JUMO digiLine O-DO S10

Oxygen sensor with digital interface

Modbus interface description

20261400T92Z001K000

V2.00/EN/00753777/2021-08-20

Contents

Contents

1 Safety information

1.1 Warning symbols

DANGER!

This symbol indicates that **personal injury from electrocution** may occur if the appropriate precautionary measures are not taken.

WARNING!

This symbol in connection with the signal word indicates that **personal injury** may occur if the respective precautionary measures are not carried out.

CAUTION!

This symbol in connection with the signal word indicates that **material damage or data loss** will occur if the respective precautionary measures are not taken.

CAUTION!

This symbol indicates that **components could be destroyed** by electrostatic discharge (ESD = Electro Static Discharge) if the respective cautionary measures are not taken.

Only use the ESD packages intended for this purpose to return device inserts, assembly groups, or assembly components.

READ THE DOCUMENTATION!

This symbol, which is attached to the device, indicates that the associated **documentation for the device** must be **observed**. This is necessary to identify the nature of the potential hazard, and to take measures to prevent it.

1.2 Note signs

NOTE!

This symbol refers to **important information** about the product, its handling, or additional benefits.

REFERENCE!

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

FURTHER INFORMATION!

This symbol is used in tables and indicates that **further information** is provided after the table.

DISPOSAL!

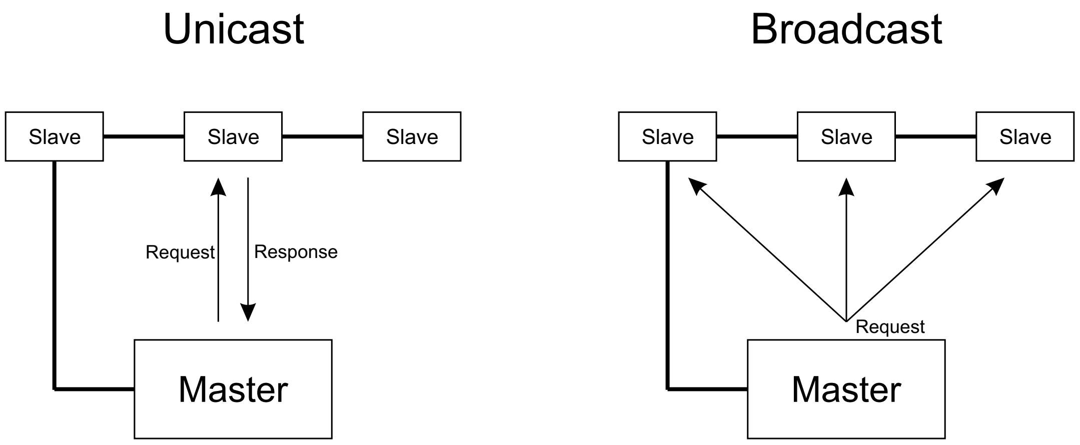
At the end of its service life, the device and any batteries present do not belong in the trash! Please ensure that they are **disposed of** properly and in an **environmentally friendly** manner.

1 Safety information

2.1 Master-slave principle

Communication between a master (e.g. a SCADA system or PLC) and a JUMO digiLine O-DO S10 as slave via Modbus takes place on the basis of the master-slave principle in the form of a data request/ instruction – response. Slaves are identified by the device address. Master devices do not need an address.

⇨ chapter 5 "Modbus over a serial interface", Page 25



The master controls data exchange by cyclically querying the slaves on the overall bus. The slaves (e.g. JUMO digiLine O-DO S10) have only a response function. The master has write and read access to the slaves. This enables data to be communicated between master and slave devices in real-time. Slaves cannot communicate with one another directly. In order to transmit data from slave to slave, the master must extract data from one slave and transfer it to the next.

As a rule, the master directs its queries to individual slaves. For this, it must communicate with the individual slaves by means of their unicast address. Queries can be directed to all slaves on the bus also in the form of a broadcast message. In this case, the broadcast address is used as the slave address. Slaves do not respond to broadcast queries. Data collisions would otherwise result in serial bus systems. For this reason, use of broadcast addresses only makes sense with function codes for writing data. Broadcasts cannot be used with function codes for reading data.

NOTE!

The JUMO digiLine O-DO S10 can only be operated as a slave.

2.2 Transmission media for Modbus

Serial interface

For data communication via a **serial interface**, the Modbus specification provides the transmission modes **RTU mode** (**R**emote **T**erminal **U**nit) and ASCII mode (transmission of data in ASCII format). The JUMO digiLine O-DO S10 only supports the **RTU mode**. Here, the data is transferred via the serial bus in binary format (RS485).

