output |
| 0x0032 | int | R/O | Controller 2: relay status heating |
| 0x0033 | int | R/O | Controller 2: relay status cooling |
| 0x0034 | int | R/O | Controller 2: parameter set |
| 0x0035 | int | R/O | Relay status 0=inactive / 1=active |
| | bit 0 | R/O | Output 1 |
| | bit 1 | R/O | Output 3 |
| | bit 2 | R/O | Output 4 |
| | bit 3 | R/O | Output 2 |
| | bit 4—7 | R/O | Not used |
| | bit 8 | R/O | Ext. relay module output 1 |
| | bit 9 | R/O | Ext. relay module output 2 |
| | bit 10 | R/O | Ext. relay module output 3 |
| | bit 11 | R/O | Ext. relay module output 4 |
| | bit 12 | R/O | Ext. relay module output 5 |
| | bit 13 | R/O | Ext. relay module output 6 |
| | bit 14 | R/O | Ext. relay module output 7 |
| | bit 15 | R/O | Ext. relay module output 8 |
| 0x0036 | int | R/O | Status of logic inputs 0=open/1=closed |
| | bit 0 | R/O | Logic input 1 |
| | bit 1 | R/O | Logic input 2 |
| | bit 2 | R/O | Logic input 3 |
| | bit 3 | R/O | Logic input 4 |
| | bit 4 | R/O | Logic input 5 |
| | bit 5—15 | R/O | Not used |
| 0x0037 | int | R/O | Status of limit comparators 0=inactive/1=active |
| | bit 0 | R/O | Limit comparator 1 |
| | bit 7 | R/O | Limit comparator 8 |
| | bit 8—15 | R/O | Not used |
| 0x0038 | int | R/O | Status of logic outputs 0=inactive/1=active |
| | bit 0 | R/O | Logic output 1 |
| | bit 1 | R/O | Logic output 2 |
| | bit 2—15 | R/O | Not used |
| *0x0039* | *int* | *R/O* | *Operating modes* |
| | *bit 2* | *R/O* | *Stop = 1* |
| | *bit 3* | *R/O* | *Basic status = 1* |
| | *bit 4* | *R/O* | *Manual / auto-manual = 1* |
| | *bit 11* | *R/O* | *Profile program delay=1* |
| | *bit 12* | *R/O* | *Profile end=1* |
| | *bit 13* | *R/O* | *Automatic=1* |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| | bit 0, 1, 5 —10, 14, 15 | R/O | Not used |
| 0x003A | float | R/O | Math 1 |
| 0x003C | float | R/O | Math 2 |
| *0x003E* | *int* | *R/O* | *Profile program number* |
| *0x003F* | *int* | *R/O* | *Segment number profile 0* |
| *0x0040* | *int* | *R/O* | *Segment number profile 1* |
| *0x0041* | *int* | *R/O* | *Last segment profile 0* |
| *0x0042* | *int* | *R/O* | *Last segment profile 1* |
| *0x0043* | *int* | *R/O* | *Tolerance signal* |
| *0x0044* | *float* | *R/O* | *Setpoint profile 0* |
| *0x0046* | *float* | *R/O* | *Setpoint profile 1* |
| *0x0048* | *int* | *R/O* | *Status of operating contacts 0 = inactive / 1 = active* |
| | *bit 0* | *R/O* | *Operating contact 1* |
| | *bit 7* | *R/O* | *Operating contact 8* |
| *0x0049* | *int* | *R/O* | *Parameter set number profile 0* |
| *0x004A* | *int* | *R/O* | *Parameter set number profile 1* |
| *0x004B* | *int* | *R/O* | *Controller editing* |
| *0x004C* | *int* | *R/O* | *Limit comparator editing* |
| *0x004D* | *long* | *R/O* | *Profile program run time* |
| *0x004F* | *long* | *R/O* | *Residual profile program run time* |
| *0x0051* | *long* | *R/O* | *Program time profile 0* |
| *0x0053* | *long* | *R/O* | *Program time profile 1* |
| *0x0055* | *long* | *R/O* | *Segment run time profile 0* |
| *0x0057* | *long* | *R/O* | *Segment run time profile 1* |
| *0x0059* | *long* | *R/O* | *Residual segment time profile 0* |
| *0x005B* | *long* | *R/O* | *Residual segment time profile 1* |
| *0x005D* | *long* | *R/O* | *Total segment time profile 0* |
| *0x005F* | *long* | *R/O* | *Total segment time profile 1* |
| *0x007F* | *int* | *R/O* | *Number of free segments* |

