# Pro-16

# Pro-16 Industrial Controller

## Interface Description Manual

## MODBUS protocol


Manual Part number: 59553-1

March 2014

**Explanation of symbols:**

**General information**

**General warning**

**Caution: ESD-sensitive components**

MODBUS®
is a registered trademark of the MODBUS-IDA Organization

BluePort® and BlueControl®
are registered trademarks of PMA Prozeß- und Maschinen-Automation GmbH

© 2004-2005 PMA Prozeß- und Maschinen-Automation GmbH • Printed in Germany • All rights reserved •
Without prior written consent, reprinting or photocopying of this document, entirely or in part, is prohibited.

This is a publication of the following companies:
PMA Prozeß- und Maschinen Automation
P.O. Box 310229
D-34058 Kassel
Germany

West Control Solutions
The Hyde Business Park
Brighton
BN2 4JU
England

## Content

# 1 General

We thank you for purchasing a device from the ***Pro Series*** product range. This document describes the implementation and operation of the MODBUS interface used with the industrial controller Pro-16 which will be called 'device' in the rest of this document.

Devices with a MODBUS interface permit the transmission of process data, parameters, and configuration data. Electrical connections are made at the base of the device in the channel of the top-hat DIN rail. The serial communication interface provides a simple link to superordinate PLCs, visualization tools, etc.

An additional interface that is always fitted in the device's front panel is the BluePort® (PC) interface. This interface is not bussable, and serves for a direct connection with the BlueControl® software package that runs on a PC or laptop. Communication is done according to the master/slave principle. The device is always operated as a slave.

**The most important characteristics and physical/electrical properties of the bus connection are:**

- **Network topology**
  linear bus, possible with bus termination at both ends (see below).
- **Transmission media**
  screened and twisted 2-wire copper leads
- **Lead lengths (without repeater)**
  A maximum lead length of 1000 m should not be exceeded.
- **Transmission speeds**
  The following transmission speeds are supported:
  2400 … 38400 bits/s
- **Physical interface**
  RS 485 with bus connections in the top-hat rail; connections made on site.
- **Address range**
  1 ... 247
  (32 devices in one segment. Expandable to 247 with repeaters.)

## 1.1 References

Further information on the MODBUS-Protocol:

**[1] MODBUS Specifications**
- MODBUS application Protocol Specification V1,1
- MODBUS over serial line specification and implementation guide V1.1
- http://www.modbus.org

Further information on RS 485:
**[2] ANSI/TIA/EIA-485-A**

Additional documentation for Pro-16
devices:**[3] industrial controller Pro-16**
- Data sheet Pro-16 9498 737 40513
- Operating instructions Pro-16 59537

# 2 Commissioning the interface

Instrument field bus connection is via the pins of connector B on the rear, via flat-pin connectors or via screw terminals dependent on version.
Construction of suitable cables must be done by the user.

## 2.1 Mounting hints

If possible, the place of installation should be exempt of vibration, aggressive media (e.g. acid, lye), liquid, dust or aerosol.

**The unit may be operated only in environments for which it is suitable due to its protection type.**

**The housing ventilation slots must not be covered.**

**In plants where transient voltage peaks are susceptible to occur, the instruments must be equipped with additional protective filters or voltage limiters!**

**Caution! The instrument contains electrostatically sensitive components.**

**Please, follow the instructions given in the safety hints.**

## 2.2 Electrical connections

The electrical connection of the interface can be done as two-wire RS 485, as well as four-wire RS 485 (often called RS 422).

## 2.2.1 RS 485 version (two-wire )

The bus is build as RS 485 - two-wire cable with common ground main.
All the participants of an RS 485 bus are connected in parallel to the signals 'Data A' and 'Data B'.

The meaning of the data line terms are defined in the unit as follows:

- for signal 1 (off) Data A is positive to Data B
- for signal 0 (on) Data A is negative to Data B

**The terms Data A and Data B are reverse to A und B defined in [2] .**

For the purpose of limiting ground current loops, signal ground (GND) can be grounded at one point via a resistor 'RGND' (100 ohms, ¼ watt).

Association of terms for the two-wire-MODBUS definition according to [1]:

| Definition MODBUS | according to unit |
|---|---|
| D1 | Data A |
| D0 | Data B |
| Common | RGND |

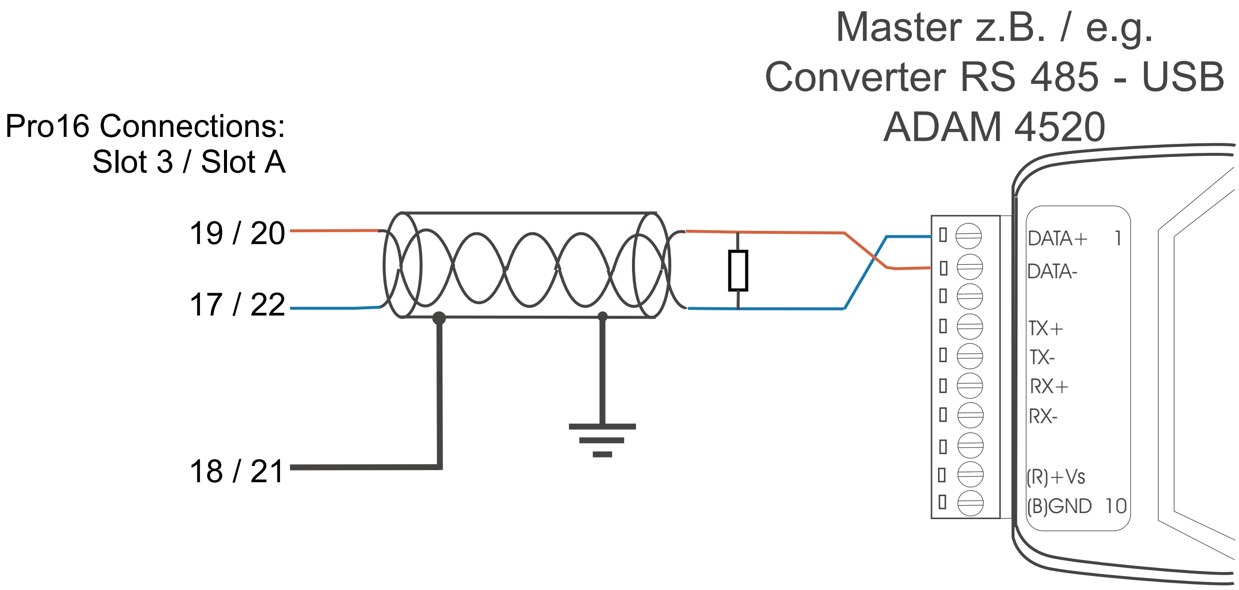
Notes:

❶ Terminating resistors between Data A and B at the cable ends (see figure 2 on page 8)
❷ Screening (see figure 2 on page 8)
❸ GND lead (see Fig. 6)

| Pro-16 | |
|---|---|
| Signal | Terminal |
| Option 3 position | |
| TXD-B | 19 |
| TXD-A | 17 |
| GND | 18 |
| Option A position | |
| TXD-B | 20 |
| TXD-A | 22 |
| GND | 21 |

The following cable connection methods are possible.

*Fig. 2 connection example RS 485*



This page has been intentionally left blank

This page has been intentionally left blank

## 2.2.3 Cable installation

Depending on each application, suitable cables are to be used for the bus. When installing the cables, all relevant regulations and safety codes (e.g. VDE 0100) must be observed:

- Cable runs inside buildings (inside and outside of control cabinets)
- Cable runs outside buildings
- Potential balancing conductors
- Screening of cables
- Measures against electrical interference
- Length of spur lines

In particular, the following points must be considered:

- The RS 485 bus technology used here permits up to 32 devices in a segment to be connected to one bus cable. Several segments can be coupled by means of repeaters.
- The bus topology is to be designed as a line with up to 1000 m length per segment. Extensions by means of repeaters are permitted.
- The bus cable is to be taken from device to device (daisy chaining), i.e. not star connected**.**
- If possible, spur lines should be avoided, in order to prevent reflections and the associated disturbances in communication.
- The general notes on interference-free wiring of signal and bus leads are to be observed (see Operating notes "EMC – General information' (9407 047 09118)).
- To increase signal transmission reliability, we recommend using screened, twisted pairs for the bus leads.

## 2.2.4 Screening

The type of screening is determined primarily by the nature of the expected interference.

- For the suppression of electrical fields, one end of the screened cable must be grounded. This should always be done as the first measure.
- Interference due to alternating magnetic fields can only be suppressed, if the screened cable is grounded at both ends. However, this can lead to ground current earth loops: galvanic disturbance along the reference potential lead can interfere with the useful signal, and the screening effect is reduced.
- If several devices are linked to a single bus, the screen must be connected at each device, e.g. by means of screen clamps.
- The bus screen must be connected to a central PE point, using short, low-impedance connections with a large surface, e.g. by means of screen clamps.

## 2.2.5 Terminating resistors

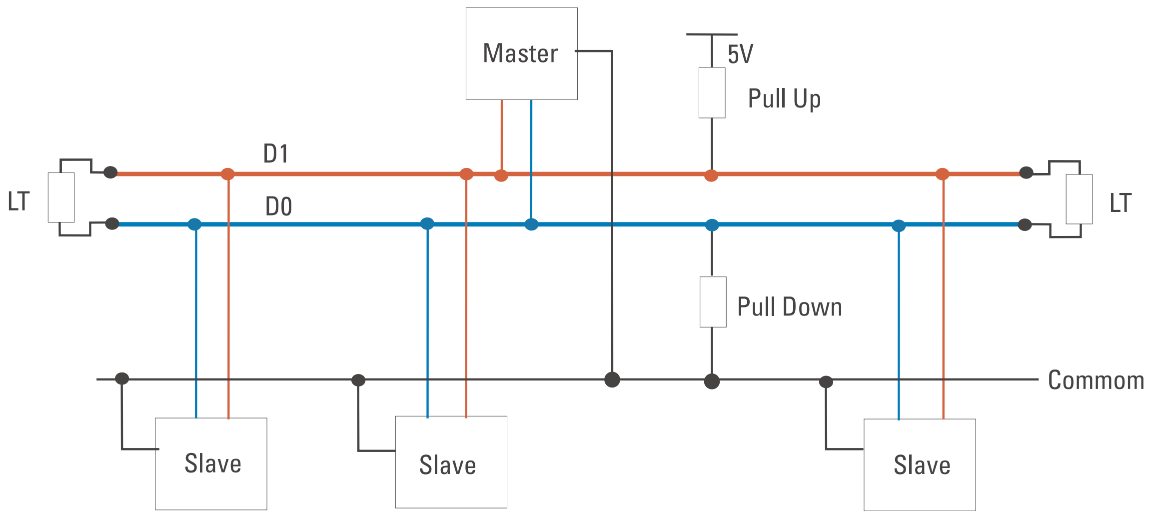
The widespread US Standard EIA RS 485 recommends fitting terminating resistors at each end of the bus cable. Terminating resistors usually have a value of approx. 120 ohms, and are connected in parallel between the data lines A and B (depending on the cable impedance; for details, see the cable manufacturer's data sheet). Their purpose is to eliminate reflections at the end of the leads, thus obtaining a good transmission quality. Termination becomes more important, the higher the transmission speed is, and the longer the bus leads are.

However, if no signals are applied to the bus, it must be ensured that the signal levels are clearly defined. This done by means of pull-up and pull-down resistors between +5V or GND, and the drivers. Together with the bus terminating resistor, this forms a voltage divider. Moreover, it must be ensured that there is a voltage difference of at least ±200mV between the data lines A and B, as seen by the receiver.

ⓘ Normally, an external voltage source is provided.

Fig. 6 shows the device connections as recommended by the MODBUS User Organization [1].

*Fig. 3 Recommended connections*




**With four-wire connection (RS 422), each wire pair corresponds to the drawing above.**

**If no external voltage source is available, and if there are only a few participants on the bus (e.g. only a master and a slave device), and the transmission speed is low (e.g. 9600 bits/s), the lead lengths are short, and terminating resistors have been fitted, it is possible that the minimum signal level cannot be reached. This will cause disturbances in signal transmission.**

**Therefore, if only a few PMA devices are connected, we recommend the following procedure before fitting terminating resistors:**

| Baudrate | Lead length | No. of PMA devices | Terminating resistor |
|---|---|---|---|
| ≤ 9600 Bist/s | ≤ 1000 m | < 8 | no |
| 19200 Bit/s | ≤ 500 m | < 8 | no |
| 38400 Bit/s | ≤ 250 m | < 8 | no |
| beliebig | | ≥ 8 | useful |
| | | | other cases: try out |

**If less than 8 PMA devices are connected to a bus with the above maximum lead lengths, no terminating resistors should be fitted.**

**Note: If additional devices from other manufacturers are connected to the bus, no general recommendations are possible – this means: trial and error!**

### 2.2.6 Installation notes

- Measurement and data leads should be kept separate from control leads and power cables.
- Twisted and screened cables should be used to connect sensor. The screen must be grounded.
- Connected contactors, relays, motors, etc. should be fitted with RC snubber circuits in accordance with manufacturer specifications.
- The device must not be installed near powerful electrical or electromagnetic fields.

**- The device is not certified for installation in explosion-hazarded areas.**
**- Incorrect electrical connections can result in severe damage to the device.**
**- Please observe all safety instructions.**

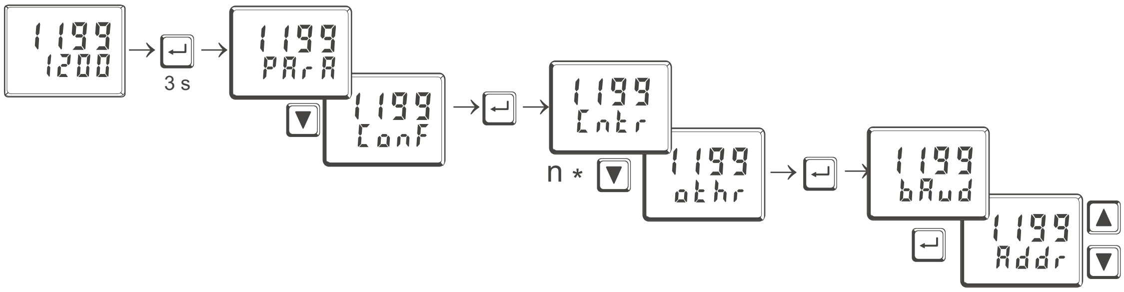
## 2.3 Bus settings

### 2.3.1 Bus address

The participant address of a device connected to a bus must be adjusted by one of the following means:

- the Engineering Tool BlueControl® using the menu item Othr/Addr
- or via the device's front panel (see below)

*Fig. 4 Setting a bus address*




**Every device connected to a bus must have a different, unique address.**

**Please regard: When allocating the unit's addresses don't give the same address to two units. In this case a strange behaviour of the whole bus becomes possible and the busmaster will not be able to communicate with the connected slave-units.**

### 2.3.2 Transmission parameters

**The transmission parameters of all devices linked to a bus must have the same settings.**

**Baudrate** (bAud)
The baudrate is the measure of data transmission speed. The devices support the following transmission speeds:

- 38000 bits/s
- 19200 bits/s
- 9600 bits/s
- 4800 bits/s
- **2400 bits/s**

**Parity / Stop bit (PrtY)**
The parity bit is used to check whether an individual fault has occurred within a byte during transmission.

The device supports:

- **even parity**
- **odd parity**
- **no parity**

With even parity, the parity bit is adjusted so that the sum of the set bits in the 8 data bits and the parity bit result in an even number. Conversely, the same applies for uneven parity.

**If a parity error is detected upon receipt of a message, the receiving device will not generate an answer.**

Other parameters are:

- 8 data bits
- 1 start bit
- 1 stop bit
  1 or 2 stop bits can be selected when adjusting 'no parity'.

**The max. length of a message may not exceed 256 bytes.**

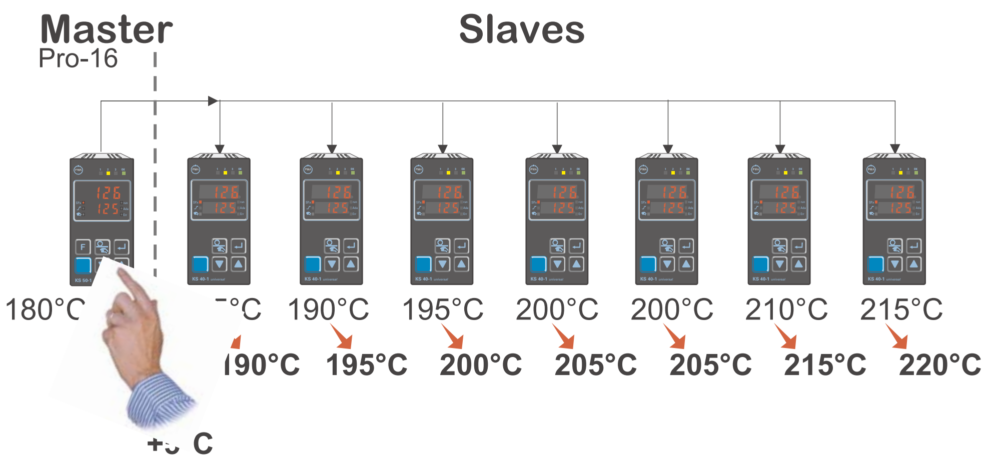
## 2.4 Master operation (MASt)

The Pro-16 master function is limited to broadcast messages (data transmission to all connected slaves).
For operation as a master, the instrument must be configured accordingly by means of BlueControl® (engineering software for Pro-16).

*Fig. 5 : Master function parameter setting*


A possible MODBUS master configuration is given in the drawing shown above. In this example, the actual master set-point (source address 3170) is transmitted to the slaves (target address 3180) at intervals of 5 seconds.

*Fig. 6 : Example*



## 2.5 System layout

**Please observe the guidelines and notes provided by the manufacturer of the master device regarding the layout of a communication system.**

### 2.5.1 Minimum configuration of a MODBUS installation

A MODBUS installation consists of not less than the following components:

- a bus master, which controls the data traffic
- one or more slave participants, which provide data upon demand by the master
- the transmission media, consisting of the bus cable and bus connectors to link the individual participants, plus a bus segment (or several, which are connected by means of repeaters).