⇨ chapter 5 "Modbus over a serial interface", Page 25

2.3 Structure of an RTU Modbus telegram

Modbus telegrams have the following structure:

Slave address	Function code	Data field	Checksum CRC
1 byte	1 byte	x bytes	2 bytes

Every telegram has four fields:

Slave address	Device address of a specific slave
Function code	Function selection (read/write words)
Data field	Contains the information (depending on the function code) - word address/bit address - number of words/number of bits - word value(s)/bit value(s)
Checksum	Detection of transmission errors

2.4 Function codes

Function overview

The functions described in the following (from the Modbus standard) are available for extracting measured values, device and process data, and for writing data.

Function code		**Function**	**Limit**
Hex	**Dec.**		
03 or 04	3 or 4	Reading n words	Max. 125 words (250 bytes)
06	6	Writing one word	Max. 1 word (2 bytes)
10	16	Writing n words	Max. 123 words (246 bytes)

2.4.1 Reading n words

This function is used to read n words, starting from a specific address.

Data request

Slave address	Function 0x03 or 0x04	Address of first word	Number of words x	Checksum CRC
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 CRC
1 byte	1 byte	1 byte	2 x bytes	2 bytes

Example

Reading the device name. In the example here, the device name is "dl CR". Since the device name is up to 10 characters long (including ZERO characters as end identifier) and 2 ASCII characters of the device name are saved per word, it is necessary here to read 5 words, i.e. 10 bytes. For these and other Modbus addresses, see chapter 6 "Modbus address tables", Page 27.

Hex code for the data request:

01	03	00 00	00 05	85 C9
Slave	Function	Address of 1st word	Number of words	CRC

Hex code for the response (values in byte format):

01	03	0 A	64 6C 20 43 52 00 00 00 00 00	E6 B1
Slave	Function	Bytes read	Text in ASCII	CRC

2.4.2 Writing one word

The data blocks for the instruction and response are identical when writing a word.

CAUTION!

Write operations in some R/W parameters result in them being saved in the EEPROM or flash memory. These memory modules have only a limited number of write cycles (approx. 100,000 or 10,000).

Thus, frequent writing of certain variables can result in a memory error.

▶ The number of write operations should therefor be kept as low as possible.

Instruction

Slave address	Function 0x06	Word address	Word value	Checksum CRC
1 byte	1 byte	2 bytes	2 bytes	2 bytes

Response

Slave address	Function 0x06	Word address	Word value	Checksum CRC
1 byte	1 byte	2 bytes	2 bytes	2 bytes

Example

In this example, a command is to write the year "2021" (hexadecimal value: 0x07E5) to the data for the date and time of the device. In this case, the slave address of the device should be 1 and the relevant word address of the year is 0x2840 (chapter 6.9 "Date and time for time stamp", Page 35).

Hex code for the instruction:

01	06	28 40	07 E5	42 05
Slave	Function	Word address	Value	CRC

Hex code for the response:

01	06	28 40	07 E5	42 05
Slave	Function	Word address	Value	CRC

2.4.3 Writing n words

Instruction

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

Response

Slave address	Function 0x10	Address of first word	Number of words x	Checksum CRC
1 byte	1 byte	2 bytes	2 bytes	2 bytes

Example

Writing the word "North plant" (UTF-8 encoding with end identifier: 0x41 0x6E 0x6C 0x61 0x67 0x65 0x20 0x4E 0x6F 0x72 0x64 0x00) after word address 0x021E as description of the measuring point.

⇨ chapter 6.3 "Measuring point info", Page 28

Hex code for the instruction:

01	10	02 1E	00 06	0C	41 6E 6C 61 67 65 20 4E 6F 72 64 00	F8 70
Slave	Function	Address 1st word	Number of words	Number of bytes	Text in UTF-8 encoding	CRC

Hex code for the response:

01	10	02 1E	00 06	21 B5
Slave	Function	Address 1st word	Number of words	CRC

2.5 Data types

Data type	Description	Access	Possible Function codes	Number Modbus register
bool	Least significant bit in a word as bit value 0000 0000 0000 000**1** = 1 or TRUE 0000 0000 0000 000**0** = 0 or FALSE	read only	03, 04	1
		read/ write	03, 04, 06, 16	
word	Word (16 bit) as bit field The specific meanings of the individual bits can be found in the Modbus address tables in the last chapter of this document.	read only	03, 04	1
		read/ write	03, 04, 06, 16	