## 6.2 Non-cyclic data

### 6.2.1 Commands

Base address: 0x0A36

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Controller 1: start auto-tuning = 1 |
| 0x0003 | int | R/W | Controller 2: start auto-tuning = 1 |
| 0x0004 | int | R/W | Controller 1: abort auto-tuning = 1 |
| 0x0005 | int | R/W | Controller 2: abort auto-tuning = 1 |
| **0x0006** | **int** | **R/W** | **Controller 1: switch to manual = 1** |
| **0x0007** | **int** | **R/W** | **Controller 2: switch to manual = 1** |
| **0x0008** | **int** | **R/W** | **Controller 1: switch to auto = 1** |
| **0x0009** | **int** | **R/W** | **Controller 2: switch to auto = 1** |
| **0x000A** | **int** | **R/W** | **Controller 1: inhibit manual = 1** |
| **0x000B** | **int** | **R/W** | **Controller 2: inhibit manual = 1** |
| **0x000C** | **int** | **R/W** | **Controller 1: stop ramp function = 1** |
| **0x000D** | **int** | **R/W** | **Controller 2: stop ramp function = 1** |
| **0x000E** | **int** | **R/W** | **Controller 1: switch off ramp function = 1** |
| **0x000F** | **int** | **R/W** | **Controller 2: switch off ramp function = 1** |
| 0x0010 | int | R/W | Key inhibit = 1 |
| 0x0011 | int | R/W | Inhibit parameter/configuration levels |
| 0x0012 | long | R/W | Text display texts 1—23 |
| | bit 0 — 4 | | Logic input 1—5 |
| | bit 8, 9 | | Logic 1 and 2 |
| | bit 16 — 23 | | Limit comparator 1— 8 |
| | bit 24 — 31 | | Operating contact 1—8 |
| 0x0014 | int | R/W | Switch off display |
| *0x0015* | *int* | *R/W* | *Inhibit profile program* |
| *0x0016* | *int* | *R/W* | *Profile program status* |
| | *bit 2* | | *Profile program hold* |
| | *bit 3* | | *Profile program stop* |
| | *bit 4* | | *Operating mode manual* |
| | *bit 5* | | *Profile program start* |
| | *bit 6* | | *Segment change (fast forward)* |
| | *bit 13* | | *Automatic* |
| *0x0017* | *int* | *R/W* | *Setpoint number controller 1* |
| *0x0018* | *int* | *R/W* | *Setpoint number controller 2* |
| *0x0019* | *int* | *R/W* | *Parameter number controller 1* |
| *0x001A* | *int* | *R/W* | *Parameter number controller 2* |
| *0x001B* | *int* | *R/W* | *Process value number controller 1* |
| *0x001C* | *int* | *R/W* | *Process value number controller 2* |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x001D* | *int* | *R/W* | *Profile program* |
| 0x001E | int | R/W | Reserved |
| 0x001F | long | R/W | Reserved |

### 6.2.2 Manual output

Controller 1: base address 0x0294

Controller 2: base address: 0x029C

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *float* | *R/W* | *Manual output* |

### 6.2.3 Setpoints

Base address: 0x09F6

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | float | R/W | Controller 1: setpoint 1 |
| 0x0004 | float | R/W | Controller 1: setpoint 2 |
| 0x0006 | float | R/W | Controller 1: setpoint 3 |
| 0x0008 | float | R/W | Controller 1: setpoint 4 |
| 0x000A | | | reserved |
| ... | | | reserved |
| 0x0010 | | | reserved |
| 0x0012 | float | R/W | Controller 2: setpoint 1 |
| 0x0014 | float | R/W | Controller 2: setpoint 2 |
| 0x0016 | float | R/W | Controller 2: setpoint 3 |
| 0x0018 | float | R/W | Controller 2: setpoint 4 |

**Frequent setpoint programming**

In order to avoid damaging the EEPROM (10 000 writing cycles max.) it is recommended to use the following addresses for frequent setpoint programming.

Since the data (setpoints) are stored in a volatile memory (RAM) they are lost after a supply failure.

(For address tables see next page!)

Base address: 0x0484

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | float | R/W | Setpoint controller 1[1] |
| 0x0004 | float | R/W | Setpoint controller 2[1] |

1. Function is de-activated by mains off or programming with w = 200003

### 6.2.4 Controller parameters

Base address controller 1: 0x076E  Base address controller 2: 0x07AF

Controller 1: parameter set 1—4 = parameter set number 0—3

Controller 2: parameter set 1—4 = parameter set number 4—7

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Parameter set number [0—7] |
| 0x0003 | int | R/W | Controller structure heating contact |
| 0x0004 | int | R/W | Controller structure cooling contact |
| 0x0005 | float | R/W | XP1- proportional band heating contact |
| 0x0007 | float | R/W | XP2- proportional band cooling contact |
| 0x0009 | float | R/W | TV1- derivative time heating contact |
| 0x000B | float | R/W | TV2- derivative time cooling contact |
| 0x000D | float | R/W | TN1- reset time heating contact |
| 0x000F | float | R/W | TN2- reset time cooling contact |
| 0x0011 | float | R/W | CY1- cycle time heating contact |
| 0x0013 | float | R/W | CY2- cycle time cooling contact |
| 0x0015 | float | R/W | XSH- contact spacing |
| 0x0017 | float | R/W | XD1- switching differential heating contact |
| 0x0019 | float | R/W | XD2- switching differential cooling contact |
| 0x001B | float | R/W | TT- stroke time |
| 0x001D | float | R/W | Y0- working point |
| 0x001F | float | R/W | Y1- output limit heating contact |
| 0x0021 | float | R/W | Y2- output limit cooling contact |
| 0x0023 | float | R/W | TK1- minimum relay ON time heating contact |
| 0x0025 | float | R/W | TK2- minimum relay ON time cooling contact |
| 0x0027 | float | R/W | FC1- fuzzy intensity |
| 0x0029 | float | R/W | FC2- fuzzy parameter adjustment |

### 6.2.5 Configuration controller 1

Base address: 0x00B4

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Controller mode |
| 0x0003 | float | R/W | Time constant control loop monitor |

### 6.2.6 Configuration controller 2

Base address: 0x00BD

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Controller mode |
| 0x0003 | float | R/W | Time constant control loop monitor |

