# JUMO di 308

Digital Indicator


## B 70.1550.2.0

Interface Description
Modbus

JUMO

12.07/00485372

# Contents

# Contents

## 1.1 Preface

This operating manual is addressed to the system manufacturer with adequate technical background and PC related knowledge.

Please read this operating manual prior to commissioning the instrument. Keep the manual in a place accessible to all users at all times. Your comments are appreciated and may assist us in improving this manual.

**Warranty**

All necessary settings are described in this operating manual. Should problems be encountered during commissioning, please refrain from carrying out any manipulations that are not described in the manual. Any such intervention will jeopardize your warranty rights. Please contact the nearest subsidiary or the head office.

**Electrostatic discharge**

When accessing the inner parts of the unit and returning plug-in modules, assemblies or components, please observe the regulations according to EN 61340-5-1 and EN 61340-5-2 „Protection of electrostatic sensitive devices“. Only use **ESD** packaging for transport.

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

**ESD**=**E**lectro **S**tatic **D**ischarge

## 1.2 Typographical conventions

### 1.2.1 Warning signs

**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!

**ESD**

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.

**Reference**

This symbol refers to **further information** in other manuals, chapters or sections.

**Footnote**

abc[1]

Footnotes are remarks that **refer to specific points** in the text. Footnotes consist of two parts:
A marker in the text and the foot note text itself.
The markers in the text are arranged as continuous superscript numbers.

### 1.2.3 Carry out action

| | | |
|---|---|---|
| **Instruction to act** | ✱ | This symbol indicates that an **action to be performed** is described. The individual steps are marked by this asterisk, e.g.:<br>✱ Start PLC software<br>✱ Click on hardware catalog |
| **Vital text** |  | This text contains important information, and it is vital that you read it before going any further. |
| **String of commands** | File ➔ save as | Small arrows between words indicate a **string of Commands** to be executed in succession. |

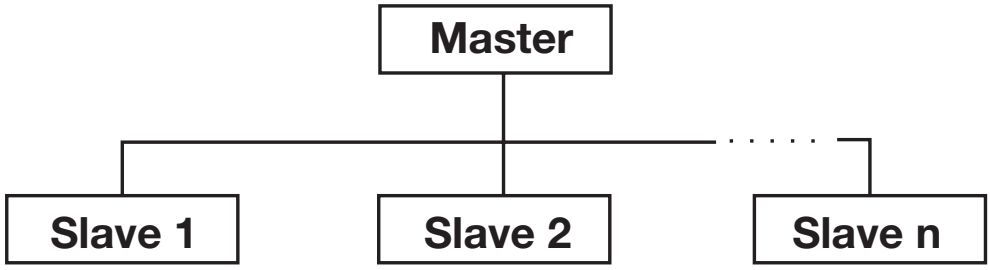
### 1.2.4 Number types

| | | |
|---|---|---|
| **Hexadecimal number** | 0x0010 | A hexadecimal number is identified by „0x“ preceding the actual number (here: 16 decimal). |

# 1 Introduction

## 2.1 Master-Slave principle

Communication between a master (e.g. PC) and a slave (e.g. measuring and control system) using Modbus 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 device address.

## 2.2 Transmission mode (RTU)

The transmission mode used is the RTU mode (Remote Terminal Unit). Data is transmitted in binary format (hexadecimal) with 8 bits. The LSB (least significant bit) is transmitted first. The ASCII operating mode is not supported.

**Data format**

The data format describes the structure of a character transmitted. The following data format options are available:

| Data word | Parity bit | Stop bit 1/2 bit | Number of bits |
|---|---|---|---|
| 8 bits | — | 1 | 9 |
| 8 bits | even | 1 | 10 |
| 8 bits | odd | 1 | 10 |
| 8 bits | — | 2 | 10 |

# 2 Protocol description

## 2.3 Device address

The device address of the slave can be set between 0 and 254. Controller address 0 is reserved.

A maximum of 31 slaves can be addressed via the RS422/485 interface.