Data type	Description	Access	Possible Function codes	Number Modbus register
char	Low byte of a word as an integer value; the high byte is not used. Value ranges for numerical interpretation: 0 to 255 for unsigned data -128 to 127 for signed data For Boolean interpretation: Least significant bit in a word as bit value 0000 0000 0000 000**1** = 1 or TRUE 0000 0000 0000 000**0** = 0 or FALSE	read only	03, 04	1
		read/ write	03, 04, 06, 16	
short	Word (16 bit) as integer value Value ranges: 0 to 65535 for unsigned data -32768 to 32767 for signed data	read only	03, 04	1
		read/ write	03, 04, 06, 16	
Uint32	Double word (32-bit) as unsigned integer value Value range: 0 to 4.294.967.295	read only	03, 04	2
		read/ write	03, 04, 16	
enum	Word (16 bit) as integer value Numbering for settings (the following number range is possible: 0 to 65535). The meaning of the specific numbering for enumeration variables can be found in the Modbus address tables in the last chapter of this document.	read only	03, 04	1
		read/ write	03, 04, 06, 16	

Data type	Description	Access	Possible Function codes	Number Modbus register

float	2 words as 32-bit floating-point number with encoding to IEEE 754, where the correct order of transmitting the 4 bytes must be observed during transmission. In the configuration of the RS485 interface, the desired encoding can be selected under the "Floating-point format" setting.
	S = Sign bit
	E = Exponent (two's complement)
	M = 23-bit normalized mantissa
		read only	03, 04	2

		read/ write	03, 04, 16	

IEEE 754 Big Endian

Byte 1	Byte 2	Byte 3	Byte 4
$SE_7EEEEEE_1$	$E_0M_{23}MMMMMM_{16}$	$M_{15}MMMMMMM_8$	$M_7MMMMMMM_0$

IEEE 754 Little Endian

Byte 1	Byte 2	Byte 3	Byte 4
$M_7MMMMMMM_0$	$M_{15}MMMMMMM_8$	$E_0M_{23}MMMMMM_{16}$	$SE_7EEEEEE_1$

Standard modbus coding

Byte 1	Byte 2	Byte 3	Byte 4
1. Modbus register		2. Modbus register	
$M_{15}MMMMMMM_8$	$M_7MMMMMMM_0$	$SE_7EEEEEE_1$	$E_0M_{23}MMMMMM_{16}$

Data type	Description	Access	Possible Function codes	Number Modbus register
string[n]	Character string for a length of up to n bytes. The character encoding of the individual variables can be found in the Modbus address tables in the last chapter of this document. UTF-8, ISO 8859-1 or ASCII encoding appear as character set codings. Each of the n Modbus register words contains 2 successive bytes of the character string. Note that the character string must always have "\0" (hexadecimal: 0x00) as the terminating code. In addition, the character string length must have an even number of bytes overall. To ensure this in the case of an odd number of bytes, including "/0", an additional "/0" must be appended.	read only	03, 04	n/2
		read/ write	03, 04, 06, 16	

2.6 Examples of data transmission

The function 0x03 or 0x04 (writing of n words) is used to extract integers, floating-point values and text values.

Data request

Slaveaddress	Function 0x03 or 0x04	Address of first word	Number of words	Checksum CRC
1 byte	1 byte	2 bytes	2 bytes	2 bytes

Integer values are transferred via Modbus in the following format:
First the high, then the low byte.

Response

Slaveaddress	Function 0x03 or 0x04	Number of bytes read	Word value(s)	Checksum CRC
1 byte	1 byte	1 byte	x bytes	2 bytes

2.6.1 Integer values

Example

In this example, the year of the initial startup at address 0x0500 (chapter 6.7.1 "Initial startup", Page 30) is to be extracted. The value here is to be "2020" (word value 0x07E4).

Data request:

01	03	05 00	00 01	84 C6
Slave	Function	Address of 1st word	Number of words	CRC

Response (values in Modbus floating-point format):

01	03	02	07 E4	BA 3F
Slave	Function	Bytes read	Integer value	CRC

2.6.2 Float values

For floating-point values, this device functions with the IEEE 754 standard format (32-bit), but with the difference that bytes 1 and 2 are interchanged with bytes 3 and 4.

Single floating-point format (32-bit) according to standard IEEE 754

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

S - Prefix sign bit
E - Exponent (two's complement)
M - 23-bit normalized mantissa

Modbus floating-point format

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

Example

In this example, the measured temperature value at address 0x2620 in the device is to be extracted. The value here should be 20.25 (0x41A20000 in IEEE 754 format).

Data request:

01	03	26 20	00 02	CE 89
Slave	Function	Address of 1st word	Number of words	CRC

Response (values in Modbus floating-point format):

01	03	04	00 00	41 A2	4B DA
Slave	Function	Bytes read	Floating-point value		CRC

After being transmitted from the device, the bytes for the floating-point value must be interchanged accordingly. Many compilers (for example, Microsoft® Visual C++) store the floating-point values in the following sequence:

Floating-point value

Address x	**Address x+1**	**Address x+2**	**Address x+3**
MMMMMMMM	MMMMMMMM	EMMMMMMM	SEEEEEEE
Byte 4	Byte 3	Byte 2	Byte 1

NOTE!