### 6.2.7 Limit comparator configuration

Base address: 0x07F0

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Limit comparator 0=absolute / 1=relative |
| | bit 0 | R/W | Limit comparator 1 |
| | bit 7 | R/W | Limit comparator 8 |
| 0x0003 | int | R/W | Limit comparator action on OutOfRange |
| | bit 0 | R/W | Limit comparator 1 0=inactive / 1=active |
| | bit 7 | R/W | Limit comparator 8 0=inactive / 1=active |
| 0x0004 | int | R/W | Limit comparator 1: lk function lk1— lk8 |
| 0x0005 | float | R/W | Limit comparator 1: lk limit |
| 0x0007 | float | R/W | Limit comparator 1: lk differential |
| 0x0009 | int | R/W | Limit comparator 2: lk function lk1—lk8 |
| 0x000A | float | R/W | Limit comparator 2: lk limit |
| 0x000C | float | R/W | Limit comparator 2: lk differential |
| 0x000E | int | R/W | Limit comparator 3: lk function lk1— lk8 |
| 0x000F | float | R/W | Limit comparator 3: lk limit |
| 0x0011 | float | R/W | Limit comparator 3: lk differential |
| 0x0013 | int | R/W | Limit comparator 4: lk function lk1— lk8 |
| 0x0014 | float | R/W | Limit comparator 4: lk limit |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0016 | float | R/W | Limit comparator 4: lk differential |
| 0x0018 | int | R/W | Limit comparator 5: lk function lk1—lk8 |
| 0x0019 | float | R/W | Limit comparator 5: lk limit |
| 0x001B | float | R/W | Limit comparator 5: lk differential |
| 0x001D | int | R/W | Limit comparator 6: lk function lk1— lk8 |
| 0x001E | float | R/W | Limit comparator 6: lk limit |
| 0x0020 | float | R/W | Limit comparator 6: lk differential |
| 0x0022 | int | R/W | Limit comparator 7: lk function lk1—lk8 |
| 0x0023 | float | R/W | Limit comparator 7: lk limit |
| 0x0025 | float | R/W | Limit comparator 7: lk differential |
| 0x0027 | int | R/W | Limit comparator 8: lk function lk1—lk8 |
| 0x0028 | float | R/W | Limit comparator 8: lk limit |
| 0x002A | float | R/W | Limit comparator 8: lk differential |

### 6.2.8 Configuration of the analogue inputs

Base address: 0x00E8

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Unit: 0 = degree C / 1 = degree F |
| 0x0003 | int | R/W | Analogue input 1: transducer |
| 0x0004 | int | R/W | Analogue input 1: linearisation table |
| 0x0005 | int | R/W | Analogue input 1: input number of cold junction |
| 0x0006 | int | R/W | Analogue input 1: output number heater current relay |
| 0x0007 | float | R/W | Analogue input 1: constant ambient temperature |
| 0x0009 | float | R/W | Analogue input 1: display start |
| 0x000B | float | R/W | Analogue input 1: display end |
| 0x000D | float | R/W | Analogue input 1: range start |
| 0x000F | float | R/W | Analogue input 1: range end |
| 0x0011 | float | R/W | Analogue input 1: process value correction (offset) |
| 0x0013 | float | R/O | Analogue input 1: customized recalibration X0 |
| 0x0015 | float | R/O | Analogue input 1: customized recalibration X1 |
| 0x0017 | float | R/W | Analogue input 1: filter time constant |
| 0x0019 | int | R/W | Analogue input 2: transducer |
| 0x001A | int | R/W | Analogue input 2: linearisation table |
| 0x001B | int | R/W | Analogue input 2: input number of cold junction |
| 0x001C | int | R/W | Analogue input 2: output number heater current relay |
| 0x001D | float | R/W | Analogue input 2: constant ambient temperature |
| 0x001F | float | R/W | Analogue input 2: display start |
| 0x0021 | float | R/W | Analogue input 2: display end |
| 0x0023 | float | R/W | Analogue input 2: range start |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0025 | float | R/W | Analogue input 2: range end |
| 0x0027 | float | R/W | Analogue input 2: process value correction (offset) |
| 0x0029 | float | R/O | Analogue input 2: customized recalibration X0 |
| 0x002B | float | R/O | Analogue input 2: customized recalibration X1 |
| 0x002D | float | R/W | Analogue input 2: filter time constant |
| 0x002F | int | R/W | Analogue input 3: transducer |
| 0x0030 | int | R/W | Analogue input 3: linearisation table |
| 0x0031 | int | R/W | Analogue input 3: input number of cold junction |
| 0x0032 | int | R/W | Analogue input 3: output number heater current relay |
| 0x0033 | float | R/W | Analogue input 3: constant ambient temperature |
| 0x0035 | float | R/W | Analogue input 3: display start |
| 0x0037 | float | R/W | Analogue input 3: display end |
| 0x0039 | float | R/W | Analogue input 3: range start |
| 0x003B | float | R/W | Analogue input 3: range end |
| 0x003D | float | R/W | Analogue input 3: process value correction (offset) |
| 0x003F | float | R/O | Analogue input 3: customized recalibration X0 |
| 0x0041 | float | R/O | Analogue input 3: customized recalibration X1 |
| 0x0043 | float | R/W | Analogue input 3: filter time constant |
| 0x0045 | int | R/W | Analogue input 4: transducer |
| 0x0046 | int | R/W | Analogue input 4: linearisation table |
| 0x0047 | int | R/W | Analogue input 4: input number of cold junction |
| 0x0048 | int | R/W | Analogue input 4: output number heater current relay |
| 0x0049 | float | R/W | Analogue input 4: constant ambient temperature |
| 0x004B | float | R/W | Analogue input 4: display start |
| 0x004D | float | R/W | Analogue input 4: display end |
| 0x004F | float | R/W | Analogue input 4: range start |
| 0x0051 | float | R/W | Analogue input 4: range end |
| 0x0053 | float | R/W | Analogue input 4: process value correction (offset) |
| 0x0055 | float | R/O | Analogue input 4: customized recalibration X0 |
| 0x0057 | float | R/O | Analogue input 4: customized recalibration X1 |
| 0x0059 | float | R/W | Analogue input 4: filter time constant |