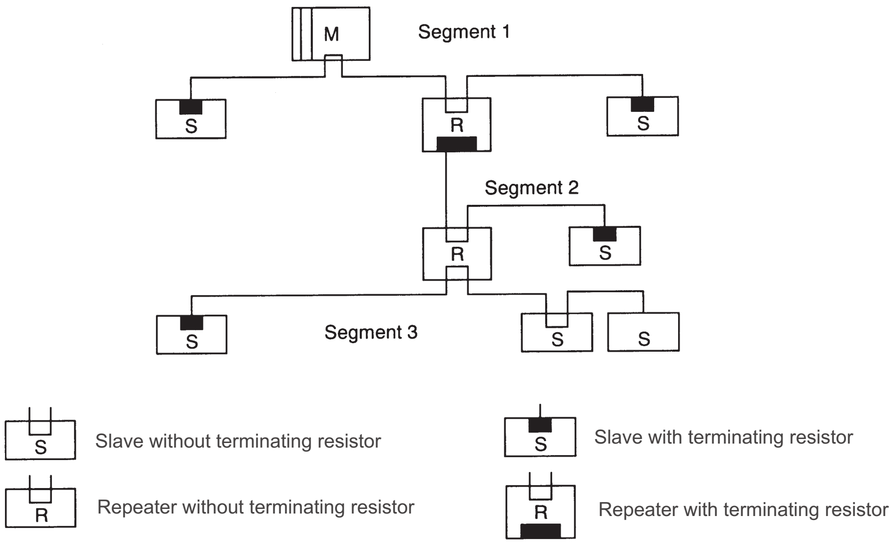
### 2.5.2 Maximum configuration of a MODBUS installation

A bus segment consists of max. 32 field units (active and passive). The greatest number of slave participants that can be operated by one MODBUS master via several segments, is determined by the internal memory structure of the master. Therefore, you should know the specifications of the master when planning a MODBUS installation.
The bus cable can be opened at any point in order to add another participant by means of a bus connector. At the end of a segment, the bus cable can be extended up to the total permissible length for a segment. The permissible length of a bus segment depends on the selected transmission speed, which in turn is determined mainly by plant layout (length of each segment, distributed inputs/outputs) and the required scan cycles for individual participants. All participants connected to the bus must be configured for the same transmission speed (bit rate).

**MODBUS devices must be connected in a line structure.**

If more than 32 participants are required, or larger distances than the permissible length of one segment are needed, the MODBUS installation can be extended by means of repeaters.

*Fig. 7 structure*

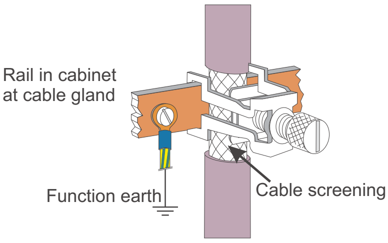



A fully configured MODBUS installation may contain max. 247 participants with the address range 1...247. Every installed repeater reduces the max. number of participants with a segment. Repeaters are passive participants and do not require a MODBUS address. However, its input circuit represents an additional load in the segment due to the current consumption of the bus driver. Nonetheless, a repeater has no influence on the total number of participants connected to the bus. The maximum number of series-connected repeaters can differ, depending on the manufacturer. Therefore, you should ask the manufacturer about possible limitations when planning a MODBUS installation.

## 2.5.3 Wiring inside buildings

The following wiring hints apply for twisted-pair cables with screen. The cable screen serves to improve overall electromagnetic compatibility.

*Fig. 8 Screen connection*




Depending on requirements, the one or both ends of the cable screen must be connected to a central earth point (PE) by means of low-impedance connections with a large surface, e.g. screen clamps. When installing a repeater or field unit in a control cabinet, the cable screen should be connected to an earth rail mounted as close as possible to the cable entry into the cabinet.

The screen must be taken right up to the field unit, where it is to be connected to the conductive housing and/or the metal connector. Hereby, it must be ensured that the device housing (and possibly the control cabinet in which the device is installed), are held at equal ground potential by means of low-impedance connections with a large surface. Connecting a screen to a lacquered or painted surface is useless. By observing these measures, high-frequency interference will be grounded reliably via the cable screens. Should external interference voltages still reach the data lines, the voltage potential will be raised symmetrically on both lines, so that in general, no destructive voltage differences can arise. Normally, a shift of the ground potential by several volts will not have an effect on reliable data transmission. If higher voltages are to be expected, a potential balancing conductor with a minimum cross-section of 10 $mm^2$ should be installed parallel to the bus cable, with connections to the reference ground of every field unit. In case of extreme interference, the bus cable can be installed in a metal conduit or channel. The conduit tube or the channel must be earthed at regular distances.

The bus cable must always be installed with a minimum separation of 20 cm from other cables carrying voltages above 60 V. Similarly, the bus cable must be run separately from telephone lines, as well as from cables leading into explosion-hazarded areas. In these cases, we recommend installing the bus cable in a separate cable tray or channel.

Cable trays or channels should always be made of conductive materials, and must be earthed at regular distances. Bus cables should not be subjected to any mechanical strains or obvious risks of damage. If this cannot be ensured, suitable measures must be undertaken, such as installation in conduit.

**Floating installation:**
If the installation must be floating (no earth connection) for certain reasons, the device reference ground must only have a high-impedance connection to earth (e.g. an RC combination). The system will then find its own earth potential. When connecting repeaters for the purpose of linking two bus segments, a floating installation is recommended, to prevent possible potential differences being transferred from one segment to the next.

# 3 Bus protocol

## 3.1 Composition of a transmission byte

Originally, the MODBUS protocol was defined for the communication between a supervisory system and the Modicon® PLC. It used a master/slave structure, in which only one device (master) is able to initiate data transactions (queries). The query message from the master is answered (response) by other devices (slaves), which supply the requested data. Moreover, the master can address a specific slave via its MODBUS address, or address all connected slaves by means of a general message (broadcast).
The MODBUS protocol determines the transmission formats for the query and the response. Function codes define the actions to be executed by the slaves.
Within the device, the MODBUS protocol uses the RTU (remote terminal unit) mode, i.e. every transmitted byte of a message contains two hexadecimal characters (0...9, A...F).

The composition of a byte in the RTU-protocol is as follows:

| Start bit | 8 data bits | Parity/Stop bit | Stop bit |
|---|---|---|---|

## 3.2 General message frame

The message is read into a data buffer with a defined maximum length. Longer messages are not accepted, i.e. the device does not answer.

The message consist of the following elements:

| Device address | Function code | Data field | CRC | End of frame detection |
|---|---|---|---|---|
| 1 byte | 1 byte | N * 1 bytes | 2 bytes | |

- Device address (Addr)
  The device address is used for identification. Device addresses can be assigned in the range of 1...127.
  The device address ‘0’ is reserved for ‘Broadcast’ messages to all slaves. A broadcast message can be transmitted e.g. with a write instruction that is then executed by all the slaves on the bus. Because all the slaves execute the instruction, no response messages are generated.
- Function code
  The function code defines the transaction type in a message. The MODBUS specification defines more than 17 different function codes. Supported codes are described in Section 3.6. „Function codes“.
- Data field
  The data field contains the detailed specifications of the transaction defined by the function code. The length of the data field depends on the function code.
- CRC
  As a further means of fault detection (in addition to parity bit detection) a 16-bit cyclical redundancy check (CRC) is performed. The CRC code ensures that communication errors are detected. For additional information, see Section 3.2.1. “CRC”.
- End of frame detection
  The end of a message is defined by a period of 3,5 characters, during which no data transfer occurs. For additional information, see Section 3.2.2. „End of frame detection“

Further information is given in the documents named in **[1]** or under http://www.modbus.org.

### 3.2.1 CRC

The CRC is a 16-bit value that is attached to the message. It serves to determine whether a transmitted message has been received without errors. Together with the parity check, this should detect all possible communication errors.

If a parity fault is detected during reading, no response message will be generated.

The algorithm for generating a CRC is as follows:

① Load CRC register with FFFFhex.
② Exclusive OR the first transmit/receive byte with the low-order byte of the CRC register, putting the result into the CRC register, zero-filling the MSB.
③ Shift the CRC register one bit to the right.
④ If the expelled bit is a '0' repeat step 3.
If the expelled bit is a ‘1’, exclusive OR the CRC register with value A001hex.
⑤ Repeat steps 3 and 4 for the other 7 data bits.
⑥ Repeat steps 2 to 5 for all further transmit/receive bytes.
⑦ Attach the result of the CRC register to the message (low-order byte first, then the high-order byte).
When checking a received message, the CRC register will return ‘0’, when the message including the CRC is processed.

### 3.2.2 End of frame detection

The end of a message (frame) is defined as a silence period of 3.5 characters on the MODBUS.
A slave may not start its response, and a master may not start a new transmission before this time has elapsed.

However, the evaluation of a message may begin, if a silence period of more than 1.5 characters occurs on the MODBUS. But the response may not start before 3,5 characters of silence.

## 3.3 Transmission principles

Two transmission modes are used with MODBUS:

- **Unicast mode**
- **Broadcast mode**

In the Unicast mode, the master addresses an individual device, which processes the received message and generates a response. The device address can be 1...247. Messages always consist of a query (request) and an answer (response). If no response is read within a defined time, a timeout error is generated.

In the Broadcast mode, the master sends a write instruction (request) to all participants on the bus, but no responses are generated. The address ‘0’ is reserved for broadcast messages.

## 3.4 Response delay (dELY)

Some devices require a certain period to switch from transmit to receive. The adjusted delay is added to the silent period of 3,5 characters at the end of a message, before a response is generated. The delay is set in ms.

## 3.5 Modem operation (C.dEL)

The end of frame detection of a received MODBUS message can be increased by the period ‘C.del’. This time is needed e.g. for transmission via a modem, if messages cannot be transmitted continuously (synchronous operation). The delay is set in ms.

## 3.6 Function codes

Function codes serve to execute instructions. The device supports the following function codes:

| Function code hex | dez | Description | Explanation |
|---|---|---|---|
| 0x03 | 3 | Read Holding (Output) Register | Reading of process data, parameters, and configuration data |
| 0x04 | 4 | Read Input Register | Reading of process data, parameters, and configuration data |
| 0x06 | 6 | Preset Single Register (Output) | Wordwise writing of a value (process value, parameter, or configuration data) |
| 0x08 | 8 | Diagnostics | Reading the MODBUS diagnostic register |
| 0x10 | 16 | Preset Multiple Register (Output) | Wordwise writing of several values (process data, parameter or configuration data) |

The behaviour of function codes 3 and 4 is identical.
The following sections show various examples of message composition.

### 3.6.1 Reading several values

Messages with function codes 3 or 4 are used for (wordwise) reading of process data, parameters or configuration data. For reading 'Float' type data, 2 values must be requested for each datum.

The composition of a read message is as follows:
Request:

| Field name | Value (hex) | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 03 or 04 | Reading process data, parameters or configuration data |
| Start address High<br>Start address Low | 02<br>8A | Starting address 650 |
| No. of values | 00<br>02 | 2 datums (2 words) |
| CRC | CRC-Byte1<br>CRC-Byte2 | |

Response:

| Field name | Value (hex) | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 03 oder 04 | Reading process data, parameters or configuration data |
| No. of bytes | 04 | 4 data bytes are transmitted |
| Word 1 | 00<br>DE | Process data, parameters or configuration data.<br>Address 650= 222 |
| Word 2 | 01<br>4D | Process data, parameters or configuration data.<br>Address 651= 333 |
| CRC | CRC-byte1<br>CRC-byte2 | |

**A broadcast message is not possible for function codes 3 and 4.**

**If the first addressed value is not defined, an error message "ILLEGAL DATA ADDRESS" is generated. If no further data are defined in the areas to be read following the first value, these areas will be entered with the value "NOT DEFINED VALUE". This enables areas with gaps to be to be read in a message.**

### 3.6.2 Writing a single value

Messages with function code 6 are used for (wordwise) writing of process data, parameters or configuration data as integers. This function is not suitable for writing 'Float' type data.

The composition of a write message is as follows:
Request:

| Field name | Value (hex) | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 06 | Writing a single value (process data, parameter or configuration) |
| Write address High<br>Write address Low | 02<br>8A | Write address 650 |
| Value | 00<br>7B | Preset value = 123 |
| CRC | CRC-byte1<br>CRC-byte2 | |

Response:

| Field name | Value (hex) | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 06 | Writing a single datum (process data, parameter or configuration) |
| Write address High<br>Write address Low | 02<br>8A | Write address 650 |
| Value | 00<br>7B | Preset value = 123 |
| CRC | CRC-Byte1<br>CRC-Byte2 | |

If everything is correct, the response message corresponds exactly to the default.

**The devices can also receive this message as a broadcast with the address '0'.**

**A default value in the 'Real' data format is not possible, as only 2 bytes can be transmitted as value.**

**If a value is outside the adjustable range, the error message "ILLEGAL DATA VALUE" is generated. The datum remains unchanged. Also if the datum cannot be written (e.g. configuration data, and the device is online), an error message "ILLEGAL DATA VALUE" is generated.**

## 3.7 Writing several values

Messages with function code 16 are used for (wordwise) writing of process data, parameters or configuration data. For writing 'Float' type data, 2 values must be transmitted for each datum.

The composition of a write message is as follows:
Request:

| Field name | Value (hex) | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 10 | Writing several process values, parameters or configuration data |
| Start address High<br>Start address Low | 02<br>8A | Write address 650 |
| No. of values | 00<br>02 | 2 values |
| No. of bytes | 04 | 4 data bytes are transmitted |
| Word 1 | 00<br>DE | Process value, parameters or configuration data.<br>Address 650 = 222 |
| Word 2 | 01<br>4D | Process value, parameters or configuration data.<br>Address 651 = 333 |
| CRC | CRC byte1<br>CRC byte2 | |

Response:

| Field name | Value (hex) | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 10 | Writing several process values, parameters or configuration data |
| Start address High<br>Start address Low | 02<br>8A | Write address 650 |
| No. of values | 00<br>02 | 2 process values, parameters or configuration data |
| CRC | CRC byte1<br>CRC byte2 | |

**The devices can also receive this message as a broadcast with the address '0'.**

**If the first value is not defined, an error message "ILLEGAL DATA ADDRESS" is generated.**
**If the first value cannot be written (e.g. configuration data, and the device is online), an error message "ILLEGAL DATA VALUE" is generated.**

If no further data are defined or cannot be written in the specified areas following the first value, these areas will be skipped. The data in these locations remains unchanged. This enables areas with gaps, or that are currently not writable, to be changed with a message. No error message is generated.

If a value is outside the adjustable range, the error message "ILLEGAL DATA VALUE" is generated. Subsequent data are not evaluated. Previously accepted correct data are active.

## 3.8 Error record

An error record is generated, if a message is received correctly, but message interpretation or the modification of a datum is not possible.

**If a transmission error is detected, no response is generated. The master must retransmit the message.**

Detected transmission errors are:

- Parity fault
- Framing error (no stop bit received)
- Overrun error (receiving buffer has overflowed or data could not be retrieved quickly enough from the UART)
- CRC error

The composition of the error record is as follows:

| Field name | Value | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 90 | Error record for the message 'Writing several parameters or configuration data'.<br>Composition: $80_{hex}$ + function code |
| Error code | 02 | ILLEGAL DATA ADDRESS |
| CRC | CRC byte1<br>CRC byte2 | |

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

### 3.8.1 Error codes

The following error codes are defined:

| Code | Name | Explanation |
|---|---|---|
| 01 | ILLEGAL FUNCTION | The received function code is not defined in the device. |
| 02 | ILLEGAL DATA ADDRESS | The received address is not defined in the device, or the value may not be written (read only).<br>If several data are read simultaneously (function codes 01, 03, 04) or written simultaneously (function codes 0F, 10), this error is only generated if the first datum is not defined. |
| 03 | ILLEGAL DATA VALUE | The received value is outside the adjusted limits or it cannot be written at present (device is not in the configuration mode).<br>If several data are written simultaneously (function codes 0F, 10), this error is only generated if the first datum cannot be written. |
| 04 | SLAVE DEVICE FAILURE | More values are requested than permitted by the transmission buffer. |

Other error codes specified in the MODBUS protocol are not supported.

## 3.9 Diagnosis

By means of the diagnosis message, the device can be prompted to send check messages, go into operational states, output counter values or to reset the counters.
This message can never be sent as a broadcast message.

The following functions have been defined:

| Code | Explanation |
|---|---|
| 0x00 | Return transmission of the received message |
| 0x01 | Restart of communication (terminates the Listen Only mode) |
| 0x02 | Return transmission of the diagnosis register |
| 0x04 | Change to the Listen Only mode |
| 0x0A | Delete the counter and reset the diagnosis register |
| 0x0B | Return transmission of the message counter (all messages on the bus) |
| 0x0C | Reset of the counter for faulty message transmissions to this slave (parity or CRC error) |
| 0x0D | Return transmission of the counter for messages answered with error code |
| 0x0E | Return transmission of the message counter for this slave |
| 0x0F | Return transmission of the counter for unanswered messages |
| 0x10 | Return transmission of the counter for messages answered with NAK |
| 0x11 | Return transmission of the counter for messages answered with Busy |
| 0x12 | Return transmission of the counter for too long messages |
| 0x40 | Return transmission of the parity error counter |
| 0x41 | Return transmission of the framing error counter (stop bit not detected) |
| 0x42 | Return transmission of the counter for full buffer (message longer than receiving buffer) |

- ❑ Request in the Integer format:
  If the setting for Integer with decimals (most significant 3 bits) is used for the address, the counter contents will be transmitted in accordance with the necessary conversion factor.

- ❑ Request in the Float format:
  If the setting for Float (most significant 3 bits are 010) is used for the address, the counter contents will be transmitted in the IEEE format. The largest value is 65535, because the counters in the device are designed as word counters.
  In the Float format, a 4-byte data field is returned with a request for counter contents. In all other cases, a 2-byte data field is returned.

When switching into the Listen mode (0x04) and at restart after the device has changed into the Listen mode, no response is generated.
If a restart diagnosis message is received while the device is not in the Listen mode, the device generates a response.

A diagnosis message is composed as follows:

Request:

| Field name | Value | Explanation |
|---|---|---|
| Address | 11 | Address 17 |
| Function | 08 | Diagnosis message |
| Sub-function High<br>Sub-function Low | 00<br>YY | Sub-function code |
| Data field | Byte 1<br>Byte 2 | Further data definitions |
| CRC | CRC byte1<br>CRC byte2 | |

### 3.9.1 Return transmission of the received message (0x00)

The message serves as a check whether communication is operational.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 00 | 2 bytes of any content | Return transmission of the received datum |

### 3.9.2 Restart of communication (terminates the Listen Only mode) (0x01)

The slave is instructed to initialize its interface, and to delete the event counters. In addition, the device is instructed to exit the Listen Only mode. If the device already is in the Listen Only mode, no response is generated.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 01 | 00 00 | 00 00 |

### 3.9.3 Return transmission of the diagnosis register (0x02)

The slave sends its 16-bit diagnosis register to the master. The data contained in this register are freely definable. For example, the information could be: EEPROM faulty, LED defective, etc.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 02 | 00 00 | Contents of the diagnosis register |

### 3.9.4 Change to the Listen Only mode (0x04)

The slave is instructed not to execute or answer any messages addressed to it. The device can only return to normal operation by means of the diagnosis message ‘Sub-function 00 01’ or by means of a new power up.