The sequence of the bytes depends on how floating-point values are saved in the application concerned. Conventional standardized floating-point formats can be selected in the settings of the serial interface. However, under certain circumstances it may be necessary to adjust the byte sequence by making changes in the interface program of the master.

2.6.3 Character strings (texts)

Character strings are coded in UTF8 for transmission.

NOTE!

A "\0" (hexadecimal: 0x00) must always be transmitted as the terminating code. Subsequent characters have no meaning.
Since texts are transmitted word by word (16-bit register), 0x00 is also added to the end for an uneven number of bytes (incl. "\0").
The maximum lengths for character strings specified in the address tables contain the final "\0". This means that, in the case of "String[60]", the text, including "\0", must not be longer than 60 bytes.
In the case of 19 Unicode characters having a length of 3 bytes each, only 2 bytes remain for the 20th character. 1 byte is needed for the terminating code.

Example

Querying the text from address 0x021E, this address holds the character string for the description of the measuring point "North plant" (ASCII code: 0x41 0x6E 0x6C 0x61 0x67 0x65 0x20 0x4E 0x6F 0x72 0x64).

Hex code for the query:

01	03	02 1E	00 07	65 B6
Slave	Function	Address 1st word	Number of words	CRC

Hex code for the response:

01	03	00 0E	41 6E 6C 61 67 65 20 4E 6F 72 64 00 00 AA	C5 DF
Slave	Function	Bytes read	Word values (ASCII characters)	CRC

NOTE!

The values (here: 00 AA) before the CRC sum (here: C5DF) are not taken into consideration, since they are after the end identifier "\0".

2.7 Checksum (CRC16)

Calculation principle

Transmission errors are detected with the aid of the checksum (CRC16). If an error is detected during evaluation, the device concerned does not respond.

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

NOTE!

The low byte of the checksum is transmitted first!
Example: The CRC16 checksum CC DD is transmitted and represented in the sequence DD CC.

2.8 Error messages

2.8.1 Modbus error codes

Requirements for Modbus communication

The following conditions must be met for a slave to receive, process, and respond to queries:

- Baud rate and data format of master and slave must match.
- The correct slave address must be used in the query.
- Slave devices respond only after a successful checksum check of the query by the slave. Otherwise, the query is rejected by the slave.
- The instruction from the master must be complete and conform to the Modbus protocol.
- The number of words to be read must be greater than 0.

Error codes

If the data request from the master has been received by the slave without transmission errors but cannot be processed, the slave responds with an error code. The following error codes may occur:

- 01 = Invalid function; the function codes supported by this device are listed in chapter 2.4 "Function codes", Page 8.
- 02 = Invalid address or too many words or bits are to be read or written
- 03 = Value is outside the admissible range
- 08 = Value is write-protected

Response to malfunction

Slave address	Function XX OR 80h	Error code	Checksum CRC
1 byte	1 byte	1 byte	2 bytes

The function code is OR-gated with 0x80. As a result, the highest-value bit (msb) is set to 1.

Example

Data query:

01	06	23 45	00 01	52 5B
Slave	Write word	Word address	Word value	CRC

Response (with error code 2):

01	86	02	C3 A1
Slave	OR function	Errors	CRC

Response with error code 02, because the address 0x2345 does not exist.

2.8.2 Error messages for invalid values

For measured values in float format, the error number itself is displayed as a value, i.e. instead of the measured value, the error number is returned.

Error code for floating-point values	**Error**
1.0×10^{37}	Underrange
2.0×10^{37}	Overrange
3.0×10^{37}	Not a valid input value
4.0×10^{37}	Division by zero
5.0×10^{37}	Mathematical error
6.0×10^{37}	Invalid compensation temperature
7.0×10^{37}	Probe short circuit
8.0×10^{37}	Probe break
9.0×10^{37}	Timeout

Example

Loading of the measured oxygen value at Modbus address 0x2600:

Data request:

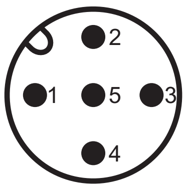
08	03	26 00	00 02	CF DA
Slave	Function	Word address	Number of words	CRC

Response:

08	03	04	BD C2	7D 70	C6 17
Slave	Function	Bytes read	Error code		CRC

The measured value provided by the analog input 0x7D70BDC2 (=2.0×10^{37}) indicates that an over-range situation exists.

3.1 Interface assignment

Function	**Pin**	**Figure** (socket)
not connected	1	
+24 V voltage supply from indicating device/controller	2	
GND	3	
RS485 B (RxD/TxD-)	4	
RS485 A (RxD/TxD+)	5	
The connection to the serial interface of a master device or a transmitter with screw or spring-cage terminals is established with the aid of a JUMO M12 digiLine master connecting cable (see accessories in data sheet/operating manual).		