### 6.2.9 Configuration of the analogue outputs

Base address: 0x0161

| Address offset MOD-Bus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Analogue output 1: output type |
| 0x0003 | float | R/W | Analogue output 1: signal start |
| 0x0005 | float | R/W | Analogue output 1: signal end |
| 0x0007 | int | R/W | Output 1: signal on OutOfRange |
| 0x0008 | int | R/O | Not used |
| 0x0009 | float | R/O | Not used |
| 0x000B | float | R/O | Not used |
| 0x000D | int | R/W | Output 2: signal on OutOfRange |
| 0x000E | int | R/W | Analogue output 3: output type |
| 0x000F | float | R/W | Analogue output 3: signal start |
| 0x0011 | float | R/W | Analogue output 3: signal end |
| 0x0013 | int | R/W | Output 3: signal on OutOfRange |
| 0x0014 | int | R/W | Analogue output 4: output type |
| 0x0015 | float | R/W | Analogue output 4: signal start |
| 0x0017 | float | R/W | Analogue output 4: signal end |
| 0x0019 | int | R/W | Output 4: signal on OutOfRange |

### 6.2.10 Configuration of the maths and logics module

Base address: 0x0193

| Address offset MOD-Bus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Math function 1 |
| 0x0003 | float | R/W | Math function 1: range start |
| 0x0005 | float | R/W | Math function 1: range end |
| 0x0007 | int | R/W | Logic function 1 |
| 0x0008 | int | R/W | Math function 2 |
| 0x0009 | float | R/W | Math function 2: range start |
| 0x000B | float | R/W | Math function 2: range end |
| 0x000D | int | R/W | Logic function 2 |

### 6.2.11 Configuration of the special functions

Base address: 0x0830

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| **0x0002** | **int** | **R/W** | **Ramp controller 1: function 0=OFF / 1=ON** |
| **0x0003** | **int** | **R/W** | **Ramp controller 1: unit of ramp slope** |
| **0x0004** | **float** | **R/W** | **Ramp controller 1: ramp slope** |
| **0x0006** | **int** | **R/W** | **Ramp controller 2: function 0=OFF / 1=ON** |
| **0x0007** | **int** | **R/W** | **Ramp controller 2: unit of ramp slope** |
| **0x0008** | **float** | **R/W** | **Ramp controller 2: ramp slope** |
| 0x000A | float | R/W | Manual operation controller 1: manual output |
| 0x000C | int | R/W | Manual operation controller 1: inhibited = 0 |
| 0x000D | float | R/W | Manual operation controller 2: manual output |
| 0x000F | int | R/W | Manual operation controller 2: inhibited = 0 |
| 0x0010 | int | R/W | Auto-tuning controller 1: inhibited = 0 |
| 0x0011 | int | R/W | Auto-tuning controller 2: inhibited = 0 |
| 0x0012 | float | R/W | Setpoint limit controller 1: setpoint start |
| 0x0014 | float | R/W | Setpoint limit controller 1: setpoint end |
| 0x0016 | float | R/W | Setpoint limit controller 2: setpoint start |
| 0x0018 | float | R/W | Setpoint limit controller 2: setpoint end |
| 0x001A | int | R/W | Controller type:<br>0=single contrl. / 1=cascade contrl. / 2 = trim cascade contrl. |
| 0x001B | float | R/W | Cascade output conversion: start |
| 0x001D | float | R/W | Cascade output conversion: end |
| 0x001F | int | R/W | Display brightness |
| 0x0020 | int | R/W | Time for automatic channel scrolling |
| 0x0021 | int | R/W | Time-out |
| 0x0022 | int | R/W | Supply frequency 0=50 Hz / 1=60 Hz |
| 0x0023 | int | R/W | Logic input 1: function |
| 0x0024 | int | R/W | Logic input 2: function |
| 0x0025 | int | R/W | Logic input 3: function |
| 0x0026 | int | R/W | Logic input 4: function |
| 0x0027 | int | R/W | Logic input 5: function |
| 0x0028 | int | R/W | Limit comparator 1: function |
| 0x0029 | int | R/W | Limit comparator 2: function |
| 0x002A | int | R/W | Limit comparator 3: function |
| 0x002B | int | R/W | Limit comparator 4: function |
| 0x002C | int | R/W | Limit comparator 5: function |
| 0x002D | int | R/W | Limit comparator 6: function |
| 0x002E | int | R/W | Limit comparator 7: function |
| 0x002F | int | R/W | Limit comparator 8: function |
| 0x0030 | int | R/W | Logic output 1: function |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0031 | int | R/W | Logic output 2: function |
| *0x0032* | *int* | *R/W* | *Operating contact 1: function* |
| *0x0033* | *int* | *R/W* | *Operating contact 2: function* |
| *0x0034* | *int* | *R/W* | *Operating contact 3: function* |
| *0x0035* | *int* | *R/W* | *Operating contact 4: function* |
| *0x0036* | *int* | *R/W* | *Operating contact 5: function* |
| *0x0037* | *int* | *R/W* | *Operating contact 6: function* |
| *0x0038* | *int* | *R/W* | *Operating contact 7: function* |
| *0x0039* | *int* | *R/W* | *Operating contact 8: function* |
| *0x003A* | *int* | *R/W* | *Real time clock* |

### 6.2.12 Configuration of the universal interface

Base address: 0x01EE

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Instrument address |
| 0x0003 | int | R/W | Protocol |
| 0x0004 | int | R/W | Parity |
| 0x0005 | int | R/W | Stop bits |
| 0x0006 | int | R/W | Baud rate |
| 0x0007 | int | R/W | Minimum response time |