The function serves to disable a module that is behaving erratically on the MODBUS, so that the bus can continue operations. The device does not generate a response after receiving this message.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 04 | 00 00 | No response |

### 3.9.5 Delete the counter and reset the diagnosis register (0x0A)

The slave is instructed to delete the contents of its event counter and to reset the diagnosis register.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 0A | 00 00 | 00 00 |

### 3.9.6 Return transmission of the message counter (0x0B)

The slave is instructed to return the value of its message counter.
The counter contains the sum of all messages, which the slave has recorded on the bus. This count includes all the messages transmitted by the master and the other slaves. The count does not include the response messages of this slave.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 0B | 00 00 | Message counter |

### 3.9.7 Return transmission of the counter for faulty message transmissions

The slave is instructed to return the value of its counter for faulty message transmissions.
The counter contains the sum of all messages addressed to the slave, in which an error was detected. Hereby, the faults can be CRC or parity errors.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 0C | 00 00 | Contents of counter for faulty message transmissions |

### 3.9.8 Return transmission of the counter for messages with error code

The slave is instructed to return the value of its counter for the messages answered with error code. The counter contains the sum of all messages addressed to the slave, and which were answered with an error code.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 0D | 00 00 | Contents of counter for messages answered with an error code |

### 3.9.9 Return transmission of the message counter for this slave

The slave is instructed to return the value of its counter for messages to this slave.
The counter contains the sum of all messages addressed to the slave.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 0E | 00 00 | Contents of counter for messages addressed to this slave |

### 3.9.10 Return transmission of the counter for unanswered messages

The slave is instructed to return the value of its counter for unanswered messages.
The counter contains the sum of all messages addressed to the slave, which were not answered because of internal events or detected errors.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 0F | 00 00 | Contents of counter for unanswered messages |

## 3.9.11 Return transmission of the counter for messages answered with NAK

The slave is instructed to return the value of its counter for messages answered with NAK.
The counter contains the sum of all messages addressed to the slave, which were answered with NAK.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 10 | 00 00 | Contents of counter for messages answered with NAK |

## 3.9.12 Return transmission of the counter for messages answered with Busy

The slave is instructed to return the value of its counter for messages answered with Busy.
The counter contains the sum of all messages addressed to the slave, which were answered with Busy.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 12 | 00 00 | Contents of counter for messages answered with Busy |

## 3.9.13 Return transmission of the parity error counter

The slave is instructed to return the value of its counter for parity errors.
The counter contains the sum of all messages addressed to the slave, in which a parity error was detected.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 40 | 00 00 | Contents of counter for the number of parity errors |

## 3.9.14 Return transmission of the framing error counter

The slave is instructed to return the value of its counter for the number of framing errors.
The counter contains the sum of all messages addressed to the slave, in which a framing error was detected. A framing error occurs, if the stop bit at the end of a byte is not detected.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 41 | 00 00 | Contents of counter for the number of framing errors |

## 3.9.15 Return transmission of the counter for too long messages

The slave is instructed to return the value of its counter for too long messages.
The counter contains the sum of all messages addressed to the slave, which caused an overflow of the receiving buffer, or if the data were not retrieved from the UART quickly enough.
Definition of the received and returned data:

| Sub-function | Received data field | Transmitted data field |
|---|---|---|
| 00 42 | 00 00 | Counter for too long messages |

# 4 MODBUS addresses, address areas, and address formats

## 4.1 Area definitions

The address is coded in 2 bytes. The most significant 3 bits determine the data transmission format.
The following formats are available for *rail line* devices:

- **Integer**
- **Integer with 1 decimal**
- **(Float acc. to IEEE)**

| Address area | | Data transfer format | Smallest transferable value | Largest transferable value | Resolution |
|---|---|---|---|---|---|
| hex | dez. | | | | |
| 0x0000 ... 0x1FFF | 0 ... 8191 | Integer without decimals | -30000 | +32000 | +/- 1 |
| 0x2000 ... 0x3FFF | 8192 ... 16383 | Integer with 1 decimal | -3000.0 | +3200.0 | +/- 0.1 |
| 0x4000 ... 0x7FFF | 16384...32767 | Float (IEEE format) | -1.0 E+037 | +1.0 E+037 | +/-1.4E-045 |

**For integer numbers with and without decimals, the value range -30000 to +32000 is transmitted via the interface. Scaling with the factor 1 or 10 must be carried out by the transmitting device as well as by the receiving device.**

**- Values are transmitted in the Motorola format (big endian).**
**- The relevant areas are grouped for process data, parameter and configuration data reading and writing.**
**- Multiple definition of process data in different groups is possible.**

## 4.2 Special values

The following special values are defined for transmission in the integer format:

- -31000 Sensor fault
  This value is returned for data that do not represent a meaningful value due to a sensor fault.

- -32000 Switch-off value
  The function is disabled.

- -32500 Undefined value
  The device returns this value, if a datum is not defined within the requested range („NOT DEFINED VALUE").

- -32768 Corresponds to 0x8000 hex.
  The value to be transmitted lies outside the transferable integer value range.

The following special values are defined for transmission in the Float format:

- -1.5E37 This datum is not defined.
  The device returns this value, if a datum is not defined within the requested range.

## 4.3 Composition of the address tables

In the address tables shown in Section 5, the addresses for every parameter of the corresponding data format are specified in decimal values.
The tables are structured as follows:

| Name | R/W | Address | Integer | Real | Type | Value/off | Description |
|---|---|---|---|---|---|---|---|
| | | base<br>1dP | | | | | |

- Name — Description of the datum
- R/W — permitted type of access: R = read, W = write
- Address integer — Address for integer values
- base — Integer without decimals
- 1 dP — Integer with 1 decimal
- Real — Floating point number / Float (IEEE format)
- Type — internal data type
- Value/off — permissible value range, switch-off value available
- Description — Explanations

## 4.4 Internal data types

The following data types are assigned to data used in the device:

- Float
  Floating point number
  Value range: -1999 ... -0.001, 0, 0.001 ... 9999
- INT
  Positive whole integer number
  Value range: 0 ... 65535
  Exception: Switch-off value '-32000'
- Text
  Text string consisting of n characters, currently defined n = 5
  Permissible characters: 20H...7FH
- Long
  Positive whole Long number
  Value range: 0 … 99999
- Enum
  Selection value

# 5 Index

Index

# 6 Address tables

The following sections describe the address tables for:

- **industrial  controller Pro-16**

## Table Of Contents

## Table Of Contents

## 1 Cntr

### ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP.Fn | r/w | base | 3150 | 39068 | Enum | *Enum_SPFN* | Basic configuration for setpoint processing, e.g. 'setpoint controller switchable to external setpoint'. Configuration of special, controller-dependent setpoint functions. |
| | | 1dP | 11342 | | | | |
| | | 2dP | 19534 | | | | |
| | | 3dP | 27726 | | | | |

| Value | Description |
|---|---|
| 0 | Setpoint controller can be switched over to external setpoint (->LOGI/SP.E) |
| 1 | Program controller for setpoint profile. The program profile is definable by the user. |
| 2 | Timer, operating mode 1 (bandwidth monitoring, switch-off at the end). After timer start, the controller lines out at the defined setpoint. The timer time (t.SP) runs when the process value enters the adjusted band around the setpoint (x = SP ± b.ti). When the timer has elapsed, the controller is switched to Y2 (= fixed positioning value) and the lower display alternates between 'End' and the setpoint. |
| 3 | Timer, operating mode 2 (bandwidth monitoring, pause at the end). After timer start, the controller lines out at the defined setpoint. The timer time ( t.SP) runs when the process value enters the adjusted band around the setpoint (x = SP ± b.ti). When the timer has elapsed, the controller continues with setpoint SP, and the lower display alternates between 'End' and the setpoint. |
| 4 | Timer, operating mode 3 (switch-off at the end). After timer start, the controller lines out at the defined setpoint. The timer time (t.SP) runs immediately after switch-over. When the timer has elapsed, the controller is switched to Y2 (= fixed positioning value) and the lower display alternates between 'End' and the setpoint. |
| 5 | Timer, operating mode 4 (pause at the end). After timer start, the controller lines out at the defined setpoint. The timer time (t.SP) runs immediately after switch-over. When the timer has elapsed, the controller continues with setpoint SP, and the lower display alternates between 'End' and the setpoint. |
| 6 | Timer, operating mode 5 (delayed start). The timer starts immediately. The controller continues with Y2 (= fixes positioning value). When the timer (t.SP) has elapsed, the controller switches over to the adjusted setpoint. |
| 7 | Timer, operating mode 6 (setpoint switch-over). After switching over from SP to SP.2, the controller lines out at SP.2. The time (t.SP) runs when the process value enters the adjusted band around the setpoint (x = SP ± b.ti). When the timer has elapsed, the controller switches back to setpoint SP, and the lower display alternates between 'End' and the setpoint. |
| 10 | Setpoint controller with start-up function. The start-up function is a protective function, e.g. with hot runner control. To prevent destruction of high-performance heating elements, they must be heated slowly to remove any humidity. With activated start-up function, the controller maintains the reduced starting temperature for a defined dwell period. Subsequently, the controller switches over to the main setpoint. |
| 11 | Setpoint controllers are switchable to external setpoint and to a second setpoint, always with the start-up function.The start-up function is a protective function, e.g. with hot runner control. To prevent destruction of high-performance heating elements, they must be heated slowly to remove any humidity. With activated start-up function, the controller maintains the reduced starting temperature for a defined dwell period. Subsequently, the controller switches over to the main setpoint. |

# 1 Cntr

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| b.ti | r/w | base<br>1dP<br>2dP<br>3dP | 3152<br>11344<br>19536<br>27728 | 39072 | Float | 0...9999 ☐ | Timer tolerance band for operating mode:1 (bandwidth monitoring with switch-off at the end)2 (bandwidth monitoring with pause at the end), and6 (setpoint switchover). The timer runs as long as the process value is within the bandwidth limits (setpoint ± b.ti). |
| C.Fnc | r/w | base<br>1dP<br>2dP<br>3dP | 5050<br>13242<br>21434<br>29626 | 42868 | Enum | *Enum_CFnc* | Control behaviour (algorithm) referred to output value: e.g. 2- or 3-point controller, signaller, 3-point stepping control. |

| | |
|---|---|
| 0 | On/Off (2-point) controller or signaller with one output. The on/off controller or the signaller switches its output when the process value leaves the defined hysteresis band around the setpoint. |
| 1 | PID control, e.g. heating, with one output: Switched as a digital output (2-point) or used as an analog output (continuous). PID controllers respond quickly to changes of the control deviation, and typically do not exhibit any permanent control offset. |
| 2 | D / Y / Off, or 2-point controller with partial/full load switch-over. 2 digital outputs: Y1 is the switching output and Y2 is the changeover contact for D/Y. |
| 3 | 2 x PID control, e.g. heating/cooling. Two outputs: Switched as a digital output (3-point) or used as an analog output (continuous). PID controllers respond quickly to changes of the control deviation, and typically do not exhibit any permanent control offset. |
| 4 | 3-point stepping controller, e.g. for motor actuators. Two digital outputs. No actuating pulses are generated when the process is lined out. |
| 7 | 3-point signaller.<br>The 3-point signaller switches two digital outputs, depending on their switching difference (Sd1 and Sd2), the trigger point separation, and from the control deviation. |
| 8 | 3-point stepping controller that can be switched over to signaller operation. Via interface or front panel key (depending on configuration), the operating mode can be switched between 3-point stepping controller and signaller (1 output). |
| 9 | 3-point stepping controller can be switched over to 3-point-signaller operation. Via interface or front panel key (depending on configuration), the operating mode can be switched between 3-point stepping controller and 3-point-signaller (2 outputs). |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| mAn | r/w | base<br>1dP<br>2dP<br>3dP | 5051<br>13243<br>21435<br>29627 | 42870 | Enum | *Enum_mAn* | Enables the output value to be adjusted in manual operation. If adjustment is not enabled, the output value cannot be changed in manual operation, neither with the front keys nor via the interface.Note: This setting does not affect the auto/manual switchover function. |

| | |
|---|---|
| 0 | The output value cannot be changed in manual operation, neither with the front keys nor via the interface. |
| 1 | The output value is to be adjusted in manual operation (see also LOGI/mAn). |

# 1 Cntr

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| C.Act | r/w | base<br>1dP<br>2dP<br>3dP | 5052<br>13244<br>21436<br>29628 | 42872 | Enum | *Enum_CAct* | Operating sense of the controller. Inverse operation (e.g. heating) means increased actator value when the process value falls.<br>Direct operation (e.g. cooling) means increased actuator value when the process value increases. |

| | |
|---|---|
| 0 | Inverse or opposed-sense response, e.g. heating. The controller output is increased with a falling process value, and decreased with a rising process value. |
| 1 | Direct or same-sense response, e.g. cooling. The controller output is increased with a rising process value, and decreased with a falling process value. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAIL | r/w | base<br>1dP<br>2dP<br>3dP | 5053<br>13245<br>21437<br>29629 | 42874 | Enum | *Enum_FAIL* | With the sensor break response, the operator determines the instrument's reaction to a sensor break, thus ensuring a safe process condition. |

| | |
|---|---|
| 0 | Controller outputs disabled. |
| 1 | y = parameter Y2 (Caution: fixed parameter Y2, not controller output Y2!). Note for three-point stepping controller: With Y2 < 0.01 CLOSED is set (DY= -100%), with 0.01 =< Y2 =< 99.9 no output is set (DY=0%), with Y2 > 99.9 OPEN is set (DY= +100%).<br>Note for signallers: With Y2 < 0.01 OFF is set, with 0.01 =< Y2 =< 99.9 status keeps unchanged, with Y2 > 99.9 ON is set. |
| 2 | y = mean output. The maximum permissible output can be adjusted with parameter Ym.H. To prevent determination of inadmissible values, mean value formation is only if the control deviation is lower than parameter L.Ym. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| rnG.L | r/w | base<br>1dP<br>2dP<br>3dP | 5059<br>13251<br>21443<br>29635 | 42886 | Float | -1999...9999 ☐ | Lower limit for the controller's operating range. The control range is independent of the measurement range. Reducing the control range will increase the sensitivity of the self-tuning process. |
| rnG.H | r/w | base<br>1dP<br>2dP<br>3dP | 5060<br>13252<br>21444<br>29636 | 42888 | Float | -1999...9999 ☐ | Upper limit for the controller's operating range. The control range is independent of the measurement range. Reducing the control range will increase the sensitivity of the self-tuning process. |
| SP2C | r/w | base<br>1dP<br>2dP<br>3dP | 5054<br>13246<br>21438<br>29630 | 42876 | Enum | *Enum_SP2C* | When switching over to the 2nd setpoint SP.2, control is performed without cooling. |

| | |
|---|---|
| 0 | Standard (cooling permitted with all setpoints). |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| CYCL | r/w | base<br>1dP<br>2dP<br>3dP | 5055<br>13247<br>21439<br>29631 | 42878 | Enum | *Enum_CYCL* | Duty cycle for 2-point and 3-point controllers. Internally, the controller calculates a continuous output value, which is converted into switching pulses for digital outputs. The user can adapt the setting to calculate various duty cycles (on/off ratio). |

| | |
|---|---|
| 0 | Standard. 'Bathtub curve'. The adjusted duty cycles t1 and t2 are valid for ± 50% control output. With very small and very large control outputs, the effective duty cycle is increased sufficiently to prevent nonsensically short operating pulses. The shortest pulses are limited to ¼ of t1 and ¼ of t2. |

# 1 Cntr

## ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| tunE | r/w | base 5056<br>1dP 13248<br>2dP 21440<br>3dP 29632 | 42880 | Enum | *Enum_tune* | Self-tuning procedure / sequence. Choice between:step response tuning during start-up and pulse response tuning at setpoint; or pulse response tuning during start-up and at setpoint; or only step response tuning during start-up, and no tuning at setpoint (no pulse). |

| | |
|---|---|
| 0 | At start-up with step function, impulse function at setpoint.<br>The step function at start up requires a control deviation of more than 10% of the control range. At setpoint, with control deviation less than 10% of the control range, tuning is done with the impulse function. |
| 1 | At start-up with impulse function. Setting for fast controlled systems (e.g. hot runner control).<br>Always tuning with impulse function. At start up, with a control deviation of more than 10% of the control range, the control loop is optimized for a wide control range. At setpoint the control deviation during self-tuning is small. |
| 2 | At start up and at setpoint always tune step function at start up.<br>Tuning is done with step function at start up, regardless of the control deviation. |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Strt | r/w | base 5057<br>1dP 13249<br>2dP 21441<br>3dP 29633 | 42882 | Enum | *Enum_Strt* | Start of self-tuning. Self-tuning can always be started manually at the request of the operator.<br>Here, it is possible to determine that self-tuning is started automatically under the following conditions: On power-up or when an oscillation of the process value is detected. |

| | |
|---|---|
| 0 | No automatic start (manual start via front interface) |
| 1 | Manual or automatic start of auto-tuning at power on or when oscillating is detected (oscillating of process value by more than ± 0.5% of the control range, and simultaneously the output value by more than 20%.)<br>Note: Though the process is unchanged, at power on always the (time-consuming) auto-tuning is started. |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Adt0 | r/w | base 5061<br>1dP 13253<br>2dP 21445<br>3dP 29637 | 42890 | Enum | *Enum_Adt0* | Optimization of the switching cycles t1 and t2 for the DED conversion can be disabled here. In order to fine-tune the positioning action, the switching periods are changed by the self-tuning function, if automatic tuning is configured. |

| | |
|---|---|
| 0 | The duty cycle is determined by the self-tuning procedure. This ensures the best control results. |
| 1 | The cycle duration is not determinated by auto-tuning. An oversized cycle duration causes bad control behavior. An undersized cycle duration causes a more frequent switching, which can raise the wearout of mechanical actuators (relay, contactor). |

## PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Pb1 | r/w | base 5000<br>1dP 13192<br>2dP 21384<br>3dP 29576 | 42768 | Float | 1...9999 ☐ | Proportional band 1 (heating) in engineering unit, e.g. °C.<br>Pb defines the relationship between controller output and control deviation. The smaller Pb is, the stronger is the control action for a given control deviation. If Pb is too large or too small, the control loop will oscillate (hunting). |