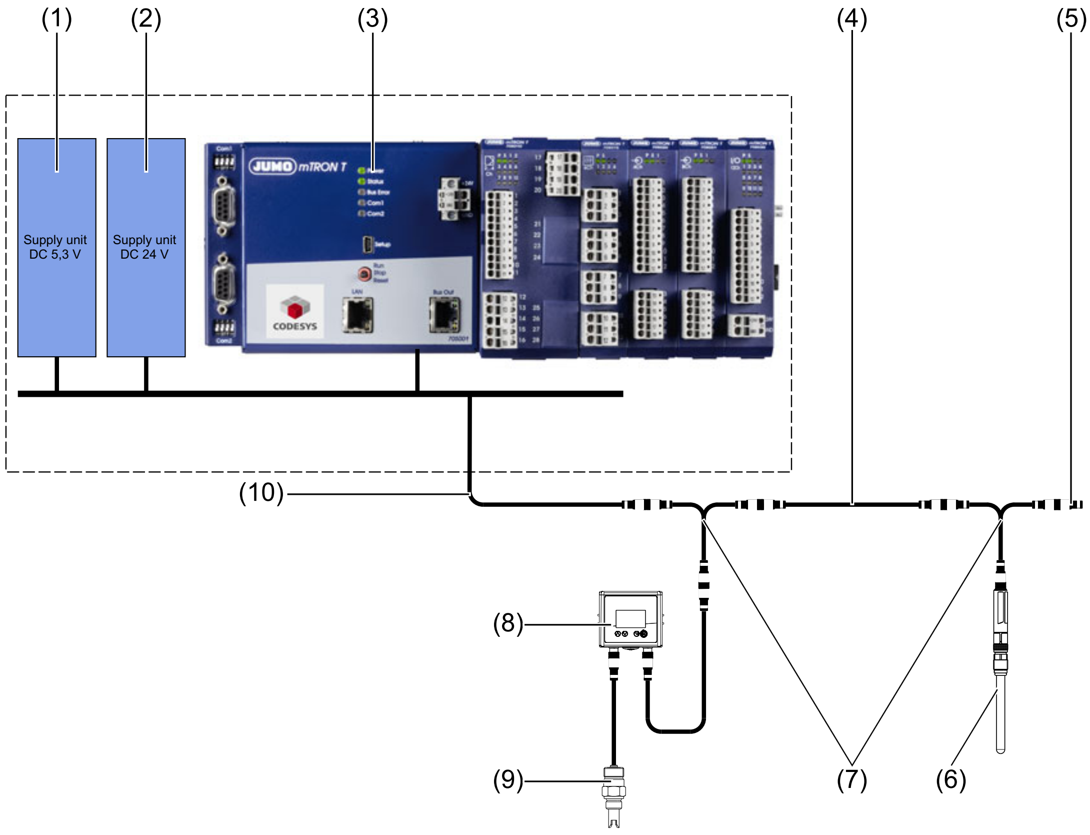
NOTE!

Preassembled connecting cables are available for connecting sensors to JUMO digiLine electronics. Use these to assure a reliable cable connection and data transmission in the bus system. In the order details provided in the operating manual, you can find a list of cables and connectors available as accessories.

3.1.1 Terminating resistors

A JUMO digiLine bus cable needs to be terminated with terminating connectors at the ends distant from the master unit (part no.: 00461591). These are simply connected to the last free M12 coupling at the final Y-splitter (part no.: 00638327) in the bus. At the master unit end of the bus cable, termination of the RS-485 bus cable must be made in accordance with the operating manual for the Modbus master unit.

Example of connecting 2 sensors with JUMO digiLine electronics on a JUMO mTRON T



(1) Stabilized power supply unit with DC 5.3 V output for feeding sensors with JUMO digiLine pH/ORP/T five-pole (current limiting via 3 A fuse required)

(2) Stabilized power supply unit with DC 24 V output for feeding sensors with JUMO digiLine electronics and the JUMO mTRON T (current limiting via 3 A fuse required)

(3) JUMO mTRON T central processing unit with activated PLC function and RS422/485 Modbus RTU (see order data for JUMO mTRON T)

(4) JUMO M12 connecting cable, 5-pole and A-coded

(5) JUMO M12 terminating connector, 5-pole for bus termination

(6) JUMO sensors with JUMO digiLine pH/ORP/T five-pole

(7) JUMO Y-splitter, 5-pole with 2× M12 cable sockets and 1× M12 connector, each of which is A-coded

(8) JUMO digiLine CR in device version with RS485 interface

(9) Conductivity sensor

(10)JUMO digiLine master connecting cable with exposed wire ends at one end for connection to devices with screw or spring-cage terminals (see accessories)

Notes

CAUTION!

Improper installation or the wrong settings on equipment can result in unexpected operating states of a plant.

This can disrupt processes from functioning properly or result in damage.