There are two different forms of data exchange:

**Query**

Data request/instruction by the master to a slave via the corresponding device address.

The slave addressed responds.

**Broadcast**

Instruction by the master to all slaves via the device address 0 (e.g. to transmit a specific value to all slaves).

The connected slaves do not respond. In such a case, the correct acceptance of the value by the slaves should be checked by a subsequent readout at each individual slave.

Data request with the device address 0 is meaningless.

## 2.4 Timing of the communication

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

The character transmission time (time required to transmit one single character) depends on the baud rate and the data format used (stop bits and parity bit).

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

**character transmission time [ms] = 1000 * 9 bit/baud rate**

For the other data formats, this is:

**character transmission time [ms]**
**= 1000 * (8 bits+parity bit+stop bit(s)) bit/baud rate**

**Timing**

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

**Example**

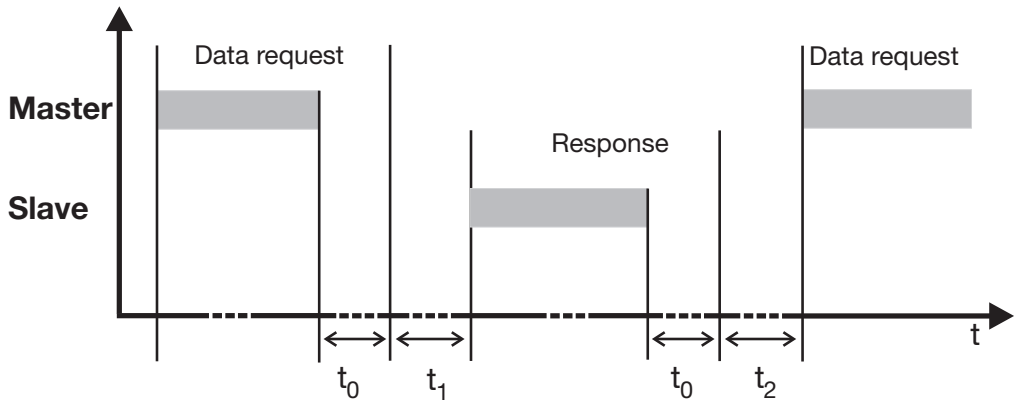
Marker for end of data request or end of response for a 10/9 bit data format

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

| **Baud rate** [baud] | **Data format** [bit] | **Waiting time** [ms] (3 characters) |
|---|---|---|
| 38400 | 10 | 0.79 |
| | 9 | 0.71 |
| 19200 | 10 | 1.57 |
| | 9 | 1.41 |
| 9600 | 10 | 3.13 |
| | 9 | 2.82 |

**Timing scheme**

A data request runs according to the following timing scheme:




$t_0$ End marker = 3 characters
(time depending on the baud rate)

$t_1$ This time depends on the internal processing.
The maximum processing time is 250 ms.

A minimum response time can be set in the device under the menu item „Interface“. This preset time is the minimum waiting time before an answer is transmitted (0…500 ms). If a smaller value is set, then the response time may be longer than the preset value (internal processing takes longer), the device answers as soon as internal processing is completed. The preset time of 0 ms means that the controller responds at the maximum possible speed.

The minimum response time which can be set is required by the RS485 interface in the master, to be able to switch over the interface drivers from transmit to receive. This parameter is not required for the RS422 interface.

$t_2$ This time is needed by the slave to change from transmit back to receive. The master has to observe this waiting time before presenting a new data request. This time must always be observed, even when the new data request is directed to another device.

RS422 interface: $t_2$ = 1ms
RS485 interface: $t_2$ = 10ms

No data requests from the master are permitted during $t_1$, $t_2$ and during the response time of the slave. Data requests made during $t_1$ and $t_2$ are ignored by the slave. Data requests during the response time will result in the invalidation of all data currently on the bus.