### 6.2.13 Configuration of the linking table

Base address: 0x0200

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Input controller 1: process value |
| 0x0003 | int | R/W | Input controller 1: external setpoint |
| 0x0004 | int | R/W | Input controller 1: external setpoint with correction |
| 0x0005 | int | R/W | Input controller 1: output retransmission |
| 0x0006 | int | R/W | Input controller 1: additive disturbance compensation |
| 0x0007 | int | R/W | Input controller 1: multiplying disturbance compensation |
| 0x0008 | int | R/W | Input controller 2: process value |
| 0x0009 | int | R/W | Input controller 2: external setpoint |
| 0x000A | int | R/W | Input controller 2: external setpoint with correction |
| 0x000B | int | R/W | Input controller 2: output retransmission |

# 6 Address tables

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x000C | int | R/W | Input controller 2: additive disturbance compensation |
| 0x000D | int | R/W | Input controller 2: multiplying disturbance compensation |
| 0x000E | int | R/W | Input Ik 1: actual value |
| 0x000F | int | R/W | Input Ik 1: setpoint |
| 0x0010 | int | R/W | Input Ik 2: actual value |
| 0x0011 | int | R/W | Input Ik 2: setpoint |
| 0x0012 | int | R/W | Input Ik 3: actual value |
| 0x0013 | int | R/W | Input Ik 3: setpoint |
| 0x0014 | int | R/W | Input Ik 4: actual value |
| 0x0015 | int | R/W | Input Ik 4: setpoint |
| 0x0016 | int | R/W | Input Ik 5: actual value |
| 0x0017 | int | R/W | Input Ik 5: setpoint |
| 0x0018 | int | R/W | Input Ik 6: actual value |
| 0x0019 | int | R/W | Input Ik 6: setpoint |
| 0x001A | int | R/W | Input Ik 7: actual value |
| 0x001B | int | R/W | Input Ik 7: setpoint |
| 0x001C | int | R/W | Input Ik 8: actual value |
| 0x001D | int | R/W | Input Ik 8: setpoint |
| 0x001E | int | R/W | Signal output 1 |
| 0x001F | int | R/W | Signal output 2 |
| 0x0020 | int | R/W | Signal output 3 |
| 0x0021 | int | R/W | Signal output 4 |
| 0x0022 | int | R/W | External relay 1: signal |
| 0x0023 | int | R/W | External relay 1: signal on OutOfRange |
| 0x0024 | int | R/W | External relay 2: signal |
| 0x0025 | int | R/W | External relay 2: signal on OutOfRange |
| 0x0026 | int | R/W | External relay 3: signal |
| 0x0027 | int | R/W | External relay 3: signal on OutOfRange |
| 0x0028 | int | R/W | External relay 4: signal |
| 0x0029 | int | R/W | External relay 4: signal on OutOfRange |
| 0x002A | int | R/W | External relay 5: signal |
| 0x002B | int | R/W | External relay 5: signal on OutOfRange |
| 0x002C | int | R/W | External relay 6: signal |
| 0x002D | int | R/W | External relay 6: signal on OutOfRange |
| 0x002E | int | R/W | External relay 7: signal |
| 0x002F | int | R/W | External relay 7: signal on OutOfRange |
| 0x0030 | int | R/W | External relay 8: signal |
| 0x0031 | int | R/W | External relay 8: signal on OutOfRange |
| 0x0032 | int | R/W | Math function 1: variable a |
| 0x0033 | int | R/W | Math function 1: variable b |
| 0x0034 | int | R/W | Math function 2: variable a |
| 0x0035 | int | R/W | Math function 2: variable b |
| 0x0036 | int | R/W | Controller 1: display 1 |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0037 | int | R/W | Controller 1: display 2 |
| 0x0038 | int | R/W | Controller 1: display 3 |
| 0x0039 | int | R/W | Controller 1: display 4 |
| 0x003A | int | R/W | Controller 1: display 1 decimal point |
| 0x003B | int | R/W | Controller 1: display 2 decimal point |
| 0x003C | int | R/W | Controller 1: display 3 decimal point |
| 0x003D | int | R/W | Controller 1: display 4 decimal point |
| 0x003E | int | RW | Controller 2: display 1 |
| 0x003F | int | R/W | Controller 2: display 2 |
| 0x0040 | int | R/W | Controller 2: display 3 |
| 0x0041 | int | R/W | Controller 2: display 4 |
| 0x0042 | int | R/W | Controller 2: display 1 decimal point |
| 0x0043 | int | R/W | Controller 2: display 2 decimal point |
| 0x0044 | int | R/W | Controller 2: display 3 decimal point |
| 0x0045 | int | R/W | Controller 2: display 4 decimal point |

### 6.2.14 Hardware codes

Base address: 0x0746

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/O | A/D converter module 2 |
| 0x0003 | int | R/O | Output 1 |
| 0x0004 | int | R/O | Output 2 |
| 0x0005 | int | R/O | Output 3 |
| 0x0006 | int | R/O | Output 4 |
| 0x0007 | int | R/O | Setup interface |
| 0x0008 | int | R/O | Universal interface |
| 0x0009 | int | R/O | External relay module |
| 0x000A | int | R/O | Solder link S201 |
| 0x000B | int | R/O | Solder link S202 |
| 0x000C | int | R/O | Solder link S203 |
| 0x000D | int | R/O | Data backup |

### 6.2.15 Software codes

Base address: 0x075C

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/O | Reserved |
| 0x0003 | int | R/O | Reserved |
| 0x0004 | int | R/O | Language |