# 1 Cntr

## PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| Pb2 | r/w | base 5001<br>1dP 13193<br>2dP 21385<br>3dP 29577 | 42770 | Float | 1...9999 | ☐ | Proportional band 2 (cooling) in engineering units, e.g. °C.<br>Pb defines the relationship between controller output and control deviation. The smaller Pb is, the stronger is the control action for a given control deviation. If Pb is too large or too small, the control loop will oscillate (hunting). |
| ti1 | r/w | base 5002<br>1dP 13194<br>2dP 21386<br>3dP 29578 | 42772 | Float | 1...9999 | ☑ | Integral action time 1 (heating) [s]. Ti is the time constan of the integral portion. The smaller Ti is, the faster is the response of the integral action.<br>Ti too small: Control tends to oscillate.<br>Ti too large: Control is sluggish and needs a long time to line out. |
| ti2 | r/w | base 5003<br>1dP 13195<br>2dP 21387<br>3dP 29579 | 42774 | Float | 1...9999 | ☑ | Integral action time 2 (cooling) [s]. Ti is the time constan of the integral portion. The smaller Ti is, the faster is the response of the integral action.<br>Ti too small: Control tends to oscillate.<br>Ti too large: Control is sluggish and needs a long time to line out. |
| td1 | r/w | base 5004<br>1dP 13196<br>2dP 21388<br>3dP 29580 | 42776 | Float | 1...9999 | ☑ | Derivative action time 1 (heating) [s]. Td is the time constant of the derivative portion. The faster the process value changes, and the larger the value of Td is, the stronger will be the derivative action.<br>Td too small: Very little derivative action.<br>Td too large: Control tends to oscillate. |
| td2 | r/w | base 5005<br>1dP 13197<br>2dP 21389<br>3dP 29581 | 42778 | Float | 1...9999 | ☑ | Derivative action time 2 (cooling) [s]. Td is the time constant of the derivative portion. The faster the process value changes, and the larger the value of Td is, the stronger will be the derivative action.<br>Td too small: Very little derivative action.<br>Td too large: Control tends to oscillate. |
| t1 | r/w | base 5006<br>1dP 13198<br>2dP 21390<br>3dP 29582 | 42780 | Float | 0,4...9999 | ☐ | Minimum duty cycle 1 (heating) [s]. With the standard duty cycle converter, the shortest pulse duration is 1/4 x t1.<br>If the duty cycle is not to be optimized, this must be entered in the configuration.<br>(Default: Optimization of the duty cycle during self-tuning, but also if the output value is less than 5%). |
| t2 | r/w | base 5007<br>1dP 13199<br>2dP 21391<br>3dP 29583 | 42782 | Float | 0,4...9999 | ☐ | Minimum duty cycle 2 (cooling) [s]. With the standard duty cycle converter, the shortest pulse duration is 1/4 x t1.<br>If the duty cycle is not to be optimized, this must be entered in the configuration.<br>(Default: Optimization of the duty cycle during self-tuning, but also if the output value is less than 5%). |
| SH | r/w | base 5014<br>1dP 13206<br>2dP 21398<br>3dP 29590 | 42796 | Float | 0...9999 | ☐ | Neutral zone, or switching difference of the signaller [engineering unit].Too small: unnecessarily high switching frequency.Too large: reduced controller sensitivity.With 3-point controllers this slows down the direct transition from heating to cooling. With 3-point stepping controllers, it reduces the switching operations of the actuator around setpoint. |

# 1 Cntr

## PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| d.SP | r/w | base 5016<br>1dP 13208<br>2dP 21400<br>3dP 29592 | 42800 | Float | -1999...9999 ☐ | Separation of the D / Y switch-over point from the setpoint [engineering unit]. With a significant control deviation heating start  is in delta connection. When the control deviation increases, the instrument switches over to reduced power (Y connection) for line-out to the set-point. |
| tP | r/w | base 5009<br>1dP 13201<br>2dP 21393<br>3dP 29585 | 42786 | Float | 0,1...9999 ☑ | Minimum pulse duration [s]. Used for switching with constant periods. For positioning values that require a shorter pulse than adjusted for 'tp', the output is suppressed, but 'remembered'. The controller continues adding the internal short pulses until a value equal to 'tp' can be output. |
| tt | r/w | base 5015<br>1dP 13207<br>2dP 21399<br>3dP 29591 | 42798 | Float | 3...9999 ☐ | Travel time of the actuator motor [s]. If no feedback signal is available, the controller calculates the actuator position by means of an integrator and the adjusted motor travel time. For this reason, a precise definition of the motor travel time between min and max (0% and 100%) is important. |
| Y.Lo | r/w | base 5018<br>1dP 13210<br>2dP 21402<br>3dP 29594 | 42804 | Float | -105...105 ☐ | Lower output limit [%]<br>The range is depedant of the type of controller:<br>2 point controller: 0...ymax+1<br>3 point controller: -105   ymax-1 |
| Y.Hi | r/w | base 5019<br>1dP 13211<br>2dP 21403<br>3dP 29595 | 42806 | Float | -105...105 ☐ | Upper output limit [%]<br>The range is ymin+1 ....105 |
| Y2 | r/w | base 5017<br>1dP 13209<br>2dP 21401<br>3dP 29593 | 42802 | Float | -100...100 ☐ | Second positioning value [%]. Activated Y2 = positioner control.<br>Caution: The parameter 'positioning output Y2' must not be confused with the controller output Y2! |
| Y.0 | r/w | base 5020<br>1dP 13212<br>2dP 21404<br>3dP 29596 | 42808 | Float | -105...105 ☐ | Offset for die positioning value [%]. This is added to the controller output, and has the most effect with P and PD controllers. (With PID controllers, the effect is compensated by the integral action.) With a control deviation = 0, the P controller generates a control output Y0. |
| Ym.H | r/w | base 5021<br>1dP 13213<br>2dP 21405<br>3dP 29597 | 42810 | Float | -105...105 ☐ | Limit for the mean control output value Ym in case of sensor break [%]. The mean control output value is configurable as the response to sensor break. The maximum mean output value = YmH. |
| L.Ym | r/w | base 5022<br>1dP 13214<br>2dP 21406<br>3dP 29598 | 42812 | Float | 1...9999 ☐ | Max. control deviation (xw), at the start of mean value calculation [engineering unit]. When calculating the mean value, data are only taken into account if the control deviation is small enough. 'Lym' is a preset value that determines how precisely the calculated output value is matched to the setpoint. |
| E.H2O | r/w | base 5013<br>1dP 13205<br>2dP 21397<br>3dP 29589 | 42794 | Float | -1999...9999 ☐ | Min. temperature for water cooling. Below the set temperature no water cooling happens |

# 1 Cntr

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| t.on | r/w | base<br>1dP<br>2dP<br>3dP | 5010<br>13202<br>21394<br>29586 | 42788 | Float | 0,1...9999 | ☐ | Impulse length for water cooling. Fixed for all values of controller output. The pause time is varied. |
| t.oFF | r/w | base<br>1dP<br>2dP<br>3dP | 5011<br>13203<br>21395<br>29587 | 42790 | Float | 1...9999 | ☐ | Min. pause time for water cooling. The max. effective controller output results from t.on/(t.on+t.off)·100% |
| F.H2O | r/w | base<br>1dP<br>2dP<br>3dP | 5012<br>13204<br>21396<br>29588 | 42792 | Float | 0,1...9999 | ☐ | Adaptation of the (non-linear) water-cooling characteristic.If the cooling action is very strong, and causes an unfavourable transition between heating and cooling, a non-linear characteristic can reduce the cooling action considerably.Adjust FH20 = 1 for output values up to -70%; FH20 = 2 for values up to approx. -80%, and FH20 = 0.5 for up to approx. -60%. |
| HYS.L | r/w | base<br>1dP<br>2dP<br>3dP | 5028<br>13220<br>21412<br>29604 | 42824 | Float | 0...9999 | ☐ | Switching hysteresis below the setpoint of the signaller [engineering unit]. |
| HYS.H | r/w | base<br>1dP<br>2dP<br>3dP | 5029<br>13221<br>21413<br>29605 | 42826 | Float | 0...9999 | ☐ | Switching hysteresis above the setpoint of the signaller [engineering unit]. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Tu2 | r | base<br>1dP<br>2dP<br>3dP | 5145<br>13337<br>21529<br>29721 | 43058 | Float | 0...9999 | ☐ | 'Cooling' delay time of the loop. Tu is calculated by the self-tuning function: It is the time delay before the process reacts significantly. In effect, Tu is a dead time that is determined by the reaction of the process to a change of the control output. It is used for defining controller action. |
| Vmax2 | r | base<br>1dP<br>2dP<br>3dP | 5146<br>13338<br>21530<br>29722 | 43060 | Float | 0...9999 | ☐ | Max. rate of change for 'cooling', i.e. the fastest process value increase during self-tuning. Vmax is calculated by the self-tuning function, and is determined by the reaction of the process to a change of the control output It is used for defining controller action. |
| Kp2 | r | base<br>1dP<br>2dP<br>3dP | 5147<br>13339<br>21531<br>29723 | 43062 | Float | 0...9999 | ☐ | Process gain for 'cooling'. For control loops with self-regulation, process gain is the ratio determined by the change of the control output and the resulting permanent change of the process value. Kp is calculated by the self-tuning function, and is used for defining controller action. |

# 1 Cntr

## Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| St.Cntr | r | base 5100<br>1dP 13292<br>2dP 21484<br>3dP 29676 | 42968 | Int | 0...65535 | ☐ | Status informations of the controller.f.e. switching signals, controller off or informations about selftuning. The controller sratus shows the actual adjustments of the controller. |

Bit 0: Switching signal heating: 0: off 1: on
Bit 1: Switching signal cooling: 0: off 1: on
Bit 2: Sensor error 0: ok 1: error
Bit 3: Controlsignal: Manual/automatic
0: automatic 1: manual
Bit 4: Controlsignal: Y2
0: Y2 not activ 1: Y2 activ
Bit 5: Controlsignal: Ext. setting of outputsignal
0: not activ 1: activ
Bit 6: Controlsignal: Controller off
0: contr. on 1: contr. off
Bit 7: Controlsignal:The activ parameter set
0: parameterset 1
1: parameterset 2
Bit 8: Loopalarm
0: no alarm
1: alarm
Bit 9: Soft start function
0: not activ
1: activ
Bit 10: Rate to setpoint
0: not activ
1: activ
Bit 11: Not used
Bit 12-15: Internal functional statuses (operating state)
0 0 0 0 Automatic
0 0 0 1 Selftuning is running
0 0 1 0 Selftuning faulty
(Waiting for operator signal)
0 0 1 1 Sensor error
0 1 0 0 Not used
0 1 0 1 Manual
0 1 1 1 Not used
1 0 0 0 Manual, with external presetting of the outputsignal
1 0 0 1 Outputs switched off (neutral)
1 0 1 0 Abortion of the selftuning (by control- or error-signal)

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| diFF | r | base 5104<br>1dP 13296<br>2dP 21488<br>3dP 29680 | 42976 | Float | -1999...9999 | ☐ | Control deviation, is defined as process value minus setpoint. Positive Xw means that the process value is above the setpoint. A small control deviation indicates precise control. |
| POS | r | base 5105<br>1dP 13297<br>2dP 21489<br>3dP 29681 | 42978 | Float | 0...100 | ☐ | The position feedback Yp shows the actuator position with 3-point stepping controllers. If Yp is outside the limits Ymin and Ymax, the output of positioning pulses is suppressed. |
| Tu1 | r | base 5141<br>1dP 13333<br>2dP 21525<br>3dP 29717 | 43050 | Float | 0...9999 | ☐ | 'Heating' delay time of the loop. Tu is calculated by the self-tuning function: It is the time delay before the process reacts significantly. In effect, Tu is a dead time that is determined by the reaction of the process to a change of the control output. It is used for defining controller action. |
| Ypid | r | base 5103<br>1dP 13295<br>2dP 21487<br>3dP 29679 | 42974 | Float | -120...120 | ☐ | Output value Ypid is the output signal determined by the controller, and from which the switching pulses for the digital and analog control outputs are calculated. Ypid is also available as an analog signal. e.g. for visualization. |

# 1 Cntr

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Ada.St | r/w | base<br>1dP<br>2dP<br>3dP | 5150<br>13342<br>21534<br>29726 | 43068 | Enum | *Enum_AdaStart* | | Starting / stopping the self-tuning function.After the start signal, the controller waits until the process reaches a stable condition (PIR) before it starts the self-tuning process. Self-tuning can be aborted manually at any time.<br>After a successful self-tuning attempt, the controller automatically resumes normal operation. |
| | | | | | | 0 | 'Stop' will abort the self-tuning process, and the controller returns to normal operation with the previous parameter settings. | |
| | | | | | | 1 | Start of the self-tuning process is possible during manual or automatic controller operation. | |
| Yman | r/w | base<br>1dP<br>2dP<br>3dP | 5151<br>13343<br>21535<br>29727 | 43070 | Float | -110...110 | ☐ | Absolute preset output value, which is used as output value during manual operation.<br>Caution: With 3-point stepping controllers, Yman (evaluated the same as Dyman) is added to the actual output value as a relative shift. |
| dYman | r/w | base<br>1dP<br>2dP<br>3dP | 5152<br>13344<br>21536<br>29728 | 43072 | Float | -220...220 | ☐ | Differential preset output value, which is added to the actual output value during manual operation. Negative values reduce the output. |
| Yinc | r/w | base<br>1dP<br>2dP<br>3dP | 5153<br>13345<br>21537<br>29729 | 43074 | Enum | *Enum_YInc* | | Increasing the output value. There are two speeds: 40 s or 10 s for the change from 0 % to 100 %.<br>Note: The 3-point stepping controller translates the increments as UP. |
| | | | | | | 0 | not active | |
| | | | | | | 1 | increment output | |
| Ydec | r/w | base<br>1dP<br>2dP<br>3dP | 5154<br>13346<br>21538<br>29730 | 43076 | Enum | *Enum_YDec* | | Decreasing the output value. There are two speeds: 40 s or 10 s for the change from 0 % to 100 %.<br>Note: The 3-point stepping controller translates the increments as DOWN. |
| | | | | | | 0 | not active | |
| | | | | | | 1 | decrement output | |
| SP.EF | r | base<br>1dP<br>2dP<br>3dP | 5101<br>13293<br>21485<br>29677 | 42970 | Float | -1999...9999 | ☐ | Effective setpoint. The value reached at the end of setpoint processing, after taking W2, external setpoint, gradient, boost function, programmer settings, start-up function, and limit functions into account. Comparison with the effective process value leads to the control deviation, from which the necessary controller response is derived. |
| In.? | r | base<br>1dP<br>2dP<br>3dP | 5102<br>13294<br>21486<br>29678 | 42972 | Float | -1999...9999 | ☐ | Measurement value after the measurement value correction (e.g. with offset or 2-point correction, and scaling). |

# 1 Cntr

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| St.Tune | r | base<br>1dP<br>2dP<br>3dP | 5140<br>13332<br>21524<br>29716 | 43048 | Int | 0...65535 | ☐ | Status information during self-tuning, e.g. the actual condition, and possible results, warnings, and error messages. |

Bit 0 Process lined out; 0 = No; 1 = Yes
Bit 1 Operating mode 'Self-tuning controller; 0 = Off; 1 = On
Bit 2 Result of controller self-tuning; 0 = OK; 1 = Fault
Bit 3 - 7 Not used
Bit 8 - 11 Result of the 'heating' attempt
0 0 0 0 No message / Attempt still running
0 0 0 1 Successful
0 0 1 0 Successful, with risk of exceeded setpoint
0 0 1 1 Error: Wrong operating sense
0 1 0 0 Error: No response from process
0 1 0 1 Error: Turning point too low
0 1 1 0 Error: Risk of exceeded setpoint
0 1 1 1 Error: Step output too small
1 0 0 0 Error: Setpoint reserve too small
Bit 12 - 15 Result of 'cooling' attempt (same as heating attempt)

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Vmax1 | r | base<br>1dP<br>2dP<br>3dP | 5142<br>13334<br>21526<br>29718 | 43052 | Float | 0...9999 | ☐ | Max. rate of change for 'heating', i.e. the fastest process value increase during self-tuning. Vmax is calculated by the self-tuning function, and is determined by the reaction of the process to a change of the control output It is used for defining controller action. |
| Kp1 | r | base<br>1dP<br>2dP<br>3dP | 5143<br>13335<br>21527<br>29719 | 43054 | Float | 0...9999 | ☐ | Process gain for 'heating'. For control loops with self-regulation, process gain is the ratio determined by the change of the control output and the resulting permanent change of the process value. Kp is calculated by the self-tuning function, and is used for defining controller action. |

# 1 Cntr

## • Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Msg2 | r | base | 5148 | 43064 | Enum | *Enum_Msg* | The result of self-tuning for 'cooling' indicates whether self-tuning was successful, and with what result. |
| | | 1dP | 13340 | | | | |
| | | 2dP | 21532 | | | | |
| | | 3dP | 29724 | | | | |

| Value | Description |
|---|---|
| 0 | no message/tuning is active |
| 1 | Self-tuning has been completed successfully. The new parameters are valid. |
| 2 | Self-tuning was successful, but with a warning. The new parameters are valid.<br>Note: Self-tuning was aborted due to the risk of an exceeded setpoint, but useful parameters were determined. Possibly repeat the attempt with an increased setpoint reserve. |
| 3 | Process responds in the wrong direction.<br>Possible remedy: Check the output signal sense (inverse <-> direct), and re-configure the controller if necessary (inverse <-> direct). |
| 4 | No response from the process. Perhaps the control loop is open.<br>Possible remedy: Check sensor, connections, and process. |
| 5 | Self-tuning was successful, but with a warning. The new parameters are valid.<br>Note: The process value turning point of the step response is too low. Quality of control is limited.<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 6 | Self-tuning was aborted due to the risk of an exceeded setpoint. No useful parameters were determined.<br>Possible remedy: Repeat the attempt with an increased setpoint reserve. |
| 7 | The step output change is not large enough (minimum change > 5 %).<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 8 | Self-tuning was stopped before the output step change was made, because the setpoint reserve is too small (the controller waits). Confirming this error message aborts the self-tuning, and the controller continues operation in the automatic mode.<br>Possible remedy: Reduce the setpoint adjustment range, change the setpoint, or reduce the process value. |