## 2.5 Structure 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 | 2 bytes |

Each data block contains four fields:

**Slave address** device address of a specific slave

**Function code** function selection (read, write words)

**Data field** contains the following information:

- word address
- number of words
- word value(s)

**Checksum** detection of transmission errors

## 2.6 Modbus/J bus differentiation

With some of the devices, a differentiation between Modbus and J bus is made in the device configuration.

The Modbus protocol is compatible with the J bus protocol, i. e. the data blocks are of an identical structure. The only difference is that the word address (NOT the slave address) is shifted by one.

| **Modbus address** | **J bus address** |
|---|---|
| 0x0000 | 0x0001 |
| 0x0001 | 0x0002 |
| 0x0002 | 0x0003 |
| ... | ... |

**Example** Address for displayed value of analog input 1:

Modbus protocol: 0x0035 (according to address table)
J bus protocol: 0x0036

The device-specific address tables always include the Modbus addresses. Where the J bus protocol is used, the addresses contained must be increased by one.

# 2 Protocol description

## 2.7 Function codes

The functions described in the following are available for the readout of measured values, device and process data as well as to write specific data.

**Function overview**

| Function number | Function | Limitation |
|---|---|---|
| 0x03 or 0x04 | Read n words | max. 32 words (64 bytes) |
| 0x06 | Write one word | max. 1 word (2 bytes) |
| 0x10 | Write n words | max. 32 words (64 bytes) |

⇨ Please refer to Chapter 2.10 Error processing, Page 20 if the controller does not react to these functions or emits an error code.

### 2.7.1 Read n words

This function is used to read n (n ≤ 32) words starting from a specific address.

**Data request**

| Slave address | Function x03 or 0x04 | Address first word | Number of words (max. 32) | Checksum CRC16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 byte | 2 byte | 2 byte |

**Response**

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

**Example**

Read the displayed value of analog input 1 and 2

Word address = 0x0035 (displayed value of analog input 1)

Data request:

| 01 | 03 | 0035 | 0004 | 5407 |
|---|---|---|---|---|

Response (values in the Modbus float format):

| 01 | 03 | 08 | 0000 | 41C8 | 0000 | 4120 | 4A9E |
|---|---|---|---|---|---|---|---|
| | | | Value of input 1 (25.0) | | Value of input 2 (10.0) | | |

### 2.7.2 Write one word

For the Write 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 byte | 2 bytes | 2 byte |

**Response**

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

**Example**

Write the limit value AL of limit comparator 1 = 275.0
(Value = 0x80004389 in the Modbus float format)

Word address = 0x0043

Instruction: Write the first part of the value

| 01 | 06 | 0043 | 8000 | 19DE |
|---|---|---|---|---|

Response (as instruction):

| 01 | 06 | 0043 | 8000 | 19DE |
|---|---|---|---|---|

Instruction: Write the second part of the value

| 01 | 06 | 0044 | 4389 | A81F |
|---|---|---|---|---|

Response (as instruction):

| 01 | 06 | 0044 | 4389 | A81F |
|---|---|---|---|---|

### 2.7.3 Write n words

This function is used to write n (n ≤ 32) words starting from a specific address.

**Instruction**

| Slave address | Function 0x10 | Address first word | Number of words (max. 32) | Number of bytes | Word value(s) | ChecksumCRC16 |
|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 byte | 2 byte | 1 byte | x byte | 2 byte |

**Response**

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

**Example**

Write the fixed limit comparator setpoint values for LK1 and LK2

Word address = 0x3100 (fixed setpoint value LK1)

Instruction:

| 01 | 10 | 3100 | 0004 | 08 | 0000 | 41C8 | 0000 | 4120 | 2A42 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | Setpoint value LK1 (25.0) | | Setpoint value LK2 (10.0) | | |

Response:

| 01 | 10 | 3100 | 0004 | CF36 |
|---|---|---|---|---|

## 2.8 Transmission format (integer, float and text values)

**Integer values**

Integer values are transmitted via the Modbus in the following format:
The high byte first, followed by the low byte.

**Example**

Request of the integer value of address 0x0021, when this address contains the value "4" (word value 0x0004).