### 6.2.16 Code numbers

Base address: 0x02A4

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | char 4 | R/O | Mastercode number |
| 0x0004 | int | R/O | reserved |
| 0x0005 | char 4 | R/O | Access code for parameter level |
| 0x0007 | int | R/O | reserved |
| 0x0008 | char 4 | R/O | Access code for configuration level 1 |
| 0x000A | int | R/O | reserved |
| 0x000B | char 4 | R/O | Access code for configuration level 2 |
| 0x000D | int | R/O | reserved |
| 0x000E | char 4 | R/O | Access code for service mode |
| 0x0010 | int | R/O | reserved |
| 0x0011 | char 4 | R/O | Access code for customized recalibration |
| 0x0013 | int | R/O | reserved |
| 0x0014 | char 4 | R/O | Access code for profile program editor |
| 0x0016 | int | R/O | reserved |
| 0x0017 | char 4 | R/O | Access code for erasing profile program |
| 0x0019 | int | R/O | reserved |

### 6.2.17 Formulae for maths module

Base address: 0x04C4

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | char 72 | R/W | Math 1 formula (ASCII characters) |
| 0x0026 | char 200 | R/W | reserved |
| 0x008A | char 72 | R/W | Logic 1 formula (ASCII characters) |
| 0x00AE | char 200 | R/W | reserved |
| 0x0112 | int | R/W | Formula time math/logic 1 |
| 0x0113 | char 72 | R/W | Math 2 formula (ASCII characters) |
| 0x0137 | char 200 | R/W | reserved |
| 0x019B | char 72 | R/W | Logic 2 formula (ASCII characters) |
| 0x01BF | char 200 | R/W | reserved |
| 0x0223 | int | R/W | Formula time math/logic 2 |

### 6.2.18 Customized linearisation table

Base address: 0x02C4

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/W | Number of linearisation table [0, 1] |
| 0x0003 | float | R/W | Range start [Ohm, mV] |
| 0x0005 | float | R/W | Range end [Ohm, mV] |
| 0x0007 | float | R/W | Step size [Ohm, mV] |
| 0x0009 | float 50 | R/W | 50 tabulation values |
| 0x006D | float 50 | R/W | Value pairs 50 values x |
| 0x00D1 | float 50 | R/W | Value pairs 50 values y |

### 6.2.19 User texts

Base address: 0x0880

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | char 18 | R/W | User text logic input 1 |
| 0x000B | char 18 | R/W | User text logic input 2 |
| 0x0014 | char 18 | R/W | User text logic input 3 |
| 0x001D | char 18 | R/W | User text logic input 4 |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0026 | char 18 | R/W | User text logic input 5 |
| 0x002F | char 18 | R/W | User text logic 1 |
| 0x0038 | char 18 | R/W | User text logic 2 |
| 0x0041 | char 18 | R/W | User text limit comparator 1 |
| 0x004A | char 18 | R/W | User text limit comparator 2 |
| 0x0053 | char 18 | R/W | User text limit comparator 3 |
| 0x005C | char 18 | R/W | User text limit comparator 4 |
| 0x0065 | char 18 | R/W | User text limit comparator 5 |
| 0x006E | char 18 | R/W | User text limit comparator 6 |
| 0x0077 | char 18 | R/W | User text limit comparator 7 |
| 0x0080 | char 18 | R/W | User text limit comparator 8 |
| 0x0089 | char 18 | R/W | User text operating contact 1 |
| 0x0092 | char 18 | R/W | User text operating contact 2 |
| 0x009B | char 18 | R/W | User text operating contact 3 |
| 0x00A4 | char 18 | R/W | User text operating contact 4 |
| 0x00AD | char 18 | R/W | User text operating contact 5 |
| 0x00B6 | char 18 | R/W | User text operating contact 6 |
| 0x00BF | char 18 | R/W | User text operating contact 7 |
| 0x00C8 | char 18 | R/W | User text operating contact 8 |

### 6.2.20 Configuration C-level measurement

Base address: 0x0444

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/O | Analogue input C level |
| 0x0003 | int | R/W | Analogue input temperature |
| 0x0004 | int | R/W | Analogue input CO content |
| 0x0005 | float | R/W | CO content |
| 0x0007 | float | R/W | Correction factor |
| 0x0009 | float | R/W | Furnace correction |
| 0x000B | int | R/W | Cycle time (minutes) |
| 0x000C | int | R/W | Flushing time (minutes) |
| 0x000D | int | R/W | Recovery time (minutes) |

### 6.2.21 Instrument options

Base address: 0x09D0

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0000 | int | R/W | Status of data structure |
| 0x0001 | int | R/W | Error code |
| 0x0002 | int | R/O | Reserved |
| 0x0003 | int | R/O | Options 0= inhibited / 1= de-inhibited |
| | bit 0 | | Maths |
| | bit 1 | | Reserved |
| | bit 2 | | Second controller or profile program controller |
| | bit 3 | | C-level measurement |
| 0x0004 | int | R/O | Reserved |

### 6.2.22 Configuration programmer

Base address: 0x00C6

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *int* | *R/W* | *Profile program after supply failure* |
| *0x0003* | *int* | *R/W* | *Condition for program profile start* |
| *0x0004* | *int* | *R/W* | *Condition for setpoint programming* |
| *0x0005* | *int* | *R/W* | *Time/slope profile programming* |
| *0x0006* | *int* | *R/W* | *Controller activation on profile program off* |
| *0x0007* | *int* | *R/W* | *Auto-manual when outside tolerance* |
| *0x0008* | *int* | *R/W* | *Function operation controller 1* |
| *0x0009* | *int* | *R/W* | *Function operation controller 2* |
| *0x000A* | *int* | *R/W* | *Function operation limit comparator 1* |
| *0x000B* | *int* | *R/W* | *Function operation limit comparator 2* |
| *0x000C* | *int* | *R/W* | *Function operation limit comparator 3* |
| *0x000D* | *int* | *R/W* | *Function operation limit comparator 4* |
| *0x000E* | *int* | *R/W* | *Function operation limit comparator 5* |
| *0x000F* | *int* | *R/W* | *Function operation limit comparator 6* |
| *0x0010* | *int* | *R/W* | *Function operation limit comparator 7* |
| *0x0011* | *int* | *R/W* | *Function operation limit comparator 8* |
| *0x0012* | *int* | *R/W* | *Process value deviation before/after supply failure* |
| *0x0013* | *int* | *R/W* | *Profile program end signal* |
| *0x0014* | *long* | *R/W* | *Reserved* |