## 1 Cntr

### Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Msg1 | r | base<br>1dP<br>2dP<br>3dP | 5144<br>13336<br>21528<br>29720 | 43056 | Enum | *Enum_Msg* | The result of self-tuning for 'heating' indicates whether self-tuning was successful, and with what result. |

| Value | Description |
|---|---|
| 0 | no message/tuning is active |
| 1 | Self-tuning has been completed successfully. The new parameters are valid. |
| 2 | Self-tuning was successful, but with a warning. The new parameters are valid.<br>Note: Self-tuning was aborted due to the risk of an exceeded setpoint, but useful parameters were determined. Possibly repeat the attempt with an increased setpoint reserve. |
| 3 | Process responds in the wrong direction.<br>Possible remedy: Check the output signal sense (inverse <-> direct), and re-configure the controller if necessary (inverse <-> direct). |
| 4 | No response from the process. Perhaps the control loop is open.<br>Possible remedy: Check sensor, connections, and process. |
| 5 | Self-tuning was successful, but with a warning. The new parameters are valid.<br>Note: The process value turning point of the step response is too low. Quality of control is limited.<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 6 | Self-tuning was aborted due to the risk of an exceeded setpoint. No useful parameters were determined.<br>Possible remedy: Repeat the attempt with an increased setpoint reserve. |
| 7 | The step output change is not large enough (minimum change > 5 %).<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 8 | Self-tuning was stopped before the output step change was made, because the setpoint reserve is too small (the controller waits).<br>Confirming this error message aborts the self-tuning, and the controller continues operation in the automatic mode.<br>Possible remedy: Reduce the setpoint adjustment range, change the setpoint, or reduce the process value. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| YGrw | r/w | base<br>1dP<br>2dP<br>3dP | 5155<br>13347<br>21539<br>29731 | 43078 | Enum | *Enum_YGrwLs* | Gradient of Y-variation 'slow' or 'fast'. Changes the positioning output speed. There are two speeds for output variation: from 0% to 100% in 40s or in 10s. |

| Value | Description |
|---|---|
| 0 | Slow change of Y, from 0% to 100% in 40 seconds. |
| 1 | Fast change of Y, from 0% to 100% in 10 seconds. |

## 2 InP.1

### • ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| S.tYP | r/w | base | 1150 | 35068 | Enum | *Enum_StYP* | Sensor type selection |
| | | 1dP | 9342 | | | | |
| | | 2dP | 17534 | | | | |
| | | 3dP | 25726 | | | | |

| Value | Description |
|---|---|
| 0 | Thermocouple type L (-100...900°C), Fe-CuNi DIN<br>Fahrenheit: -148...1652°F |
| 1 | Thermocouple type J (-100...1200°C), Fe-CuNi<br>Fahrenheit: -148...2192°F |
| 2 | Thermocouple type K (-100...1350°C), NiCr-Ni<br>Fahrenheit: -148...2462°F |
| 3 | Thermocouple type N (-100...1300°C), Nicrosil-Nisil<br>Fahrenheit: -148...2372°F |
| 4 | Thermocouple type S (0...1760°C), PtRh-Pt 10%<br>Fahrenheit: 32...3200°F |
| 5 | Thermocouple type R (0...1760°C), PtRh-Pt13%<br>Fahrenheit: 32...3200°F |
| 6 | Thermocouple type T (-200...400°C), Cu-CuNi<br>Fahrenheit: -328...752°F |
| 7 | Thermocouple type C (0...2315°C), W5%Re-W26%Re<br>Fahrenheit: 32...4199°F |
| 8 | Thermocouple type D (0...2315°C), W3%Re-W25%Re<br>Fahrenheit: 32...4199°F |
| 9 | Thermocouple type E (-100...1000°C), NiCr-CuNi<br>Fahrenheit: -148...1832°F |
| 10 | Thermocouple type B (0/100...1820°C), PtRh-Pt6%<br>Fahrenheit: 32/212 ... 3308°F |
| 18 | Special thermocouple with a linearization characteristic selectable by the user. This enables non-linear signals to be simulated or linearized. |
| 20 | Pt100 (-200.0 ... 100.0(150.0)°C)<br>Measuring range up to 150°C at reduced lead resistance.<br>Fahrenheit: -328...212(302) °F |
| 21 | Pt100 (-200.0 ... 850,0°C)<br>Fahrenheit: -328...1562 °F |
| 22 | Pt 1000 (-200.0...850.0°C)<br>Fahrenheit: -328...1562 °F |
| 23 | Special : 0...4500 Ohms.<br>For KTY 11-6 with preset special linearization (-50...150°C or -58...302 °F). |
| 30 | current : 0/4...20 mA |
| 40 | voltage : 0...10 V |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| S.Lin | r/w | base | 1151 | 35070 | Enum | *Enum_SLin* | Special linearization (not adjustable for all sensor types S.tYP). The linearization table can be created with the Engineering Tool. |
| | | 1dP | 9343 | | | | |
| | | 2dP | 17535 | | | | |
| | | 3dP | 25727 | | | | |

| Value | Description |
|---|---|
| 0 | No special linearization. |
| 1 | Special linearization. Definition of the linearization table is possible with the Engineering Tool. |

## 2 InP.1

### ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Corr | r/w | base<br>1dP<br>2dP<br>3dP | 160<br>8352<br>16544<br>24736 | 33088 | Enum | *Enum_Corr3* | Measured value correction / scaling |

| | |
|---|---|
| 0 | without scaling |
| 1 | The offset correction (in the CAL Level) can be done on-line in the process. If InL shows the lower input value of the scaling point, then OuL must be adjusted to the corresponding display value. Adjustments are made via the front panel keys of the device only. |
| 2 | Two-point correction (in CAL-Level) is possible offline via process value transmitter or on-line in the process. Set process value for the upper and lower scaling point and confirm as input value InL or InH, then set the belonging displayed value OuL and OuH. The settings are done via the front of the device. |
| 3 | Scaling (at PArA-level). The input values for the upper (InL, OuL) and lower scaling point (InH. OuH) are visible at the parameter level. Adjustment is made via front operation or the engineering tool. |

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| InL.? | r/w | base<br>1dP<br>2dP<br>3dP | 110<br>9292<br>17484<br>25676 | 34968 | Float | -1999...9999 ☐ | Input value of the lower scaling point. The display of the value is done using the corresponding measured electrical value. |
| OuL.? | r/w | base<br>1dP<br>2dP<br>3dP | 101<br>9293<br>17485<br>25677 | 34970 | Float | -1999...9999 ☐ | Display value of the lower scaling point. This is the physical value, which is assigned to the measured lower input value. |
| InH.? | r/w | base<br>1dP<br>2dP<br>3dP | 102<br>9294<br>17486<br>25678 | 34972 | Float | -1999...9999 ☐ | Input value of the upper scaling point. The display of the value is done using the corresponding measured electrical value. |
| OuH.? | r/w | base<br>1dP<br>2dP<br>3dP | 103<br>9295<br>17487<br>25679 | 34974 | Float | -1999...9999 ☐ | Display value of the upper scaling point. This is the physical value, which is assigned to the measured upper input value. |
| t.F? | r/w | base<br>1dP<br>2dP<br>3dP | 104<br>9296<br>17488<br>25680 | 34976 | Float | 0...100 ☐ | Filter time constant [s]. Every input is fitted with a digital (software) low-pass filter for suppressing process-related disturbances on the input leads. Higher filter settings improve the suppression, but increase the delay of the input signals. |

### Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| In.?r | r | base<br>1dP<br>2dP<br>3dP | 117<br>9362<br>17554<br>25746 | 35108 | Float | -1999...9999 ☐ | Measurement value before the measurement value correction (unprocessed). |

# 2 InP.1

## Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Fail | r | base 1171<br>1dP 9363<br>2dP 17555<br>3dP 25747 | 35110 | Enum | *Enum_InpFail* | Input circuit fault: faulty or incorrectly connected sensor. |

| | |
|---|---|
| 0 | no error |
| 1 | sensor break |
| 2 | Incorrect polarity at input. |
| 4 | short circuit at input. |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| In.? | r | base 1172<br>1dP 9364<br>2dP 17556<br>3dP 25748 | 35112 | Float | -1999...9999 ☐ | Measurement value after the measurement value correction (e.g. with offset or 2-point correction, and scaling). |
| F.Inp | r/w | base 1180<br>1dP 9372<br>2dP 17564<br>3dP 25756 | 35128 | Float | -1999...9999 ☐ | Forcing the value for an analog input INP. Forcing involves the external operation of an input. The instrument takes over the value at this input like a measurement value (preset value for inputs from a superordinate system, e.g. for a function test.) |

# 3 InP.2

## ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| I.Fnc | r/w | base 161<br>1dP 8353<br>2dP 16545<br>3dP 24737 | 33090 | Enum | *Enum_IFnc* | Selection of the function assigned to the value at INP2, e.g. value at INP2 is the external setpoint. |

| | |
|---|---|
| 0 | no function (subsequent input data are skipped) |
| 1 | heating current input. |
| 2 | External setpoint SP.E or (depending on version) external setpoint shift SP.E.<br>(Switch-over is done via -> LOGI/SP.E). |
| 5 | Preset for external positioning value Y.E (switch-over via -> LOGI/Y.E) |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| S.tYP | r/w | base 1250<br>1dP 9442<br>2dP 17634<br>3dP 25826 | 35268 | Enum | *Enum_StYP2_1* | Sensor type selection |

| | |
|---|---|
| 30 | Current : 0...20 mA / 4...20 mA |
| 31 | 0...30 mA current (AC) |

# 3 InP.2

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Corr | r/w | base | 162 | 33092 | Enum | *Enum_Corr* | Measured value correction / scaling |
| | | 1dP | 8354 | | | | |
| | | 2dP | 16546 | | | | |
| | | 3dP | 24738 | | | | |

| | |
|---|---|
| 0 | without scaling |
| 1 | The offset correction (in the CAL Level) can be done on-line in the process. If InL shows the lower input value of the scaling point, then OuL must be adjusted to the corresponding display value. Adjustments are made via the front panel keys of the device only. |
| 2 | Two-point correction (in CAL-Level) is possible offline via process value transmitter or on-line in the process. Set process value for the upper and lower scaling point and confirm as input value InL or InH, then set the belonging displayed value OuL and OuH. The settings are done via the front of the device. |
| 3 | Scaling (at PArA-level). The input values for the upper (InL, OuL) and lower scaling point (InH. OuH) are visible at the parameter level. Adjustment is made via front operation or the engineering tool. |

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| InL.? | r/w | base | 1200 | 35168 | Float | -1999...9999 ☐ | Input value of the lower scaling point. The display of the value is done using the corresponding measured electrical value. |
| | | 1dP | 9392 | | | | |
| | | 2dP | 17584 | | | | |
| | | 3dP | 25776 | | | | |
| OuL.? | r/w | base | 1201 | 35170 | Float | -1999...9999 ☐ | Display value of the lower scaling point. This is the physical value, which is assigned to the measured lower input value. |
| | | 1dP | 9393 | | | | |
| | | 2dP | 17585 | | | | |
| | | 3dP | 25777 | | | | |
| InH.? | r/w | base | 1202 | 35172 | Float | -1999...9999 ☐ | Input value of the upper scaling point. The display of the value is done using the corresponding measured electrical value. |
| | | 1dP | 9394 | | | | |
| | | 2dP | 17586 | | | | |
| | | 3dP | 25778 | | | | |
| OuH.? | r/w | base | 1203 | 35174 | Float | -1999...9999 ☐ | Display value of the upper scaling point. This is the physical value, which is assigned to the measured upper input value. |
| | | 1dP | 9395 | | | | |
| | | 2dP | 17587 | | | | |
| | | 3dP | 25779 | | | | |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| In.? | r | base | 1270 | 35308 | Float | -1999...9999 ☐ | Measurement value after the measurement value correction (e.g. with offset or 2-point correction, and scaling). |
| | | 1dP | 9462 | | | | |
| | | 2dP | 17654 | | | | |
| | | 3dP | 25846 | | | | |

## 3 InP.2

### Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Fail | r | base | 1271 | 35310 | Enum | *Enum_InpFail* | Input circuit fault: faulty or incorrectly connected sensor. |
| | | 1dP | 9463 | | | | |
| | | 2dP | 17655 | | | | |
| | | 3dP | 25847 | | | | |

| | |
|---|---|
| 0 | no error |
| 1 | sensor break |
| 2 | Incorrect polarity at input. |
| 4 | short circuit at input. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| In.?r | r | base | 1272 | 35312 | Float | -1999...9999 ☐ | Measurement value before the measurement value correction (unprocessed). |
| | | 1dP | 9464 | | | | |
| | | 2dP | 17656 | | | | |
| | | 3dP | 25848 | | | | |
| F.Inp | r/w | base | 1280 | 35328 | Float | -1999...9999 ☐ | Forcing the value for an analog input INP. Forcing involves the external operation of an input. The instrument takes over the value at this input like a measurement value (preset value for inputs from a superordinate system, e.g. for a function test.) |
| | | 1dP | 9472 | | | | |
| | | 2dP | 17664 | | | | |
| | | 3dP | 25856 | | | | |

## 4 InP.3

### ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| I.Fnc | r/w | base | 166 | 33100 | Enum | *Enum_IFnc* | Selection of the function assigned to the value at INP3, e.g. value at INP3 is the external setpoint. |
| | | 1dP | 8358 | | | | |
| | | 2dP | 16550 | | | | |
| | | 3dP | 24742 | | | | |

| | |
|---|---|
| 0 | no function (subsequent input data are skipped) |
| 1 | heating current input. |
| 2 | External setpoint SP.E or (depending on version) external setpoint shift SP.E.<br>(Switch-over is done via -> LOGI/SP.E). |
| 5 | Preset for external positioning value Y.E (switch-over via -> LOGI/Y.E) |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| S.tYP | r/w | base | 1350 | 35468 | Enum | *Enum_StYP3* | Sensor type selection. For sensors with signals of resistance transducer, current or voltage measuring, scaling can be adjusted. |
| | | 1dP | 9542 | | | | |
| | | 2dP | 17734 | | | | |
| | | 3dP | 25926 | | | | |
| S.Lin | r/w | base | 1351 | 35470 | Enum | *Enum_SLin* | Special linearization (not adjustable for all sensor types S.tYP). The linearization table can be created with the Engineering Tool. |
| | | 1dP | 9543 | | | | |
| | | 2dP | 17735 | | | | |
| | | 3dP | 25927 | | | | |

| | |
|---|---|
| 0 | No special linearization. |
| 1 | Special linearization. Definition of the linearization table is possible with the Engineering Tool. |

# 4 InP.3

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Corr | r/w | base<br>1dP<br>2dP<br>3dP | 165<br>8357<br>16549<br>24741 | 33098 | Enum | *Enum_Corr3* | Measured value correction / scaling |

| | |
|---|---|
| 0 | without scaling |
| 1 | The offset correction (in the CAL Level) can be done on-line in the process. If InL shows the lower input value of the scaling point, then OuL must be adjusted to the corresponding display value. Adjustments are made via the front panel keys of the device only. |
| 2 | Two-point correction (in CAL-Level) is possible offline via process value transmitter or on-line in the process. Set process value for the upper and lower scaling point and confirm as input value InL or InH, then set the belonging displayed value OuL and OuH. The settings are done via the front of the device. |
| 3 | Scaling (at PArA-level). The input values for the upper (InL, OuL) and lower scaling point (InH. OuH) are visible at the parameter level. Adjustment is made via front operation or the engineering tool. |

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| InL.? | r/w | base<br>1dP<br>2dP<br>3dP | 1300<br>9492<br>17684<br>25876 | 35368 | Float | -1999...9999 ☐ | Input value of the lower scaling point. The display of the value is done using the corresponding measured electrical value. |
| OuL.? | r/w | base<br>1dP<br>2dP<br>3dP | 1301<br>9493<br>17685<br>25877 | 35370 | Float | -1999...9999 ☐ | Display value of the lower scaling point. This is the physical value, which is assigned to the measured lower input value. |
| InH.? | r/w | base<br>1dP<br>2dP<br>3dP | 1302<br>9494<br>17686<br>25878 | 35372 | Float | -1999...9999 ☐ | Input value of the upper scaling point. The display of the value is done using the corresponding measured electrical value. |
| OuH.? | r/w | base<br>1dP<br>2dP<br>3dP | 1303<br>9495<br>17687<br>25879 | 35374 | Float | -1999...9999 ☐ | Display value of the upper scaling point. This is the physical value, which is assigned to the measured upper input value. |
| t.F? | r/w | base<br>1dP<br>2dP<br>3dP | 1304<br>9496<br>17688<br>25880 | 35376 | Float | 0...999,9 ☐ | Filter time constant [s]. Every input is fitted with a digital (software) low-pass filter for suppressing process-related disturbances on the input leads. Higher filter settings improve the suppression, but increase the delay of the input signals. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| In.? | r | base<br>1dP<br>2dP<br>3dP | 1370<br>9562<br>17754<br>25946 | 35508 | Float | -1999...9999 ☐ | Measurement value after the measurement value correction (e.g. with offset or 2-point correction, and scaling). |

# 4 InP.3

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Fail | r | base | 1371 | 35510 | Enum | *Enum_InpFail* | Input circuit fault: faulty or incorrectly connected sensor. |
| | | 1dP | 9563 | | | | |
| | | 2dP | 17755 | | | | |
| | | 3dP | 25947 | | | | |

| | |
|---|---|
| 0 | no error |
| 1 | sensor break |
| 2 | Incorrect polarity at input. |
| 4 | short circuit at input. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| In.?r | r | base | 1372 | 35512 | Float | -1999...9999 ☐ | Measurement value before the measurement value correction (unprocessed). |
| | | 1dP | 9564 | | | | |
| | | 2dP | 17756 | | | | |
| | | 3dP | 25948 | | | | |
| F.Inp | r/w | base | 1380 | 35528 | Float | -1999...9999 ☐ | Forcing the value for an analog input INP. Forcing involves the external operation of an input. The instrument takes over the value at this input like a measurement value (preset value for inputs from a superordinate system, e.g. for a function test.) |
| | | 1dP | 9572 | | | | |
| | | 2dP | 17764 | | | | |
| | | 3dP | 25956 | | | | |

# 5 Lim

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Fnc.? | r/w | base | 2150 | 37068 | Enum | *Enum_Fcn* | Activation and adjustment of the limit value alarm (e.g. for input circuit monitoring), e.g. with/without storage. |
| | | 1dP | 10342 | | | | |
| | | 2dP | 18534 | | | | |
| | | 3dP | 26726 | | | | |

| | |
|---|---|
| 0 | No limit value monitoring. |
| 1 | Measured value monitoring. The alarm signal is generated, if the limit is exceeded. If the measured value is within the limits (including hysteresis) again, this alarm signal is resetted. |
| 2 | Measured value monitoring + alarm status latch. An alarm signal is generated, if the limit is exceeded. A latched alarm signal remains latched until it is manually resetted. |