Request: 01 03 0021 0001 (+ 2 bytes CRC16)
Response: 01 03 02 **0004** (+ 2 bytes CRC16)

**Float values**

In the case of float values, the Modbus operates with the IEEE-754 standard format (32 bits), the only difference being that byte 1 and 2 are changed over with byte 3 and 4.

**Single-float format (32 bit) as per IEEE 754 standard**

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

S - sign bit
E - exponent (two's complement)
M - 23 bits normalized mantissa

**Modbus float format**

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

**Example**

Request of the float value of address 0x0035, when this address contains the value "550.0" (0x44098000 in IEEE-754 format).

Request: 01 03 0035 0002 (+ 2 bytes CRC16)
Response: 01 03 04 **8000 4409** (+ 2 bytes CRC16)

Once transmission from the device is completed, the bytes of the float value need to be swapped accordingly.

A large number of 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 find out the way float values are stored in your application. After the request, it might be necessary to swap the bytes in the interface program you are using.

**Character strings (texts)**

Character strings (texts) are transmitted in the ASCII format.

☞ To mark the end, the last character to be transmitted can be a "\0" (ASCII code 0x00). Characters after this mark are without significance.

The address tables show the max. possible number of characters in a data type, e.g. "TEXT24" (24 characters). When an end mark is used, then only 23 readable characters are available for the text in this example.

If no end mark is used, the use of the maximum number of characters (e.g. TEXT8 = 8 characters) indicated in the data type is required. This prevents characters still contained in the memory from being appended to the text.

Knowing that the transmission of texts takes place word by word (16 bits), 0x00 is additionally appended where an odd number of characters is used (incl. "\0").

**Example for data type TEXT8**

Read the name of process data E1 (here: "INPUT1 ") at address 0x3C80 (a max. of 8 characters can be saved)

ASCII code for "INPUT1" (with double space at the end):
**0x49, 0x4E, 0x50, 0x55, 0x54, 0x31, 0x20, 0x20**

Request: 01 03 3C80 0004 (+ 2 bytes CRC16)
Slave address = 01
Function = 03, i.e. read n words
Address = 3C80
Number of words to be read = 0004, because of the maximum of 8 characters

Response: 01 03 08 **49 4E 50 55 54 31 20 20** (+ 2 bytes CRC16)
Slave address = 01
Function = 03, i.e. read n words
Number of bytes read = 08

Variant:

ASCII code for "INPUT1" (without double space at the end):
**0x49, 0x4E, 0x50, 0x55, 0x54, 0x31, 0x00**

ASCII 0x00 ("\0") marks the end of the character string.

During transmission, **0x00** is additionally appended to obtain an even number of characters.

Response in this case: 01 03 08 **49 4E 50 55 54 31 00 00** (+ 2 bytes CRC16)

## 2.9 Checksum (CRC16)

The checksum (CRC16) serves to recognize transmission errors. If an error is identified during evaluation, the device concerned does not respond.

**Calculation scheme**

| | | | |
|---|---|---|---|
| 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 checksum is transmitted with the low byte first, followed by the high byte.

**Example**

Data request: Read two words, starting at address 0x00CE
(CRC16 = 0x92A5)

| 07 | 03 | 00 | CE | 00 | 02 | A5 | 92 |
|---|---|---|---|---|---|---|---|
| | | | | | | CRC16 | |

Response: (CRC16 = 0xF5AD)

| 07 | 03 | 04 | 00 | 00 | 41 | C8 | AD | F5 |
|---|---|---|---|---|---|---|---|---|
| | | | Word 1 | | Word 2 | | CRC16 | |

## 2.10 Error processing

**Error codes**

The following error codes exist:

| | |
|---|---|
| 1 | invalid function |
| 2 | invalid parameter address or<br>too many words are to be read or written |
| 8 | write access to parameter denied |

**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, i.e. the MSB (most significant bit) is set to 1.

**Example**

Data request:

| | | | | | | |
|---|---|---|---|---|---|---|
| 01 | 03 | 40 | 00 | 00 | 04 | CRC16 |

Response (with error code 2):

| | | | |
|---|---|---|---|
| 01 | 83 | 02 | CRC16 |

**Special cases**

The slave not responding can have the following origins:

- the baud rate and/or data format of Master and Slave are not compatible
- the device address used does not coincide with that of the slave address
- 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 the data request should be transmitted again once the timeout time (2 s) has elapsed.