▶ For this reason, it is always necessary to provide safety devices that are independent of the device and to allow settings to be made only by qualified personnel.

NOTE!

Changes to the configuration settings described in this chapter for JUMO digiLine electronics can be made on a PC using the JUMO DSM software.

4 Configuring interfaces

Settings for serial interfaces

For all user devices on a bus to be able to communicate with one another, their interface settings must be correct. The table below shows the setting options for the serial interfaces on the JUMO digiLine electronics.

NOTE!

The JUMO digiLine protocol assigns the interface parameters automatically during startup (Plug & Play). For operation on a Modbus master, the interface parameters must be set prior to initial commissioning with the JUMO DSM software.

Configuration item	Selection/settings	Description
Baud rate	9600 19200 38400	Transmission speed (symbol rate) of the serial interface (must match for all bus users)
Data format	8 - 1 - no parity 8 - 1 - odd parity 8 - 1 - even parity 8 - 2 - no parity	Format of the data word (must match for all bus users) Useful bit- stop bit - parity
Floating Point	Standard IEEE754_LITTLE IEEE754_BIG	Selectable transmission format for float values (floating-point numbers)
Minimum response time	0 to 500 ms	Minimum time from receipt of a query to transmission of a response This parameter is used to adjust the response speed of the device to slower bus users.
Device address	1 to 247	A bus user's unique identifier 0 = Broadcast address[a] 1 to 247 = Unicast addresses[b]

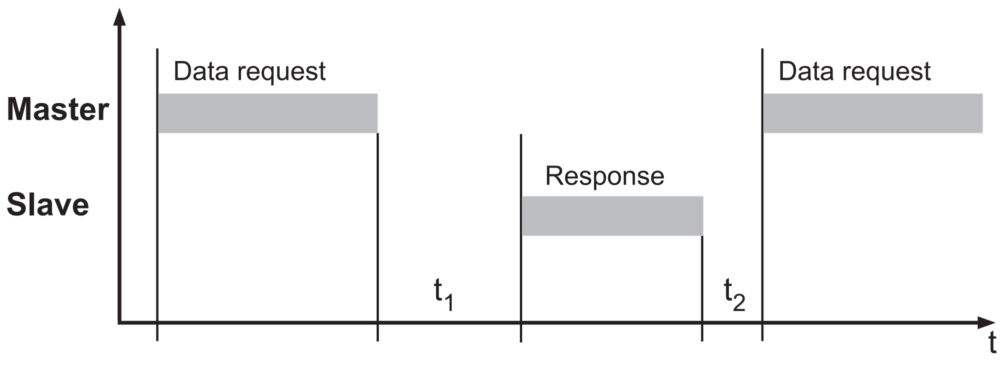
[a] Device addressing is specified in the Modbus standard. The broadcast address must not be used as a slave address. It is intended for broadcast messages.

[b] Unicast addresses are intended for use as slave addresses. They are used to identify the slave devices uniquely so that the master can communicate with them explicitly.

5.1 Modbus slave operation via RS485 serial interface

Chronological sequence of communication

A scanning cycle on a bus proceeds with the following timing:



t_1	Device's internal waiting period prior to checking of the data query and the internal processing time: min.: 3.5 byte times + minimum response time typically: 5 ms max.: 25 ms + minimum response time
t_2	The master has to observe this waiting period before starting a new data request: 3.5 characters or at least 2 ms

NOTE!

The minimum response time can be set in the configuration.
This set time must elapsed before a response is sent (0 to 500 ms). If processing of a query from a master is completed in the slave before the minimum response time has elapsed, the response is not sent until the "minimum response time" has elapsed.

NOTE!

During t_1 and t_2 and during the response time of the slave, no data requests may be generated by the master. Requests during t_1 and t_2 are ignored by the slave. Requests during the response time invalidate all the data currently on the bus.

NOTE!

The terminating code after a data query or data response is 3.5 characters long. The time needed for these 3.5 characters depends on the baud rate.

Character transmission time

The beginning and end of a data block are identified by pauses in transmission. The character transmission time (time to transmit one character) depends on the baud rate.
The following results for a data format of 8 data bits, no parity bit, and one stop bit:

Character transmission time [ms] = 1000 × 10 bits ÷ baud rate

For other data formats, the following is the result:

Character transmission time [ms] = 1000 × 11 bits ÷ baud rate

Example

Identifier for end of data request or end of response

Waiting period = 3.5 characters * 1000 * 11 bits ÷ baud rate

Baud rate [baud]	**Data format** [bit]	**Character transmission time**[ms]
38400	11	0.287
	10	0.260
19200	11	0.573
	10	0.521
9600	11	1.146
	10	1.042