# 5 Lim

## ConF

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Src.? | r/w | base<br>1dP<br>2dP<br>3dP | 2151<br>10343<br>18535<br>26727 | 37070 | Enum | *Enum_Src* | Source for limit value. Selection of which value is to be monitored. |

| Value | Description |
|---|---|
| 0 | process value = absolute alarm |
| 1 | Control deviation xw (process value - setpoint) = relative alarm<br>Note: Monitoring with the effective setpoint Weff. For example using a ramp it is the changing setpoint, not the target setpoint of the ramp. |
| 2 | Control deviation Xw (= relative alarm) with suppression during start-up and setpoint changes. Limit value monitoring is continued as soon as the control deviation comes within the alarm limits again, at the latest after 10 * Ti1. |
| 6 | effective setpoint Weff.<br>For example the ramp-function changes the effective setpoint until it matches the internal (target) setpoint. |
| 7 | correcting variable y (controller output) |
| 11 | Control deviation Xw (= relative alarm) with suppression during start-up and setpoint change. Limit value monitoring is continued as soon as the control deviation comes within the alarm limits again. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HC.AL | r/w | base<br>1dP<br>2dP<br>3dP | 2050<br>10242<br>18434<br>26626 | 36868 | Enum | *Enum_HCAL* | Activation of alarm heat current function. Either overload or break can be monitored, overload = current I > heat current limit, or break = current I < heat current limit. Sho circuit is monitored in both cases. |

| Value | Description |
|---|---|
| 0 | no heating current alarm. |
| 1 | Overload and short circuit monitoring. Overload = current I > heat current limit. |
| 2 | Break and short circuit monitoring. Break = current I < heat current limit. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| LP.AL | r/w | base<br>1dP<br>2dP<br>3dP | 5058<br>13250<br>21442<br>29634 | 42884 | Enum | *Enum_LPAL* | Monitoring of control loop interruption (not possible with 3-point stepping controller, not possible with signaller) |

| Value | Description |
|---|---|
| 0 | switched off / inactive |
| 1 | LOOP alarm is generated, if with Y=100% there is no corresponding reaction of the process variable within the time of 2 x ti.<br>Possible remedial action: Check heating or cooling circuit, check sensor and replace it, if necessary, check controller and switching device. |

## PArA

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| L.? | r/w | base<br>1dP<br>2dP<br>3dP | 2100<br>10292<br>18484<br>26676 | 36968 | Float | -1999...9999 ☑ | Lower limit value. The alarm is triggered if the value falls below the limit, and is reset with lower limit value plus hysteresis. |
| H.? | r/w | base<br>1dP<br>2dP<br>3dP | 2101<br>10293<br>18485<br>26677 | 36970 | Float | -1999...9999 ☑ | Upper limit value. The alarm is triggered if the value rises above the limit, and is reset with upper lower limit value plus hysteresis. |

# 5 Lim

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| HYS.? | r/w | base | 2102 | 36972 | Float | 0...9999 | ☐ | Hysteresis of the limit value. Switching difference for upper and lower limit value. The limit value must change by this amount (rise above upper limit or fall below lower limit) before the limit value alarm is reset. |
| | | 1dP | 10294 | | | | | |
| | | 2dP | 18486 | | | | | |
| | | 3dP | 26678 | | | | | |
| dEL.? | r/w | base | 2103 | 36974 | Float | 0...9999 | ☐ | Delayed alarm of a limit value. The alarm is only triggered after the defined delay time. It is only indicated, and possibly stored, if it is still present after the delay time has elapsed. |
| | | 1dP | 10295 | | | | | |
| | | 2dP | 18487 | | | | | |
| | | 3dP | 26679 | | | | | |
| HC.A | r/w | base | 2000 | 36768 | Float | -1999...9999 | ☐ | Heating current monitoring limit [A]. Depending on configuration, and apart from short-circuit monitoring, an overload test checks whether the heating current is above the adjusted current limit, or below the limit when the heating is switched off. The heating current is measured by means of a current transformer (accessory), and the current range can be adapted. |
| | | 1dP | 10192 | | | | | |
| | | 2dP | 18384 | | | | | |
| | | 3dP | 26576 | | | | | |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| St.HC | r | base | 2070 | 36908 | Int | 0...3 | ☐ | Status of the heating current alarm. Displayable are heating current short-circuit and/or heating current alarm. Depending on configuration, the heating current alarm is either an interruption of heating current (I < limit value) or heating current overload (I > limit value). |
| | | 1dP | 10262 | | | | | |
| | | 2dP | 18454 | | | | | |
| | | 3dP | 26646 | | | | | |
| HC | r | base | 2071 | 36910 | Float | -1999...9999 | ☐ | Measured heating current [A]. Apart from the short circu test, and depending on configuration, an overcurrent test (current I > heating current limit) and an open circuit test (current I < heating current limit) is executed. The heating current is measured by means of a (separate) current transformer, whereby the input range can be scaled. |
| | | 1dP | 10263 | | | | | |
| | | 2dP | 18455 | | | | | |
| | | 3dP | 26647 | | | | | |
| SSr | r | base | 2072 | 36912 | Float | -1999...9999 | ☐ | Measured current with SSr [A]. The heating current (SSR) is short circuited, if there is a current flow even though the controller output is switched off.Suggested remedy: check heating current circuit, replace solid-state relay if necessary. |
| | | 1dP | 10264 | | | | | |
| | | 2dP | 18456 | | | | | |
| | | 3dP | 26648 | | | | | |
| St.Lim | r | base | 2170 | 37108 | Enum | *Enum_LimStatus* | | Limit value status: No alarm present or stored. |
| | | 1dP | 10362 | | | | | |
| | | 2dP | 18554 | | | | | |
| | | 3dP | 26746 | | | | | |

| | |
|---|---|
| 0 | no alarm |
| 1 | latched alarm |
| 2 | A limit value has been exceeded. |

# 6 Lim2

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Fnc.? | r/w | base<br>1dP<br>2dP<br>3dP | 2250<br>10442<br>18634<br>26826 | 37268 | Enum | *Enum_Fcn* | Activation and adjustment of the limit value alarm (e.g. for input circuit monitoring), e.g. with/without storage. |

| Value | Description |
|---|---|
| 0 | No limit value monitoring. |
| 1 | Measured value monitoring. The alarm signal is generated, if the limit is exceeded. If the measured value is within the limits (including hysteresis) again, this alarm signal is resetted. |
| 2 | Measured value monitoring + alarm status latch. An alarm signal is generated, if the limit is exceeded. A latched alarm signal remains latched until it is manually resetted. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Src.? | r/w | base<br>1dP<br>2dP<br>3dP | 2251<br>10443<br>18635<br>26827 | 37270 | Enum | *Enum_Src* | Source for limit value. Selection of which value is to be monitored. |

| Value | Description |
|---|---|
| 0 | process value = absolute alarm |
| 1 | Control deviation xw (process value - setpoint) = relative alarm<br>Note: Monitoring with the effective setpoint Weff. For example using a ramp it is the changing setpoint, not the target setpoint of the ramp. |
| 2 | Control deviation Xw (= relative alarm) with suppression during start-up and setpoint changes. Limit value monitoring is continued as soon as the control deviation comes within the alarm limits again, at the latest after 10 * Ti1. |
| 6 | effective setpoint Weff.<br>For example the ramp-function changes the effective setpoint until it matches the internal (target) setpoint. |
| 7 | correcting variable y (controller output) |
| 11 | Control deviation Xw (= relative alarm) with suppression during start-up and setpoint change. Limit value monitoring is continued as soon as the control deviation comes within the alarm limits again. |

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| L.? | r/w | base<br>1dP<br>2dP<br>3dP | 2200<br>10392<br>18584<br>26776 | 37168 | Float | -1999...9999 | ☑ | Lower limit value. The alarm is triggered if the value falls below the limit, and is reset with lower limit value plus hysteresis. |
| H.? | r/w | base<br>1dP<br>2dP<br>3dP | 2201<br>10393<br>18585<br>26777 | 37170 | Float | -1999...9999 | ☑ | Upper limit value. The alarm is triggered if the value rises above the limit, and is reset with upper lower limit value plus hysteresis. |
| HYS.? | r/w | base<br>1dP<br>2dP<br>3dP | 2202<br>10394<br>18586<br>26778 | 37172 | Float | 0...9999 | ☐ | Hysteresis of the limit value. Switching difference for upper and lower limit value. The limit value must change by this amount (rise above upper limit or fall below lower limit) before the limit value alarm is reset. |
| dEL.? | r/w | base<br>1dP<br>2dP<br>3dP | 2203<br>10395<br>18587<br>26779 | 37174 | Float | 0...9999 | ☐ | Delayed alarm of a limit value. The alarm is only triggered after the defined delay time. It is only indicated, and possibly stored, if it is still present after the delay time has elapsed. |

- **Signal**

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| St.Lim | r | base 2270<br>1dP 10462<br>2dP 18654<br>3dP 26846 | 37308 | Enum | *Enum_LimStatus* | Limit value status: No alarm present or stored. |

| | |
|---|---|
| 0 | no alarm |
| 1 | latched alarm |
| 2 | A limit value has been exceeded. |

## 7 Lim3

- **ConF**

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Fnc.? | r/w | base 2350<br>1dP 10542<br>2dP 18734<br>3dP 26926 | 37468 | Enum | *Enum_Fcn* | Activation and adjustment of the limit value alarm (e.g. for input circuit monitoring), e.g. with/without storage. |

| | |
|---|---|
| 0 | No limit value monitoring. |
| 1 | Measured value monitoring. The alarm signal is generated, if the limit is exceeded. If the measured value is within the limits (including hysteresis) again, this alarm signal is resetted. |
| 2 | Measured value monitoring + alarm status latch. An alarm signal is generated, if the limit is exceeded. A latched alarm signal remains latched until it is manually resetted. |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Src.? | r/w | base 2351<br>1dP 10543<br>2dP 18735<br>3dP 26927 | 37470 | Enum | *Enum_Src* | Source for limit value. Selection of which value is to be monitored. |

| | |
|---|---|
| 0 | process value = absolute alarm |
| 1 | Control deviation xw (process value - setpoint) = relative alarm<br>Note: Monitoring with the effective setpoint Weff. For example using a ramp it is the changing setpoint, not the target setpoint of the ramp. |
| 2 | Control deviation Xw (= relative alarm) with suppression during start-up and setpoint changes. Limit value monitoring is continued as soon as the control deviation comes within the alarm limits again, at the latest after 10 * Ti1. |
| 6 | effective setpoint Weff.<br>For example the ramp-function changes the effective setpoint until it matches the internal (target) setpoint. |
| 7 | correcting variable y (controller output) |
| 11 | Control deviation Xw (= relative alarm) with suppression during start-up and setpoint change. Limit value monitoring is continued as soon as the control deviation comes within the alarm limits again. |

# 7 Lim3

## PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| L.? | r/w | base 2300<br>1dP 10492<br>2dP 18684<br>3dP 26876 | 37368 | Float | -1999...9999 | ☑ | Lower limit value. The alarm is triggered if the value falls below the limit, and is reset with lower limit value plus hysteresis. |
| H.? | r/w | base 2301<br>1dP 10493<br>2dP 18685<br>3dP 26877 | 37370 | Float | -1999...9999 | ☑ | Upper limit value. The alarm is triggered if the value rises above the limit, and is reset with upper lower limit value plus hysteresis. |
| HYS.? | r/w | base 2302<br>1dP 10494<br>2dP 18686<br>3dP 26878 | 37372 | Float | 0...9999 | ☐ | Hysteresis of the limit value. Switching difference for upper and lower limit value. The limit value must change by this amount (rise above upper limit or fall below lower limit) before the limit value alarm is reset. |
| dEL.? | r/w | base 2303<br>1dP 10495<br>2dP 18687<br>3dP 26879 | 37374 | Float | 0...9999 | ☐ | Delayed alarm of a limit value. The alarm is only triggered after the defined delay time. It is only indicated, and possibly stored, if it is still present after the delay time has elapsed. |

## Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| St.Lim | r | base 2370<br>1dP 10562<br>2dP 18754<br>3dP 26946 | 37508 | Enum | *Enum_LimStatus* | Limit value status: No alarm present or stored. |

| | |
|---|---|
| 0 | no alarm |
| 1 | latched alarm |
| 2 | A limit value has been exceeded. |

# 8 LOGI

## ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| L_r | r/w | base 1051<br>1dP 9243<br>2dP 17435<br>3dP 25627 | 34870 | Enum | *Enum_dInP_Ks20* | Local / remote switchover (Remote: Adjustment of all values via the front panel is blocked). |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

# 8 LOGI

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP.2 | r/w | base | 1052 | 34872 | Enum | *Enum_dInP_Ks2* *0* | Source of the control signal for activating the second (safety) setpoint (SP.2=) W2. Note: W2 is not restricted by the setpoint limits. |
| | | 1dP | 9244 | | | | |
| | | 2dP | 17436 | | | | |
| | | 3dP | 25628 | | | | |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP.E | r/w | base | 1053 | 34874 | Enum | *Enum_dInP_Ks2* *0* | Switching between internal set-point an external setpoint SP.E. The external SP.E is either the absolute set-point Wext or the offset to the set-point (dependent on instrument and configuration). |
| | | 1dP | 9245 | | | | |
| | | 2dP | 17437 | | | | |
| | | 3dP | 25629 | | | | |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y2 | r/w | base | 1054 | 34876 | Enum | *Enum_dInP_Ks2* *0* | Source of the control signal for activating the second positioning output Y2. Activated Y2 = positioner control. Caution: The parameter 'positioning output Y2' must not be confused with the controller output Y2! |
| | | 1dP | 9246 | | | | |
| | | 2dP | 17438 | | | | |
| | | 3dP | 25630 | | | | |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

# 8 LOGI

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.E | r/w | base<br>1dP<br>2dP<br>3dP | 1055<br>9247<br>17439<br>25631 | 34878 | Enum | *Enum_dInP_Ks20* | Signal for activating the external output value. The internal output value Ypid is the controllers reaction on the process, with external output value Y.E the controller output is controlled. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| mAn | r/w | base<br>1dP<br>2dP<br>3dP | 1056<br>9248<br>17440<br>25632 | 34880 | Enum | *Enum_dInP_Ks20* | Source of the control signal for auto/manual switchover. In the automatic mode, the controller is in charge. In the manual mode, the outputs can be varied independently of the process. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| C.oFF | r/w | base<br>1dP<br>2dP<br>3dP | 1057<br>9249<br>17441<br>25633 | 34882 | Enum | *Enum_dInP_Ks20* | Source of the control signal for disabling all the controller outputs.Note: Forcing has priority, and remains active; alarm processing also remains active. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

# 8 LOGI

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| m.Loc | r/w | base<br>1dP<br>2dP<br>3dP | 1058<br>9250<br>17442<br>25634 | 34884 | Enum | *Enum_dInP_Ks20* | Source of the control signal to disable the auto/manual key. If the A/M key is disabled, switchover to manual operation is not possible. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Err.r | r/w | base<br>1dP<br>2dP<br>3dP | 1059<br>9251<br>17443<br>25635 | 34886 | Enum | *Enum_dInP_Ks20* | Source of the control signal for resetting all stored entries in the error list (the list contains all error messages and alarms). If an alarm is still present, i.e. the source of trouble has not been remedied, stored alarms cannot be acknowledged (reset). |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| booS | r/w | base<br>1dP<br>2dP<br>3dP | 1060<br>9252<br>17444<br>25636 | 34888 | Enum | *Enum_dInP_Ks20* | Source of the control signal for activating the boost function: The setpoint is increased by the value SP.bo for the duration t.bo. The boost function causes a brief setpoint increase, which is used to clear blocked channels from 'frozen' material in a hot runner system. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

# 8 LOGI

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Pid.2 | r/w | base<br>1dP<br>2dP<br>3dP | 1061<br>9253<br>17445<br>25637 | 34890 | Enum | *Enum_dInP_Ks2*<br>*0* | Source of the control signal for switchover between the two parameter sets. The second parameter set is complete, and comprises Pb (= proportional band), ti (= integral action time), and td (= derivative action time) for heating and for cooling. All other control parameters, e.g. the switching duty cycles, are valid for both parameter sets. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| P.run | r/w | base<br>1dP<br>2dP<br>3dP | 1062<br>9254<br>17446<br>25638 | 34892 | Enum | *Enum_dInP_Ks2*<br>*0* | Source of the control signal for switching the programmer between Run and Stop. On units with a simple programmer (only 1 program), a stop immediately causes a reset, followed by a new start. With units that have been defined as program controllers (several programs), the program is stopped, and then continued. |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| P.oFF | r/w | base<br>1dP<br>2dP<br>3dP | 1063<br>9255<br>17447<br>25639 | 34894 | Enum | *Enum_dInP_Ks2*<br>*0* | Source of the control signal for switching off the programmer (if the programmer is switched off, the internal setpoint becomes effective). |

| | |
|---|---|
| 0 | No function (switch-over via interface is possible) |
| 1 | always on |
| 2 | Digital Input di1 switches |
| 3 | Digital Input di1 switches |
| 4 | di3 switches (only visible with OPTION) |
| 5 | di4 switches (only visible with OPTION) |
| 6 | F-key switches |
| 7 | limit 1 switches |
| 8 | limit 2 switches |
| 9 | Limit 3 schaltet |

# 8 LOGI

## ConF

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| di.Fn | r/w | base<br>1dP<br>2dP<br>3dP | 1050<br>9242<br>17434<br>25626 | 34868 | Enum | *Enum_diFn* | Function of digital inputs (valid for all inputs) |

| | |
|---|---|
| 0 | Basic setting 'Off': A permanent positive signal switches this function 'On', which is connected to the digital input. Removal of the signal switches the function 'Off' again. |
| 1 | Basic setting 'On': A permanent positive signal switches this function 'Off', which is connected to the digital input. Removal of the signal switches the function 'On' again. |
| 2 | Push-button function. Basic setting 'Off'. Only positive signals are effective. The first positive signal switches 'On'. Removal of the signal is necessary before the next positive signal can switch 'Off'. |