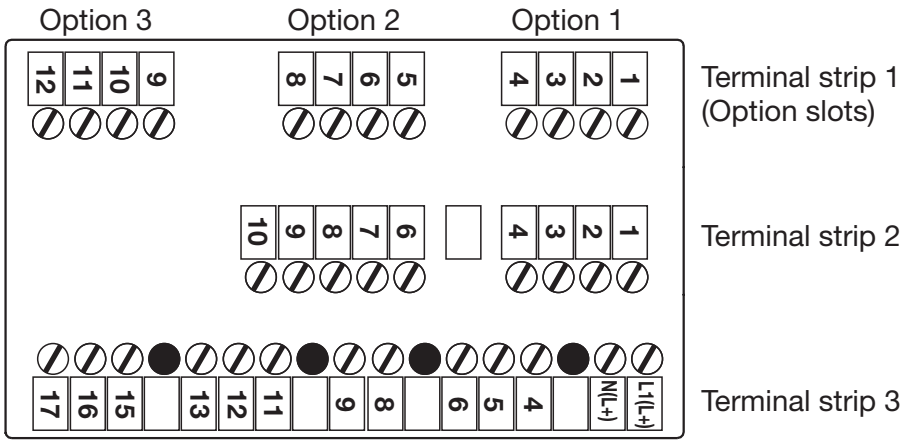
## 3.1 Connection diagram

**Equipment**

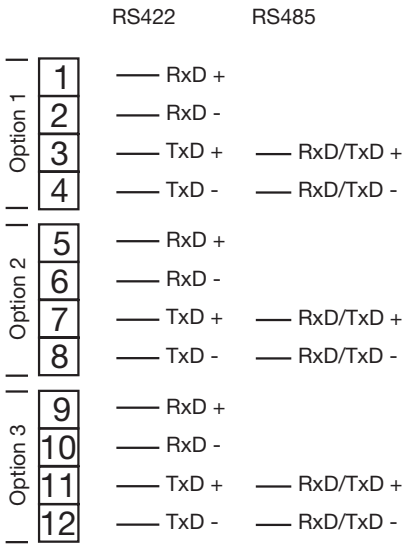
The RS422/485 interface can be plugged in one of the three optional slots. Any slot can be used, however, the number of interface modules to be used is limited to one.

When factory-equipped, the position of the interface module has to be identified by way of the type designation. Please refer to operating manual B 70.1550.0.

**View on the terminal strips**




**Optional assignment of terminal strip 1**



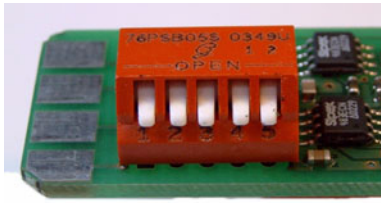
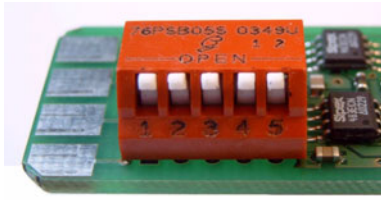
# 3 RS422/485 interface

## 3.2 Termination resistor

**Adjustment**

To ensure fault-free operation of several devices in a bus structure, the internal termination resistors must be activated for the first and the last device.

* Pull plug-in module out towards the front by pressing on the knurled areas
* Using a suitable aid (e.g. ballpoint pen), press all the white switches in the same direction

| | |
|---|---|
| Bus termination active | * Push all 5 switches down  |
| No bus termination (ex-factory) | * Push all 5 switches up  |

* Re-insert the module into the housing

**Check**

* Press the PGM + ▲ keys

When checking the software version and the termination resistors activated, an additional decimal point appears behind the version number (top display).