6.1 Nameplate

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
0	0000	string[10]	5	r	Device name	ASCII encoding with "ZERO" as end identifier
5	0005	string[12]	6	r	Software version	ASCII encoding with "ZERO" as end identifier
11	000B	short	1	r	Software number	
12	000C	char	1	r	Sensor major version	
13	000D	char	1	r	Sensor minor version	
14	000E	short	1	r	Trial version	
15	000F	char	1	r	Hardware major version	
16	0010	char	1	r	Hardware minor version	
17	0011	string[14]	7	r	VDN Version	ASCII encoding with "ZERO" as end identifier
24	0018	short	1	r	VdN number	
25	0019	short	1	r	VDN version number	

6.2 Manufacturer's data

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
256	0100	char[20]	10	r	Serial number	ASCII encoding with "ZERO" as end identifier
266	010A	enum	1	r	Sensor type	Enumeration: 0 = - 1 = JUMO digiLine pH/ORP/T 2 = JUMO digiLine CR/Ci 3 = - 4 = JUMO digiLine O-DO S10 5 = -
267	010B	enum	1	r	Oxygen sensor sub-type	Enumeration: 0 = JUMO digiLine O-DO S10
273	0111	char[12]	6	r	Part no.	ASCII encoding with "ZERO" as end identifier
279	0117	char[64]	32	r	Order code	ASCII encoding with "ZERO" as end identifier
335	014F	Uint32	2	r	Hardware address	
347	015B	short	1	r	Date of manufacture, year	
348	015C	char	1	r	Date of manufacture, month	
349	015D	char	1	r	Date of manufacture, day	
350	015E	char	1	r	Date of manufacture, hour	
351	015F	char	1	r	Date of manufacture, minute	

6 Modbus address tables

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
352	0160	char	1	r	Date of manufacture, second	
353	0161	short	1	r	Date of manufacture, time zone	Offset for UTC in minutes
354	0162	Uint32	2	r	Cal. status	

6.3 Measuring point info

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
512	0200	string[60]	30	r/w	Tag number	UTF-8 encoding + "ZERO" as end identifier The number of bytes must be even (If necessary, append an additional ZERO character)
542	021E	string[192]	96	r/w	Description	UTF-8 encoding + "ZERO" as end identifier The number of bytes must be even (If necessary, append an additional ZERO character)
638	027E	string[110]	55	r/w	Sensor origin	This variable is managed automatically by the JUMO digiLine master and may not be described otherwise! UTF-8 encoding + "ZERO" as end identifier The number of bytes must be even (If necessary, append an additional ZERO character)

6.4 Configuration of the interface

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
768	0300	char	1	r/w	Device address	
769	0301	enum	1	r/w	Baud rate	Enumeration: 0 = 9600 1 = 19200 2 = 38400
770	0302	enum	1	r/w	Data format	Enumeration: 0 = 8 – 1 – no parity 1 = 8 – 1 – odd parity 2 = 8 – 1 – even parity 3 = 8 – 2 – no parity

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
773	0305	short	1	r/w	Min. response time	
775	0307	enum	1	r/w	Float value format	Enumeration: 0 = Standard Modbus format 1 = IEEE 754 Little Endian 2 = IEEE 754 Big Endian

6.5 Measurement configuration

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding/note
Dec.	Hex.					
9520	2530	enum	1	r/w	Oxygen measurement unit	0 = % vol 1 = % SAT 2 = - 3 = ppm 4 = mbar (oxygen partial pressure)
9600	2580	float	2	r/w	Compensation salt content (electrolytic conductivity in mS/cm)	Only effective with the unit "ppm"
9602	2582	float	2	r/w	Compensation pressure in hPa	Only effective with the units "% vol" or "% SAT"

6.6 Configuration of analog output

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding/note
Dec.	Hex.					
8769	2241	enum	1	r/w	Function	0 = Inactive 1 = Fixed value 2 = Standard signal 4 to 20 mA
8770	2242	float	2	r/w	Scaling start	Measured value at 4 mA
8772	2244	float	2	r/w	Scaling end	Measured value at 20 mA
8776	2248	float	2	r/w	Fixed value	Only if function = 1
8778	224A	char	1	r/w	Warning output active	Bool value for warning output true = active, false = inactive
8779	224B	char	1	r/w	Error output active	Bool value for error output true = active, false = inactive
8780	224C	float	2	r/w	Warning value	Output value from 3.5 mA to 21.5 mA for warning output
8782	224E	float	2	r/w	Error value	Output value from 3.5 mA to 21.5 mA for error output

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding/note
Dec.	Hex.					
8784	2250	float	2	r/w	Output value when temperature is outside the permissible range	Output value from 3.5 mA to 21.5 mA for temperature outside the permissible range

6.7 Operating data

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
1024	0400	Uint32	2	r	Operation hours counter	Operating time in seconds
1026	0402	short	1	r/w	Sensor replacement counter	