## Signal

| Name | r/w | Adr. | Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| St.Di | r | base<br>1dP<br>2dP<br>3dP | 1070<br>9262<br>17454<br>25646 | 34908 | Int | 0...7 | ☑ | Status of the digital inputs or of push-buttons (binary coded).<br>Bit 0 Input 1<br>Bit 1 Input 2<br>Bit 2 Input 3<br>Bit 8 Status of 'F' key<br>Bit 9 Status of 'A/M' key<br>Bit 10 Status of 'Sel' key<br>Bit 11 Status of 'Down' key<br>Bit 12 Status of 'Up' key<br>Bit 13 Status of 'Loc' key |
| L-R | r/w | base<br>1dP<br>2dP<br>3dP | 1080<br>9272<br>17464<br>25656 | 34928 | Int | 0...1 | ☐ | Remote operation. Remote means that all values can only be adjusted via the interface. Adjustments via the front panel are blocked. |
| W_W2 | r/w | base<br>1dP<br>2dP<br>3dP | 1081<br>9273<br>17465<br>25657 | 34930 | Int | 0...1 | ☐ | Signal for activating the second (safety) setpoint (SP.2= W2.<br>Note: Setpoint W2 is not restricted by the setpoint limits! |
| Wi_We | r/w | base<br>1dP<br>2dP<br>3dP | 1082<br>9274<br>17466<br>25658 | 34932 | Int | 0...1 | ☐ | Signal for activating the external setpoint value. SP.E is the external setpoint, or dependent on the device and configuration of the setpoint shift. |
| Y_Y2 | r/w | base<br>1dP<br>2dP<br>3dP | 1083<br>9275<br>17467<br>25659 | 34934 | Int | 0...1 | ☐ | Signal for activating the 2nd output value Y2. With selected Y2, the output is operated as a positioner.Caution: Do not confuse the parameter 'fixed output Y2' with the controller output Y2! |
| Y_Y.E | r/w | base<br>1dP<br>2dP<br>3dP | 1084<br>9276<br>17468<br>25660 | 34936 | Int | 0...1 | ☐ | Signal for activating the external positioning value. The controller is operated as positioner. |

# 8 LOGI

## • Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| A-M | r/w | base<br>1dP<br>2dP<br>3dP | 1085<br>9277<br>17469<br>25661 | 34938 | Int | 0...1 | ☐ | Signal for activating manual operation. In the manual mode, the controller provides output signals independent of the process. |
| C.Off | r/w | base<br>1dP<br>2dP<br>3dP | 1086<br>9278<br>17470<br>25662 | 34940 | Int | 0...1 | ☐ | Signal for disabling all the controller outputs.<br>Note: Forcing has priority; alarm processing remains active. |
| L.AM | r/w | base<br>1dP<br>2dP<br>3dP | 1087<br>9279<br>17471<br>25663 | 34942 | Int | 0...1 | ☐ | Signal for disabling manual operation. Triggers a forced switchover to automatic mode, and disables the front panel A/M key (also if other functions have been assigned to the key). |
| Err.r | r/w | base<br>1dP<br>2dP<br>3dP | 1088<br>9280<br>17472<br>25664 | 34944 | Int | 0...1 | ☐ | Signal for resetting the entire error list. The error list contains all errors that are reported, e.g. device faults and limit values. It also contains queued as well as stored errors after their correction. The reset acknowledges all errors, whereby queued errors will reappear after the next error detection (measurement). |
| SSR.Res | r/w | base<br>1dP<br>2dP<br>3dP | 1089<br>9281<br>17473<br>25665 | 34946 | Int | 0...1 | ☐ | Reset of the alarm triggered by a solid-state relay (SSR).<br>SSRs are mostly used for frequent switching of heating elements, because they have no mechanical contacts that can wear out. However, an unnoticed short circuit could lead to overheating of the machine. |
| Boost | r/w | base<br>1dP<br>2dP<br>3dP | 1090<br>9282<br>17474<br>25666 | 34948 | Int | 0...1 | ☐ | Signal for activating the boost function. The boost function causes a brief setpoint increase, which is used e.g. to clear blocked channels ('frozen' material) in a hot-runner system. |
| Set1.2 | r/w | base<br>1dP<br>2dP<br>3dP | 1091<br>9283<br>17475<br>25667 | 34950 | Int | 0...1 | ☐ | Switch-over of parameter set. The 2nd parameter set contains one complete set each of Pb (= proportional band), ti (= integral action time), and td (= derivative action time) for heating and for cooling. All other control parameters, such as switching duty cycles, are valid for both parameter sets. |
| Prg.R.S | r/w | base<br>1dP<br>2dP<br>3dP | 1092<br>9284<br>17476<br>25668 | 34952 | Int | 0...1 | ☐ | Signal for starting the programmer. On units with a simple programmer (only 1 program), a stop immediately causes a reset, followed by a new start. With units that have been defined as program controllers (several programs), the program is stopped, and then continued. |
| Prg.Res | r/w | base<br>1dP<br>2dP<br>3dP | 1093<br>9285<br>17477<br>25669 | 34954 | Int | 0...1 | ☐ | Programmer reset switches the programmer off, and sets it back to the starting condition. Reset stops the currently active program, and activates the internal setpoint. A newly selected program becomes the active program. |

# 8 LOGI

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| F.Di | r/w | base<br>1dP<br>2dP<br>3dP | 1094<br>9286<br>17478<br>25670 | 34956 | Int | 0...7 | ☐ | Forcing of digital inputs. Forcing involves the external operation of at least one input. The instrument takes over this input value (preset value for inputs from a superordinate system, e.g. for a function test.) |

Bit 0 Forcing of digital input 1
Bit 1 Forcing of digital input 2
Bit 2 Forcing of digital input 3
Bit 3 Forcing of digital input 4
Bit 4 Forcing of digital input 5

# 9 ohnE

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Conf | r/w | base<br>1dP<br>2dP<br>3dP | 1<br>8193<br>16385<br>24577 | 32770 | Int | 0...2 | ☐ | Start/Stop and abortion of the configuration mode<br>0 = End of configuration<br>1 = Start of configuration<br>2 = Abort configuration |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| UPD | r/w | base<br>1dP<br>2dP<br>3dP | 95<br>8287<br>16479<br>24671 | 32958 | Enum | *Enum_Aenderungsflag* | | Status message indicating that parameter / configuration have been changed via the front panel. |

| | |
|---|---|
| 0 | No change via the front panel keys. |
| 1 | A change has been made via the front panel keys, which must be processed. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Hw.Opt | r | base<br>1dP<br>2dP<br>3dP | 200<br>8392<br>16584<br>24776 | 33168 | Int | 0...65535 | ☐ | KSx-1-devices hardware option<br>0000 WXYZ 0000 00BA<br>Z=1: Option Modbus + di2/di3 + TPS<br>Y=1: Option INP3 (KS90-1, KS90-1P)<br>X=1: Option 16 programs (KS90-1P)<br>W=1: Option OUT5/OUT6 (KS50-1, KS90-1, KS90-1P)<br>A=1: OUT3 is analogue output<br>B=1: OUT4 is analogue output (KS90-1, KS90-1P) |
| Sw.Op | r | base<br>1dP<br>2dP<br>3dP | 201<br>8393<br>16585<br>24777 | 33170 | Int | 0...255 | ☐ | Software version XY Major and Minor Release (e.g. 21 = Version 2.1). The software version specifies the firmware in the unit. For the correct interaction of E-Tool and device, it must match the operating version (OpVersion) in the E-Tool. |
| Bed.V | r | base<br>1dP<br>2dP<br>3dP | 202<br>8394<br>16586<br>24778 | 33172 | Int | 0...255 | ☐ | Operating version (numeric value). For the correct interaction of E-Tool and device, the software version and operating version must match. |
| Unit | r | base<br>1dP<br>2dP<br>3dP | 203<br>8395<br>16587<br>24779 | 33174 | Int | 0...255 | ☐ | Identification of the device. |

# 9 ohnE

## Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| S.Vers | r | base 204<br>1dP 8396<br>2dP 16588<br>3dP 24780 | 33176 | Int | 100...255 | ☐ | The sub-version number is given as an additional index for precise definition of software version. |
| Uident | r | base 910<br>1dP 9102<br>2dP 17294<br>3dP 25486 | 34588 | Text | ... | ☐ | Device identification. Via this Modbus address, up to 14 data units (28 bytes) can be defined.<br>Bytes 1 - 15 order number of the device<br>Bytes 16 - 19 Ident number 1<br>Bytes 20 + 21 Ident number 2<br>Bytes 22 - 25 OEM number<br>Bytes 26 - 28 Software order number |
| St.Ala | r | base 250<br>1dP 8442<br>2dP 16634<br>3dP 24826 | 33268 | Int | 0...31 | ☐ | Alarm status: Bit-wise coded status of the individual alarms, e.g. exceeded limit value or Loop. |

Bit 0 Existing/stored exceeded limit 1
Bit 1 Existing/stored exceeded limit 2
Bit 2 Existing/stored exceeded limit 3
Bit 3 Not used
Bit 4 Existing/stored loop alarm
Bit 5 Existing/stored heating current alarm
Bit 6 Existing/stored SSR alarm
Bit 7 Not used
Bit 8 Existing exceeded limit 1
Bit 9 Existing exceeded limit 2
Bit 10 Existing exceeded limit 3
Bit 11 Not used
Bit 12 Existing loop alarm
Bit 13 Existing heating current alarm
Bit 14 Existing SSR alarm
Bit 15 Not used

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| St.Do | r | base 251<br>1dP 8443<br>2dP 16635<br>3dP 24827 | 33270 | Int | 0...31 | ☐ | Status of the digital outputs<br>Bit 0 digital output 1<br>Bit 1 digital output 2<br>Bit 2 digital output 3<br>Bit 3 digital output 4<br>Bit 4 digital output 5<br>Bit 5 digital output 6 |
| St.Ain | r | base 252<br>1dP 8444<br>2dP 16636<br>3dP 24828 | 33272 | Int | 0...7 | ☐ | Bit-coded status of the analog input (fault, e.g. short circuit) |

Bit 0 Break at Input 1
Bit 1 Reversed polarity at Input 1
Bit 2 Short circuit at Input 1
Bit 3 Not used
Bit 4 Break at Input 2
Bit 5 Reversed polarity at Input 2
Bit 6 Short-circuit at Input 2
Bit 7 Not used
Bit 8 Break at Input 3 (only KS 90)
Bit 9 Reversed polarity at Input 3 (only KS 90)
Bit 10 Short-circuit at Input 3 (only KS 90)
Bit 11 Not used

## 9 ohnE

### Signal

| Name | r/w | Adr. | Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| St.Di | r | base<br>1dP<br>2dP<br>3dP | 253<br>8445<br>16637<br>24829 | 33274 | Int | 0...7 | ☐ | Status of the digital inputs or of push-buttons (binary coded). |
| | | | | | | Bit 0 Input 1<br>Bit 1 Input 2<br>Bit 2 Input 3<br>Bit 8 Status of 'F' key<br>Bit 9 Status of 'A/M' key<br>Bit 10 Status of 'Sel' key<br>Bit 11 Status of 'Down' key<br>Bit 12 Status of 'Up' key<br>Bit 13 Status of 'Loc' key | | |
| F.Di | r/w | base<br>1dP<br>2dP<br>3dP | 303<br>8495<br>16687<br>24879 | 33374 | Int | 0...1 | ☐ | Forcing of digital inputs. Forcing involves the external operation of at least one input. The instrument takes over this input value (preset value for inputs from a superordinate system, e.g. for a function test.) |
| | | | | | | Bit 0 Forcing of digital input 1<br>Bit 1 Forcing of digital input 2<br>Bit 2 Forcing of digital input 3<br>Bit 3 Forcing of digital input 4<br>Bit 4 Forcing of digital input 5 | | |
| F.Do | r/w | base<br>1dP<br>2dP<br>3dP | 304<br>8496<br>16688<br>24880 | 33376 | Int | 0...15 | ☐ | Forcing of digital outputs. Forcing involves the external operation of at least one output. The instrument has no influence on this output (use of free outputs by superordinate system). |
| Cmodules | r | base<br>1dP<br>2dP<br>3dP | 911<br>9103<br>17295<br>25487 | 34590 | Text | ... | ☐ | In the device connected modules. If an error E4 occur this data should be compared with the code number of the Uident . |

## 10 ohnE1

### Signal

| Name | r/w | Adr. | Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| In.? | r | base<br>1dP<br>2dP<br>3dP | 232<br>8424<br>16616<br>24808 | 33232 | Float | -1999...9999 | ☐ | Measurement value after the measurement value correction (e.g. with offset or 2-point correction, and scaling). |
| In.?r | r | base<br>1dP<br>2dP<br>3dP | 240<br>8432<br>16624<br>24816 | 33248 | Float | -1999...9999 | ☐ | Measurement value before the measurement value correction (unprocessed). |
| F.Inp | r/w | base<br>1dP<br>2dP<br>3dP | 300<br>8492<br>16684<br>24876 | 33368 | Float | -1999...9999 | ☐ | Forcing the value for an analog input INP. Forcing involves the external operation of an input. The instrument takes over the value at this input like a measurement value (preset value for inputs from a superordinate system, e.g. for a function test.) |

## 11 ohnE2

### Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| In.? | r | base 233<br>1dP 8425<br>2dP 16617<br>3dP 24809 | 33234 | Float | -1999...9999 ☐ | Measurement value after the measurement value correction (e.g. with offset or 2-point correction, and scaling). |
| In.?r | r | base 241<br>1dP 8433<br>2dP 16625<br>3dP 24817 | 33250 | Float | -1999...9999 ☐ | Measurement value before the measurement value correction (unprocessed). |
| F.Inp | r/w | base 301<br>1dP 8493<br>2dP 16685<br>3dP 24877 | 33370 | Float | -1999...9999 ☐ | Forcing the value for an analog input INP. Forcing involves the external operation of an input. The instrument takes over the value at this input like a measurement value (preset value for inputs from a superordinate system, e.g. for a function test.) |

## 12 ohnE3

### Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| F.Out? | r/w | base 305<br>1dP 8497<br>2dP 16689<br>3dP 24881 | 33378 | Float | 0...120 ☐ | Forcing value of the analog output. Forcing involves the external operation of an output, i.e. the instrument has influence on this output. (Used for the operation of free outputs e.g. by a supervisory PLC.) |

## 13 ohnE4

### Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| F.Out? | r/w | base 306<br>1dP 8498<br>2dP 16690<br>3dP 24882 | 33380 | Float | 0...120 ☐ | Forcing value of the analog output. Forcing involves the external operation of an output, i.e. the instrument has influence on this output. (Used for the operation of free outputs e.g. by a supervisory PLC.) |

## 14 othr

### ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| D2.Err | r/w | base 193<br>1dP 8385<br>2dP 16577<br>3dP 24769 | 33154 | Enum | *Enum_Disp2E* | Queued faults can be displayed directly in the 2nd line of the display. In case of a fault, the display then alternates between the value of the lower display line (standard = setpoint) and the error message for the fault with the highest priority (blinking display). |

| | |
|---|---|
| 0 | Display line 2 is not switched over in case of a fault. The fault is signalled via the LED, and the error message is shown in the error list. |
| 1 | In case of a fault, display line 2 alternates between the error message and the value of the lower display line- The fault with the highest priority is displayed as long as it is present. Latched (stored) faults must be acknowledged in order to remove them from the display. |

# 14 othr

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Coff | r/w | base | 192 | 33152 | Enum | *Enum_Coff* | The standard disabling procedure only switches off the controller outputs, whereby the alarms, displays, and other functions remain active. Alternatively, all functions can be switched off (including alarms and displays). |
| | | 1dP | 8384 | | | | |
| | | 2dP | 16576 | | | | |
| | | 3dP | 24768 | | | | |

| | |
|---|---|
| 0 | Only the PID controller functions are disabled. The analog controller outputs have the value 0.0, and the switching outputs generate the logical state FALSE. All other functions, e.g. alarms and displays, continue operating in the normal manner. |
| 1 | All the controller functions are disabled. The analog outputs have the value 0.0, and the switching outputs generate the logical state FALSE. If configured, an inversion is carried out. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| bAud | r/w | base | 180 | 33128 | Enum | *Enum_Baud* | Bit rate of the interface (only visible with OPTION). The bit rate determines the transmission speed. |
| | | 1dP | 8372 | | | | |
| | | 2dP | 16564 | | | | |
| | | 3dP | 24756 | | | | |

| | |
|---|---|
| 0 | 2400 Baud |
| 1 | 4800 Baud |
| 2 | 9600 Baud |
| 3 | 19200 Baud |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Addr | r/w | base | 181 | 33130 | Int | 1...247 ☐ | Address on the interface (only visible with OPTION) |
| | | 1dP | 8373 | | | | |
| | | 2dP | 16565 | | | | |
| | | 3dP | 24757 | | | | |
| PrtY | r/w | base | 182 | 33132 | Enum | *Enum_Parity* | Parity of data on the interface (only visible with OPTION). Simple possibility of checking that transferred data is correct. |
| | | 1dP | 8374 | | | | |
| | | 2dP | 16566 | | | | |
| | | 3dP | 24758 | | | | |

| | |
|---|---|
| 0 | no parity, 2 stop bits |
| 1 | Even parity |
| 2 | Odd parity |
| 3 | no parity with 1 stopbit |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| dELY | r/w | base | 183 | 33134 | Int | 0...200 ☐ | Response delay [ms] (only visible with OPTION). Additional delay time before the received message may be answered on the Modbus. (Might be necessary, if the same line is used for transmit/receive.) |
| | | 1dP | 8375 | | | | |
| | | 2dP | 16567 | | | | |
| | | 3dP | 24759 | | | | |
| Unit | r/w | base | 170 | 33108 | Enum | *Enum_Unit* | Physical unit (temperature), f.e.°C |
| | | 1dP | 8362 | | | | |
| | | 2dP | 16554 | | | | |
| | | 3dP | 24746 | | | | |

| | |
|---|---|
| 0 | without unit |
| 1 | °C |
| 2 | °F |

## 14 othr

### ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| dP | r/w | base<br>1dP<br>2dP<br>3dP | 171<br>8363<br>16555<br>24747 | 33110 | Enum | *Enum_dP* | | Decimal point (max. no of decimals). Format of the measured value display. |

| | |
|---|---|
| 0 | No digit behind the decimal point |
| 1 | One digit behind the decimal point |
| 2 | Two digits behind the decimal point |
| 3 | Three digits behind the decimal point |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| LEd | r/w | base<br>1dP<br>2dP<br>3dP | 190<br>8382<br>16574<br>24766 | 33148 | Enum | *Enum_Led* | | Meaning of the signalling LEDs. Selection of a combination of the displayable signals. |

| | |
|---|---|
| 0 | The digital outputs OUT1, OUT2, and OUT3 are displayed. |
| 1 | Display of controller output y1 (heating / open), alarm2, and alarm3. |
| 2 | Display of controller output y1 (heating / open), controller output y2 (cooling / close), alarm3 |
| 3 | Display of controller output y2 (cooling / close), controller output y1 (heating / open), alarm3 |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| C.dEL | r/w | base<br>1dP<br>2dP<br>3dP | 184<br>8376<br>16568<br>24760 | 33136 | Int | 0...200 | ☐ | For both interfaces, Modbus only. Additional acceptable delay time between 2 received bytes, before "end of message" is assumed. This time is needed if data is not transmitted continousely by the modem. |
| FrEq | r/w | base<br>1dP<br>2dP<br>3dP | 150<br>8342<br>16534<br>24726 | 33068 | Enum | *Enum_FrEq* | | Switchover of the applied mains frequency 50 / 60 Hz, thereby better adaptation of the input filter for hum suppression. |

| | |
|---|---|
| 0 | mains frequency 50 Hz |
| 1 | mains frequency 60 Hz |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| MASt | r/w | base<br>1dP<br>2dP<br>3dP | 185<br>8377<br>16569<br>24761 | 33138 | Enum | *Enum_MASt* | | Device works as Modbus master.<br>The communication is executed according to the master/slave principle, whereby the device can be operated as master or as slave. Operation as master must be configured here. |

| | |
|---|---|
| 0 | The unit is operated as a Modbus slave. |
| 1 | The unit is operated as a Modbus master. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Cycl | r/w | base<br>1dP<br>2dP<br>3dP | 186<br>8378<br>16570<br>24762 | 33140 | Int | 0...200 | ☐ | Cycle time (in seconds) during which the Modbus master transmits its message on the bus. |
| AdrO | r/w | base<br>1dP<br>2dP<br>3dP | 187<br>8379<br>16571<br>24763 | 33142 | Int | 1...65535 | ☐ | Target address to which the data specified with AdrU are output on the bus. |