Resistors active (example with version number 01.01): 01.01.

Resistors inactive: 01.01

## 3.3 Configuration

The following table shows the possible settings of the RS422/485 interface, which are made on the configuration level
(CONFIG ➔ INTERFCE ➔ RS422485).

For more detailed information about configuration, please refer to operating manual B 70.1550.0.

| | Symbol | Value/ Selection | Description |
|---|---|---|---|
| **Protocol** | PROTOCOL | **0** | **Modbus** |
| | | 1 | Modbus integer |
| **Baud rate** | BAUD RATE | **0** | **9600 bps** |
| | | 1 | 19200 bps |
| | | 2 | 38400 bps |
| **Data format** | DFORMAT | **0** | **8 data bits, 1 stop bit, no parity** |
| | | 1 | 8 data bits, 1 stop bit, odd parity |
| | | 2 | 8 data bits, 1 stop bit, even parity |
| | | 3 | 8 data bits, 2 stop bits, no parity |
| **Device address** | ADDRESS | 0…**1**…255 | Address in data network |
| **Minimum response time** | | **0**…500 ms | Time period that elapses between the request of a device in the data network and the response of the display instrument (can only be adjusted via the setup program). |

Factory settings are shown **bold**.

When the communication takes place via the setup interface, the RS422/485 interface is inactive.

# 3 RS422/485 interface

**Data type, type of access**

The following tables contain descriptions of all process values (variables) including their addresses, data type and type of access.

Meaning:

| | |
|---|---|
| **INT** | Integer (8 or 16 bit) |
| **Bit x** | Bit No. x |
| **FLOAT** | Float value (4 bytes) as per IEEE 754 |
| **TEXT8** | Text 8 characters |
| **TEXT24** | Text 24 characters |
| **R/O** | Read only access |
| **R/W** | Read/write access |

## 4.1 Process data

| Address | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0021 | INT | R/O | Binary outputs 1..10<br>(Switching states 0 = off / 1 = on) |
| | Bit 0 | | Binary output 1 (= 0x0001) |
| | Bit 1 | | Binary output 2 (= 0x0002) |
| | Bit 2 | | Binary output 3 (= 0x0004) |
| | Bit 3 | | Binary output 4 (= 0x0008) |
| | Bit 4 | | Binary output 5 (= 0x0010) |
| | Bit 5 | | Binary output 6 (= 0x0020) |
| | Bit 6 | | Binary output 7 (= 0x0040) |
| | Bit 7 | | Binary output 8 (= 0x0080) |
| | Bit 8 | | Binary output 9 (= 0x0100) |
| | Bit 9 | | Binary output 10 (= 0x0200) |
| 0x0023 | INT | R/O | Binary inputs 1..8<br>(Switching states 0 = open / 1 = closed) |
| | Bit 0 | | Binary input 1 (= 0x0001) |
| | Bit 1 | | Binary input 2 (= 0x0002) |
| | Bit 2 | | Binary input 3 (= 0x0004) |
| | Bit 3 | | Binary input 4 (= 0x0008) |
| | Bit 4 | | Binary input 6 (= 0x0010) |
| | Bit 5 | | Binary input 5 (= 0x0020) |
| | Bit 6 | | Binary input 8 (= 0x0040) |
| | Bit 7 | | Binary input 7 (= 0x0080) |