### 6.2.23 System states

Base address: 0x0980

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| | | | *System status: basic status* |
| *0x0002* | *int 3* | *R/W* | *Reserved* |
| *0x0005* | *float* | *R/W* | *Setpoint profile program 1* |
| *0x0007* | *float* | *R/W* | *Setpoint profile program 2* |
| *0x0009* | *int* | *R/W* | *Operating contacts 0= inactivated / 1= activated* |
| | *bit 0—7* | | *Operating contact 1—8* |
| *0x000A* | *int* | *R/W* | *Parameter set number profile 1* |
| *0x000B* | *int* | *R/W* | *Parameter set number profile 2* |
| *0x000C* | *int* | *R/W* | *Controller editing 0=inactive / 1=active<br>bit 0: controller 1 bit 1: controller 2* |
| *0x000D* | *int* | *R/W* | *Limit comparator editing 0=inactived / 1=activated<br>bit 0: Lk 1 bit 8: Lk8* |
| | | | *System status OutOfRange* |
| *0x000E* | *int* | *R/W* | *Condition setpoint* |
| *0x000F* | *int* | *R/W* | *Condition operating contact* |
| *0x0010* | *int* | *R/W* | *Condition parameter set* |
| *0x0011* | *float* | *R/W* | *Setpoint profile 1* |
| *0x0013* | *float* | *R/W* | *Setpoint profile 2* |
| *0x0015* | *int* | *R/W* | *Operating contacts 0= inactivated / 1= activated* |
| | *bit 0—7* | | *Operating contact 1—8* |
| *0x0016* | *int* | *R/W* | *Parameter set number profile 1* |
| *0x0017* | *int* | *R/W* | *Parameter set number profile 2* |
| *0x0018* | *int* | *R/W* | *Controller editing 0= inactivated / 1= activated<br>bit 0: controller 1 bit 1: controller 2* |
| *0x0019* | *int* | *R/W* | *Limit comparator editing 0= inactivated / 1= activated<br>bit 0: lk 1 bit 8: Lk8* |
| | | | *System status manual operation* |
| *0x001A* | *int 3* | *R/W* | *Reserved* |
| *0x001D* | *float* | *R/W* | *Setpoint profile 1* |
| *0x001F* | *float* | *R/W* | *Setpoint profile 2* |
| *0x0021* | *int* | *R/W* | *Operating contacts 0 = inactivated / 1 = activated* |
| | *bit 0—7* | | *Operating contact 1—8* |
| *0x0022* | *int* | *R/W* | *Parameter set number profile 1* |
| *0x0023* | *int* | *R/W* | *Parameter set number profile 2* |
| *0x0024* | *int* | *R/W* | *Controller editing 0= inactivated / 1= activated<br>bit 0: controller 1 bit 1: controller 2* |
| *0x0025* | *int* | *R/W* | *Limit comparator editing 0 = inactivated / 1 = activated<br>bit 0: Lk 1 bit 8: Lk 8* |
| | | | *System status stop* |

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0026* | *int 3* | *R/W* | *Reserved* |
| *0x0029* | *float* | *R/W* | *Setpoint profile 1* |
| *0x002B* | *float* | *R/W* | *Setpoint profile 2* |
| *0x002D* | *int* | *R/W* | *Operating contacts 0 = inactivated / 1 = activated* |
| | *bit 0 — 7* | | *Operating contact 1 — 8* |
| *0x002E* | *int* | *R/W* | *Parameter set number profile 1* |
| *0x002F* | *int* | *R/W* | *Parameter set number profile 2* |
| *0x0030* | *int* | *R/W* | *Controller editing 0 = inactivated / 1 = activated bit 0: controller 1 bit 1: controller 2* |
| *0x0031* | *int* | *R/W* | *Limit comparator editing 0 = inactivated / 1 = activated bit 0: Lk 1 bit 8: Lk 8* |

### 6.2.24 Profile program name

Base address: 0x0284

Profile program number 0 — 24 → profile programs 0 — 24

Profile program number 25 → basic status

Profile program number 26 → stop

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *int* | *R/W* | *Profile program number* |
| *0x0003* | *char 18* | *R/W* | *Profile program name* |

### 6.2.25 System clock

Base address: 0x0272

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *int* | *R/W* | *Hours (0x0102=12 hours)* |
| *0x0003* | *int* | *R/W* | *Minutes (0x0304=34 minutes)* |
| *0x0004* | *int* | *R/W* | *Seconds (0x0509=59 seconds)* |
| *0x0005* | *int* | *R/W* | *Day (0x0205=25)* |
| *0x0006* | *int* | *R/W* | *Month (0x0100=10)* |
| *0x0007* | *int* | *R/W* | *Year (0x0906=96)* |
| *0x0008* | *int* | *R/W* | *Weekday (0x0000 — 0x0600=SU — SA)* |
| *0x0009* | *int* | *R/W* | *Reserved* |

### 6.2.26 Profile program start

Base address: 0x01B5

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *int* | *R/W* | *Mode 0 = no start / 1= start* |
| *0x0003* | *int* | *R/W* | *Profile program number* |
| *0x0004* | *long* | *R/W* | *Delay time (real time clock not activated) in seconds* |
| *0x0006* | *long* | *R/W* | *Start time in seconds (real time clock activated), -1 = immediate start* |
| *0x0008* | *int* | *R/W* | *Weekday 0 = Sunday, 6 = Saturday* |
| *0x0009* | *int* | *R/W* | *Segment number profile 1* |
| *0x000A* | *long* | *R/W* | *Residual segment time* |