6.7.1 Initial startup

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
1280	0500	short	1	r	Year	
1281	0501	char	1	r	Month	
1282	0502	char	1	r	day	
1283	0503	char	1	r	Hour	
1284	0504	char	1	r	Minute	
1285	0505	char	1	r	Second	
1286	0506	char	1	r	Time zone of initial startup	
1287	0507	Uint32	2	r	Operations hour counter reading at initial startup	Operating time in seconds

6.7.2 Drag indicator function

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
1289	0509	float	2	r	Lowest temperature value	
1291	050B	float	2	r	Highest temperature value	

Date of lowest temperature value

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
1293	050D	short	1	r	Year	
1294	050E	char	1	r	Month	
1295	050F	char	1	r	day	
1296	0510	char	1	r	Hour	
1297	0511	char	1	r	Minute	
1298	0512	char	1	r	Second	

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
1299	0513	char	1	r	Time zone	
1300	0514	Uint32	2	r	Operating hours counter reading	Operating time in seconds

Date of highest temperature value

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
1302	0516	short	1	r	Year	
1303	0517	char	1	r	Month	
1304	0518	char	1	r	day	
1305	0519	char	1	r	Hour	
1306	051A	char	1	r	Minute	
1307	051B	char	1	r	Second	
1308	051C	char	1	r	Time zone	
1309	051D	Uint32	2	r	Operating hours counter reading	Operating time in seconds

6.8 Process values

6.8.1 Compensation

There are 2 sources for compensation values that can be used to calculate the measured value for the oxygen:

- **Flash storage:** Fixed compensation values that are stored in the non-volatile memory and are retained even if the device is powered off. Note that flash storage is not permitted to be written cyclically. The compensation data in the RAM is intended for this purpose.
- **RAM storage:** These compensation values can be written cyclically by the master device in order to obtain the latest compensation values from external sources (generally measurement data coming from other sensors on the master device). The "control byte", which controls which compensation source is selected, can also be found here.

The fixed compensation values from the flash storage are used as the initialization value for the compensation values in the RAM. However, they can also be used as pre-defined compensation values for the oxygen measurement.

The choice of whether to use the **flash** (fixed values) or **RAM** (measured values from external sources) to obtain the compensation data for the measurement must be controlled using the **control byte**.

Fixed compensation values in the flash storage

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
9600	2580	float	2	r/w	Fixed compensation salt content	
9602	2582	float	2	r/w	Fixed ambient pressure	

Compensation values in the RAM storage

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
9647	25AF	word	1	r/w	Control byte Controls which compensation source is selected	Low byte of the register comprising Boolean values; the meaning of the individual bits can be found in the following table
9648	25B0	float	2	r/w	Compensation temperature in °C	
9650	25B2	float	2	r/w	Compensation salt content	
9652	25B4	float	2	r/w	Ambient pressure	

Register content – control byte

Bit number	Meaning
0 (lsb)	Source for compensation temperature: False = internal temperature sensor True = RAM compensation temperature
1	Source for compensation salt content: False = fixed compensation salt content from flash True = RAM compensation salt content
2	Source for ambient pressure: False = fixed ambient pressure from flash True = RAM ambient pressure
3	Reserve
4 to 7 (msb)	Not relevant for Modbus (control of the cyclical transmission of compensation values by JUMO digiLine master devices)
8 to 15	-

6.8.2 Measured values

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
9728	2600	float	2	r	Measured oxygen value (unit as set in measurement configuration)	
9760	2620	float	2	r	Measured temperature value in °C	

6.8.3 Status word

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
9792	2640	word	1	r	Status word	16-bit word comprising Boolean values The meaning of the individual bits can be found in the following table

Register content – status word

Bit number	Meaning
0 (lsb)	True = warning regarding invalid measured oxygen value
1	True = warning regarding invalid measured temperature value
2	True = alarm regarding state of the sensor element; cap missing
3	Reserve
4	Reserve
5	Reserve
6	Reserve
7	Reserve
8	True = warning regarding oxygen outside the measuring range
9	True = warning regarding temperature outside the measuring range
10	True = warning regarding state of the sensor element; cap replacement is recommended
11	Reserve
12	Reserve
13	Reserve
14	Reserve
15 (msb)	Reserve

6.9 Date and time for time stamp

Modbus PDU address		Data type	Number of Modbus registers	Access	Data	Encoding
Dec.	Hex.					
10304	2840	short	1	r/w	Year	
10305	2841	char	1	r/w	Month	
10306	2842	char	1	r/w	day	
10307	2843	char	1	r/w	Hour	
10308	2844	char	1	r/w	Minute	
10309	2845	char	1	r/w	Second	
10310	2846	short	1	r/w	Time zone	Offset for UTC in minutes

JUMO

OneTemp pty ltd

measure | control | record

1300 768 887

www.onetemp.com.au