# 4 Modbus addresses

| Address | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0026 | INT | R/O | Binary signals (4LK-2LOGIK) |
| | Bit 8 | | Limit comparator 1 (= 0x0100) |
| | Bit 9 | | Limit comparator 2 (= 0x0200) |
| | Bit 10 | | Limit comparator 3 (= 0x0400) |
| | Bit 11 | | Limit comparator 4 (= 0x0800) |
| | Bit 12 | | Logic formula 1 (= 0x1000) |
| | Bit 13 | | Logic formula 2 (= 0x2000) |
| 0x0027 | INT | R/W | Control of the binary outputs |
| | Bit 0 | | Binary output 1 (= 0x8001) |
| | Bit 1 | | Binary output 2 (= 0x8002) |
| | Bit 2 | | Binary output 3 (= 0x8004) |
| | Bit 3 | | Binary output 4 (= 0x8008) |
| | Bit 4 | | Binary output 5 (= 0x8010) |
| | Bit 5 | | Binary output 6 (= 0x8020) |
| | Bit 6 | | Binary output 7 (= 0x8040) |
| | Bit 7 | | Binary output 8 (= 0x8080) |
| | Bit 8 | | Binary output 9 (= 0x8100) |
| | Bit 9 | | Binary output 10 (= 0x8200) |
| | Bit 15 | | Activation = 1 (=0x8000) |
| 0x0030 | FLOAT | R/O | Internal Pt100 [Ohm] |
| 0x0034 | INT | R/O | Sampling cycle time |
| 0x0035 | FLOAT | R/O | Analog input 1 [displayed value] |
| 0x0037 | FLOAT | R/O | Analog input 2 [displayed value] |
| 0x003D | FLOAT | R/O | Math 1 |
| 0x003F | FLOAT | R/O | Math 2 |
| 0x3300 | FLOAT | R/O | Minimum value, analog input 1 [displayed value] |
| 0x3302 | FLOAT | R/O | Minimum value, analog input 2 [displayed value] |
| 0x3308 | FLOAT | R/O | Maximum value, analog input 1 [displayed value] |
| 0x330A | FLOAT | R/O | Maximum value, analog input 2 [displayed value] |
| 0x3400 | FLOAT | R/O | Hold value, analog input 1 [displayed value] |
| 0x3402 | FLOAT | R/O | Hold value, analog input 2 [displayed value] |

## 4.2 Commands and configuration

| Address | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x0041 | INT | R/W | Binary functions |
| | Bit 1 | | Display hold value (0x0002) |
| | Bit 2 | | Acknowledge limit comparator (0x0004) |
| | Bit 3 | | Reset min-max value (0x0008) |
| | Bit 4 | | Reset tare function (0x0010) |
| | Bit 5 | | Tare function (0x0020) |
| | Bit 6 | | Assign hold value (0x0040) |
| 0x0042 | INT | R/W | Binary functions Display / Operation |
| | Bit 0 | | Key inhibit (0x0001) |
| | Bit 1 | | Configuration inhibit (0x0002) |
| | Bit 4 | | Display OFF (keys inactive) (0x0010) |
| | Bit 5 | | Color change (0x0020) |
| | Bit 6 | | Text display (0x0040) |
| | Bit 7 | | Go to the next scroll parameter (0x0080) |
| 0x0043 | FLOAT | R/W | Limit comparator 1 Limit value AL |
| 0x0045 | FLOAT | R/W | Limit comparator 1 Switching differential |
| 0x0047 | INT | R/W | Limit comparator 1 Switch-on delay |
| 0x0048 | INT | R/W | Limit comparator 1 Switch-off delay |
| 0x0049 | FLOAT | R/W | Limit comparator 2 Limit value AL |
| 0x004B | FLOAT | R/W | Limit comparator 2 Switching differential |
| 0x004D | INT | R/W | Limit comparator 2 Switch-on delay |
| 0x004E | INT | R/W | Limit comparator 2 Switch-off delay |
| 0x004F | FLOAT | R/W | Limit comparator 3 Limit value AL |
| 0x0051 | FLOAT | R/W | Limit comparator 3 Switching differential |
| 0x0053 | INT | R/W | Limit comparator 3 Switch-on delay |
| 0x0054 | INT | R/W | Limit comparator 3 Switch-off delay |
| 0x0055 | FLOAT | R/W | Limit comparator 4 Limit value AL |
| 0x0057 | FLOAT | R/W | Limit comparator 4 Switching differential |
| 0x0059 | INT | R/W | Limit comparator 4 Switch-on delay |
| 0x005A | INT | R/W | Limit comparator 4 Switch-off delay |
| 0x005B | FLOAT | R/W | Analog value |
| 0x005D | INT | R/W | Binary value |
| 0x3100 | FLOAT | R/W | Limit comparator 1 Fixed LK setpoint value |
| 0x3102 | FLOAT | R/W | Limit comparator 2 Fixed LK setpoint value |
| 0x3104 | FLOAT | R/W | Limit comparator 3 Fixed LK setpoint value |
| 0x3106 | FLOAT | R/W | Limit comparator 4 Fixed LK setpoint value |
| 0x3A00 | TEXT24 | R/W | AlarmText B1 |
| 0x3A0C | TEXT24 | R/W | AlarmText B2 |
| 0x3A18 | TEXT24 | R/W | AlarmText B3 |
| 0x3A24 | TEXT24 | R/W | AlarmText B4 |