### 6.2.27 Profile program stop

Base address: 0x01C9

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |

### 6.2.28 Profile program continue

Base address: 0x01D1

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |

### 6.2.29 Profile program fast forward

Base address: 0x01D9

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |

### 6.2.30 Profile program functions

Base address: 0x01E2

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *int* | *R/W* | *Function: 1= delete profile program store<br>2= delete profile program<br>3= copy profile program* |
| *0x0003* | *int* | *R/W* | *Source profile program number* |
| *0x0004* | *int* | *R/W* | *Target profile program number* |

### 6.2.31 Transmit profile program

Base address: 0x0A66

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *R/W* | *Status of data structure* |
| *0x0001* | *int* | *R/W* | *Error code* |
| *0x0002* | *int* | *R/W* | *Profile program number [0—24]* |
| *0x0003* | *int* | *R/W* | *Profile number [0, 1]* |
| *0x0004* | *int* | *R/W* | *Length of profile program (n*20)* |
| | | | ***Segment 00*** |
| *0x0005* | *float* | *R/W* | *Setpoint* |
| *0x0007* | *int* | *R/W* | *Operating contacts 0 = inactive / 1= active* |
| | *bit 0* | | *Operating contact 1* |
| | *bit 7* | | *Operating contact 8* |
| 0x0008 | *float* | *R/W* | *Tolerance minimum value* |
| 0x000A | *float* | *R/W* | *Tolerance maximum value* |
| 0x000C | *long* | *R/W* | *Segment time in seconds* |
| 0x000E | *int* | *R/W* | *Number of repeat cycles; -1= endless* |
| 0x000F | *int* | *R/W* | *Target segment for repeat cycles* |
| 0x0010 | *int* | *R/W* | *Parameter set number 0—3* |
| | | | ***Segment 01*** |
| 0x0011 | | | *as above* |
| ... | | | |
| 0x0001C | | | |
| | | | ***Segment 02*** |
| 0x001D | | | *as above* |
| ... | | | |
| 0x0028 | | | |
| | | | ... |

⇨ Section 7

### 6.2.32 Changeover manual operation

Base address: 0x0960

| Address offset MODbus | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| *0x0000* | *int* | *W* | *Status of data structure* |
| *0x0001* | *int* | *W* | *Error code* |
| *0x0002* | *int* | *W* | *Mode 0 =manual OFF / 1= manual ON* |
| *0x0003* | *float* | *W* | *Setpoint profile 1* |
| *0x0005* | *float* | *W* | *Setpoint profile 2* |
| *0x0007* | *int* | *W* | *Operating contacts 0 = inactivated / 1= activated* |
| | *bit 0* | | *Operating contact 1* |
| | *bit 7* | | *Operating contact 8* |
| *0x0008* | *int* | *W* | *Parameter set number profile 1* |
| *0x0009* | *int* | *W* | *Parameter set number profile 2* |
| *0x000A* | *int* | *W* | *Controller editing 0 = inactivated / 1= activated*<br>*bit 0: controller 1 bit 1: controller 2* |
| *0x000B* | *int* | *W* | *Limit comparator editing 0 = inactivated / 1= activated*<br>*bit 0: Lk 1 bit 8: Lk 8* |

# 7 Program example

Program example: program transmission JUMO DICON 1001

```
; instrument address OA
;
; erase profile program store
OA06 01E4 0001                          ;function erase profile program store
0A06 01E2 0005                          ; status data structure
0A04 01E2 0001
#DOWHILE 0A03020005
0A03 01E2 0001
#END
;
; erase profile program
OA06 01E40002                           ; function erase profile program
OA06 01E50005                           ; profile program number 5
0A06 01E20005                           , status data structure
0A03 01E20001
#DOWHILE 0A03020005
0A03 01E2 0001
#END
;
; read profile program
0A10 0A66 000408 0001000000050000       ; status data structure (profile program
                                        number 5)
                                        ; address 0A6A contains the number of
                                        segments *20d
0A03 0A66 0001
#DOWHILE =A03020005
0A03 01E2 0001
#END
0A03 0A6A 00030001                      ; read number of segments
0A03 0A6B 0036                          ; read in 3 segments
;

; write profile program
0A06 0A68 0005                          ;profile program number 5
0A06 0A69 0000                          :profile 0
;
```

```
; 3 segments
0A06 0A6A 003C                      ;segment number *20 = 3*20
;
; segment times ABS00..ABS02
0A10 0A72000204 00010000            ;segment time ABS00          1 sec

0A10 0A7E000204 003C0000            ;segment time ABS01         60 sec
0A10 0A8A000204 0E100000            ;segment time ABS02       3600 sec
;
;setpoints ABS00. .ABS02
0A10 0A6B 000204 00000000           ;setpoint ABS00    0
0A10 0A77 000204 00004270           ;setpoint ABS01   60
0A10 0A83 000204 00000000           ;setpoint ABS02    0
;
0A06 0A66 0005                      ,status data structure
0A03 0A66 0001
#DOWHILE 0A3020005
0A03 0A66 0001
#END
0A03 0A67 0001                      ;read out error code
;
```

JUMO

MEASUREMENT AND CONTROL

**M. K. JUCHHEIM GmbH & Co**

36035 Fulda
Germany
Phone (int. +49) 661 60 03-0
Fax (int. +49) 661 60 03-607
Telex 49701 juf d
email JUMO_de@e-mail.com