# 4 Modbus addresses

| Address | Data type/ bit number | Access | Signal designation |
|---|---|---|---|
| 0x3A30 | TEXT24 | R/W | AlarmText B5 |
| 0x3A3C | TEXT24 | R/W | AlarmText B6 |
| 0x3A48 | TEXT24 | R/W | AlarmText B7 |
| 0x3A54 | TEXT24 | R/W | AlarmText B8 |
| 0x3A60 | TEXT24 | R/W | AlarmText LK1 |
| 0x3A6C | TEXT24 | R/W | AlarmText LK2 |
| 0x3A78 | TEXT24 | R/W | AlarmText LK3 |
| 0x3A84 | TEXT24 | R/W | AlarmText LK4 |
| 0x3A90 | TEXT24 | R/W | AlarmText L1 |
| 0x3A9C | TEXT24 | R/W | AlarmText L2 |
| 0x3C00 | TEXT24 | R/W | Process text |
| 0x3C80 | TEXT8 | R/W | Name Process data E1 |
| 0x3C84 | TEXT8 | R/W | Name Process data E2 |
| 0x3C90 | TEXT8 | R/W | Name Math 1 |
| 0x3C94 | TEXT8 | R/W | Name Math 2 |
| 0x3200 | FLOAT | R/W | Math 1 (writable) |
| 0x3202 | FLOAT | R/W | Math 2 (writable) |
| 0x3204 | FLOAT | R/W | Analog input E1 (writable) |
| 0x3206 | FLOAT | R/W | Analog input E2 (writable) |
| 0x320C | INT | R/W | Logics (writable) |
| | Bit 0 | | Logics L1 (0x0001) |
| | Bit 1 | | Logics L2 (0x0002) |
| | Bit 6 | | Activation L2 (0x004X) / Deactivation (0x0000) |
| | Bit 7 | | Activation L1 (0x008X) / Deactivation (0x0000) |

Modbus allows direct access to the RAM memory of the device for reading and writing the measured values of the analog inputs (addresses 0x3204, 0x3206), the math function results (addresses 0x3200, 0x3202) as well as the logics functions (address 0x320C).

When writing, a range between -19999 and +99999 is available for the analog input and math data. In this case, the data written in the device is used instead of the original value. This permits, for instance, the transmission of a value to the device via Modbus that was measured at a different point for display purposes or for use as an input value for other functions (limit comparator, math).

If you wish to use the original value again on the device, write the value 200000 at the memory location concerned via Modbus.

When writing texts via Modbus, texts existing in the device will be lost. If you wish to write new texts via Modbus, we recommend to read out and save existing texts for information. The original texts can only be retrieved by re-transmitting them by either using the setup program or via Modbus.

Write operations to R/W parameters result in them being saved to the EEPROM. These memory modules only have a limited number of write cycles (approx. 10000). For this reason, this function can be switched off in the case of frequent programming. The parameter values are then only saved in the volatile memory (RAM) and will be lost after a supply failure.

⇨ *Setup/Only Setup/Undocumented parameters/Bit parameters/ Set parameter 2*

# 4 Modbus addresses

**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 CM 20 2 TT, 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