# 14 othr

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| AdrU | r/w | base | 188 | 33144 | Int | 1...65535 | ☐ | Modbus address of the data output on the bus by the Modbus master. |
| | | 1dP | 8380 | | | | | |
| | | 2dP | 16572 | | | | | |
| | | 3dP | 24764 | | | | | |
| Numb | r/w | base | 189 | 33146 | Int | 0...100 | ☐ | Quantity of data that are to be transmitted from the Modbus master. |
| | | 1dP | 8381 | | | | | |
| | | 2dP | 16573 | | | | | |
| | | 3dP | 24765 | | | | | |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| E.1 | r/w | base | 210 | 33188 | Enum | *Defect* | Err 1 (internal error)<br>Contact Service. |
| | | 1dP | 8402 | | | | |
| | | 2dP | 16594 | | | | |
| | | 3dP | 24786 | | | | |

| | |
|---|---|
| 0 | no fault exists (Reset). |
| 2 | The device is defective. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| E.2 | r/w | base | 211 | 33190 | Enum | *Problem* | Err 2 (internal error, resettable)<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8403 | | | | |
| | | 2dP | 16595 | | | | |
| | | 3dP | 24787 | | | | |

| | |
|---|---|
| 0 | no fault, resetting possible (Reset). |
| 1 | A fault has occurred and has been stored. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FbF.1 | r/w | base | 212 | 33192 | Enum | *Break* | Sensor break at input INP1.<br>Typical causes and suggested remedies:<br>Sensor fault: replace INP1 sensor.<br>Wiring fault: check connections of INP1.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8404 | | | | |
| | | 2dP | 16596 | | | | |
| | | 3dP | 24788 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the sensor break alarm possible (Reset). |
| 1 | The sensor fault alarm has been triggered and stored; the fault is no longer present. The operator must acknowledge the error message in order to delete it from the error list. |
| 2 | Sensor break: The sensor is defective or there is a wiring fault. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Sht.1 | r/w | base | 213 | 33194 | Enum | *Short* | Short circuit at input INP1.<br>Typical causes and suggested remedies:<br>Sensor fault: replace INP1 sensor.<br>Wiring fault: check connections of INP1.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8405 | | | | |
| | | 2dP | 16597 | | | | |
| | | 3dP | 24789 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the short-circuit alarm possible (Reset). |
| 1 | A short-circuit fault has occurred and has been stored. |
| 2 | A short-circuit fault has occurred. |

# 14 othr

## Signal

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| POL.1 | r/w | base | 214 | 33196 | Enum | *Polarity* | Incorrect polarity at input INP1.<br>Suggested remedy: reverse the polarity at INP1.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8406 | | | | |
| | | 2dP | 16598 | | | | |
| | | 3dP | 24790 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the incorrect polarity alarm possible (Reset). |
| 1 | An incorrect polarity fault has occurred and has been stored. |
| 2 | Incorrect polarity. The wiring of the input circuit is not correct. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FbF.2 | r/w | base | 215 | 33198 | Enum | *Break* | Sensor break at input INP2.<br>Typical causes and suggested remedies:<br>Sensor fault: replace INP2 sensor.<br>Wiring fault: check connections of INP2.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8407 | | | | |
| | | 2dP | 16599 | | | | |
| | | 3dP | 24791 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the sensor break alarm possible (Reset). |
| 1 | The sensor fault alarm has been triggered and stored; the fault is no longer present. The operator must acknowledge the error message in order to delete it from the error list. |
| 2 | Sensor break: The sensor is defective or there is a wiring fault. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Sht.2 | r/w | base | 216 | 33200 | Enum | *Short* | Short circuit at input INP2.<br>Typical causes and suggested remedies:<br>Sensor fault: replace INP2 sensor.<br>Wiring fault: check connections of INP2.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8408 | | | | |
| | | 2dP | 16600 | | | | |
| | | 3dP | 24792 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the short-circuit alarm possible (Reset). |
| 1 | A short-circuit fault has occurred and has been stored. |
| 2 | A short-circuit fault has occurred. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| POL.2 | r/w | base | 217 | 33202 | Enum | *Polarity* | Incorrect polarity at input INP2.<br>Suggested remedy: reverse the polarity at INP2.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8409 | | | | |
| | | 2dP | 16601 | | | | |
| | | 3dP | 24793 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the incorrect polarity alarm possible (Reset). |
| 1 | An incorrect polarity fault has occurred and has been stored. |
| 2 | Incorrect polarity. The wiring of the input circuit is not correct. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HCA | r/w | base | 218 | 33204 | Enum | *HeatCurr* | Heating current alarm.Possible fault s are an open heating current circuit with current I < heating current lim or current I > heating current limit (depending on configuration), or defective heater band.Suggested remedy: check heating current circuit, replace heater band if necessary.<br>(As a process value via fieldbus interface not writable!) |
| | | 1dP | 8410 | | | | |
| | | 2dP | 16602 | | | | |
| | | 3dP | 24794 | | | | |

| | |
|---|---|
| 0 | no fault, resetting of the heating current alarm possible (Reset). |
| 1 | A heating current fault has occurred and has been stored. |

# 14 othr

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SSr | r/w | base<br>1dP<br>2dP<br>3dP | 219<br>8411<br>16603<br>24795 | 33206 | Enum | *Short* | Alarm message: SSr<br>Possible causes: a current flow in the heating circuit although controller is 'off', or the SSR is defective.<br>Suggested remedy: check heating current circuit, replace the solid-state relay, if necessary.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | no fault, resetting of the short-circuit alarm possible (Reset). |
| 1 | A short-circuit fault has occurred and has been stored. |
| 2 | A short-circuit fault has occurred. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| LooP | r/w | base<br>1dP<br>2dP<br>3dP | 220<br>8412<br>16604<br>24796 | 33208 | Enum | *LoopAlarm* | Alarm message: LooP<br>Possible causes: faulty or incorrectly connected input circuit, or output not connected correctly.<br>Suggested remedy: check heating or cooling circuit, check sensor function and replace if necessary, check controller and output switching actuator.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | no fault, resetting of the loop alarm possible (Reset). |
| 1 | A control loop fault has occurred and has been stored. |
| 2 | A control loop fault has occurred, there was no clear process response following a step change of the output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| AdA.H | r/w | base<br>1dP<br>2dP<br>3dP | 221<br>8413<br>16605<br>24797 | 33210 | Enum | *Tune* | Error message from "heating" self-tuning and reason for aborted tuning attempt.<br>Hints for trouble-shooting: Check operating sense of actuator. Is the loop closed? Is there an output limit? Adapt the setpoint. Increase step output for Yopt.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | no error |
| 3 | Process responds in the wrong direction.<br>Possible remedy: Check the output signal sense (inverse <-> direct), and re-configure the controller if necessary (inverse <-> direct). |
| 4 | No response from the process. Perhaps the control loop is open.<br>Possible remedy: Check sensor, connections, and process. |
| 5 | The process value turning point of the step response is too low.<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 6 | Self-tuning was aborted due to the risk of an exceeded setpoint.<br>Possible remedy: Repeat the attempt with an increased setpoint reserve. |
| 7 | The step output change is not large enough (minimum change > 5 %).<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 8 | Setpoint reserve must be given before generating the step output change.<br>Possible remedy: decrease setpoint range, change setpoint, or change process value. |
| 9 | The pulse response attempt has failed. No useful parameters were determined. Perhaps the control loop is open.<br>Possible remedy: Check sensor, connections, and process. |

# 14 othr

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| AdA.C | r/w | base<br>1dP<br>2dP<br>3dP | 222<br>8414<br>16606<br>24798 | 33212 | Enum | *Tune* | Error message from "cooling" self-tuning and reason for aborted tuning attempt.<br>Hints for trouble-shooting: Check operating sense of actuator. Is the loop closed? Is there an output limit? Adapt the setpoint. Increase step output for Yopt.<br>(As a process value via fieldbus interface not writable!) |

| Value | Description |
|---|---|
| 0 | no error |
| 3 | Process responds in the wrong direction.<br>Possible remedy: Check the output signal sense (inverse <-> direct), and re-configure the controller if necessary (inverse <-> direct). |
| 4 | No response from the process. Perhaps the control loop is open.<br>Possible remedy: Check sensor, connections, and process. |
| 5 | The process value turning point of the step response is too low.<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 6 | Self-tuning was aborted due to the risk of an exceeded setpoint.<br>Possible remedy: Repeat the attempt with an increased setpoint reserve. |
| 7 | The step output change is not large enough (minimum change > 5 %).<br>Possible remedy: Increase the permitted step output range, i.e. increase the parameter Y.Hi ('heating') or reduce the parameter Y.Lo ('cooling'). |
| 8 | Setpoint reserve must be given before generating the step output change.<br>Possible remedy: decrease setpoint range, change setpoint, or change process value. |
| 9 | The pulse response attempt has failed. No useful parameters were determined. Perhaps the control loop is open.<br>Possible remedy: Check sensor, connections, and process. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.1 | r/w | base<br>1dP<br>2dP<br>3dP | 223<br>8415<br>16607<br>24799 | 33214 | Enum | *Limit* | Limit value 1 exceeded.<br>Hint for trouble-shooting: check the process.<br>(As a process value via fieldbus interface not writable!) |

| Value | Description |
|---|---|
| 0 | no fault, resetting of the limit value alarm possible (Reset). |
| 1 | The limit value has been exceeded, and the fault has been stored. |
| 2 | The limit value has been exceeded; the monitored (measurement) value is outside the set limits. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.2 | r/w | base<br>1dP<br>2dP<br>3dP | 224<br>8416<br>16608<br>24800 | 33216 | Enum | *Limit* | Limit value 2 exceeded.<br>Hint for trouble-shooting: check the process.<br>(As a process value via fieldbus interface not writable!) |

| Value | Description |
|---|---|
| 0 | no fault, resetting of the limit value alarm possible (Reset). |
| 1 | The limit value has been exceeded, and the fault has been stored. |
| 2 | The limit value has been exceeded; the monitored (measurement) value is outside the set limits. |

# 14 othr

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.3 | r/w | base<br>1dP<br>2dP<br>3dP | 225<br>8417<br>16609<br>24801 | 33218 | Enum | *Limit* | Limit value 3 exceeded.<br>Hint for trouble-shooting: check the process.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | no fault, resetting of the limit value alarm possible (Reset). |
| 1 | The limit value has been exceeded, and the fault has been stored. |
| 2 | The limit value has been exceeded; the monitored (measurement) value is outside the set limits. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| InF.1 | r/w | base<br>1dP<br>2dP<br>3dP | 226<br>8418<br>16610<br>24802 | 33220 | Enum | *Time* | Message from the operating hours counter that the preset no. of hours for this maintenance period has been reached. The op-hours counter for the maintenance period is reset when this message is acknowledged. Counting the operating hours is used for preventive maintenance. - Acknowledge the error to reset it.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | No signal, resetting of the time limit signal possible (Reset). |
| 1 | Operating hours - limit value (maintenance period) reached: please acknowledge. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| InF.2 | r/w | base<br>1dP<br>2dP<br>3dP | 227<br>8419<br>16611<br>24803 | 33222 | Enum | *Switch* | Message from the switching cycle counter that the preset no. of switch cycles for this maintenance period has been reached. The cycle counter for the maintenance period is reset when this message is acknowledged. Counting the switching cycles is used for preventive maintenance. - Acknowledge the error to reset it.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | No error message, resetting of the switching cycle counter possible (Reset). |
| 1 | Set limit of the switching cycle counter (maintenance period) has been reached: please acknowledge. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| E.4 | r/w | base<br>1dP<br>2dP<br>3dP | 228<br>8420<br>16612<br>24804 | 33224 | Enum | *Problem* | Hardware fault.Cause: Code number and hardware are not identical.<br>Remedy: Contact Service.<br>(As a process value via fieldbus interface not writable!) |

| | |
|---|---|
| 0 | no fault, resetting possible (Reset). |
| 1 | A fault has occurred and has been stored. |

# 15 Out.1

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.Act | r/w | base | 4150 | 41068 | Enum | *Enum_OAct* | Operating sense of the switching output.<br>Direct: Active function (e.g. limit value) switches the output ON;<br>Inverse: Active function (e.g. limit value) switches the output OFF. |
| | | 1dP | 12342 | | | | |
| | | 2dP | 20534 | | | | |
| | | 3dP | 28726 | | | | |

| | |
|---|---|
| 0 | Direct / Normally de-energized mode |
| 1 | inverse / normally closed |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.tYP | r/w | base | 4170 | 41108 | Enum | *Enum_OtYP* | Signal type selection OUT |
| | | 1dP | 12362 | | | | |
| | | 2dP | 20554 | | | | |
| | | 3dP | 28746 | | | | |

| | |
|---|---|
| 0 | relay/logic |
| 1 | 0 ... 20 mA continuous |
| 2 | 4 ... 20 mA continuous |
| 3 | 0...10 V continuous |
| 4 | 2...10 V continuous |
| 5 | Transmitter supply |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.1 | r/w | base | 4151 | 41070 | Enum | *Enum_Y1* | Output function: Controller output Y1 |
| | | 1dP | 12343 | | | | |
| | | 2dP | 20535 | | | | |
| | | 3dP | 28727 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.2 | r/w | base | 4152 | 41072 | Enum | *Enum_Y2* | Output function: Controller output Y2. Caution: Do not confuse the controller output Y2 with the parameter 'Fixed output Y2' ! |
| | | 1dP | 12344 | | | | |
| | | 2dP | 20536 | | | | |
| | | 3dP | 28728 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y2. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.1 | r/w | base | 4153 | 41074 | Enum | *Enum_Lim1* | Output function: Signal limit 1 |
| | | 1dP | 12345 | | | | |
| | | 2dP | 20537 | | | | |
| | | 3dP | 28729 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.2 | r/w | base | 4154 | 41076 | Enum | *Enum_Lim2* | Output function: Signal limit 2 |
| | | 1dP | 12346 | | | | |
| | | 2dP | 20538 | | | | |
| | | 3dP | 28730 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 2. |

# 15 Out.1

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4155<br>12347<br>20539<br>28731 | 41078 | Enum | *Enum_Lim3* | Output function: Signal limit 3 |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 3. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| LP.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4157<br>12349<br>20541<br>28733 | 41082 | Enum | *Enum_OUT_LPAL* | Output function: Signal Interruption alarm (LOOP)<br>The overall control loop is monitored and the process value has to change with an output signal of maximum value, else loop alarm is generated. |

| | |
|---|---|
| 0 | not active |
| 1 | This loop alarm (= open loop alarm) is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HC.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4158<br>12350<br>20542<br>28734 | 41084 | Enum | *Enum_OUT_HCAL* | Output function: Signal Heat current alarm. Either break (= current I < heating current limit) can be monitored or overload (= current I > heating current limit), dependent on configuration. |

| | |
|---|---|
| 0 | not active |
| 1 | The heating current alarm is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HC.SC | r/w | base<br>1dP<br>2dP<br>3dP | 4159<br>12351<br>20543<br>28735 | 41086 | Enum | *Enum_HCSC* | Output function: Signal Solid-state relay (SSR) short circuit.<br>The short circuit alarm of the SSR is triggered, if a current is detected in the heating circuit, although the controller output is switched off. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an SSR fault. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| timE | r/w | base<br>1dP<br>2dP<br>3dP | 4160<br>12352<br>20544<br>28736 | 41088 | Enum | *Enum_time* | Output function: Signal Timer running.<br>This message is generated by the setpoint processing, if a timer mode has been configured, and the time has elapsed. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the timer status |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| t.End | r/w | base<br>1dP<br>2dP<br>3dP | 4176<br>12368<br>20560<br>28752 | 41120 | Enum | *Enum_TEnd* | Output function: Signal Timer end.<br>This message is available when the timer has been completed (only when configured as a timer). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Timer end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| P.End | r/w | base<br>1dP<br>2dP<br>3dP | 4161<br>12353<br>20545<br>28737 | 41090 | Enum | *Enum_PEnd* | Output function: Signal Program end.<br>This message is available when the program has been completed (only when configured as a program controller). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Program end'. |

# 15 Out.1

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4162<br>12354<br>20546<br>28738 | 41092 | Enum | *Enum_FAi1* | Output function: Signal INP1 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP1. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP1 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4163<br>12355<br>20547<br>28739 | 41094 | Enum | *Enum_FAi2* | Output function: Signal INP2 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP2. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP2 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4165<br>12357<br>20549<br>28741 | 41098 | Enum | *Enum_PrG1* | Output function: Signal programmer's control output no. 1.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 1 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4166<br>12358<br>20550<br>28742 | 41100 | Enum | *Enum_PrG2* | Output function: Signal programmer's  control output no 2.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 2 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4167<br>12359<br>20551<br>28743 | 41102 | Enum | *Enum_PrG3* | Output function: Signal programmer's control output no. 3. T<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 3 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.4 | r/w | base<br>1dP<br>2dP<br>3dP | 4168<br>12360<br>20552<br>28744 | 41104 | Enum | *Enum_PrG4* | Output function: Signal programmer's  control output no 4.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 4 is assigned to this output. |

# 15 Out.1

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| CALL | r/w | base<br>1dP<br>2dP<br>3dP | 4169<br>12361<br>20553<br>28745 | 41106 | Enum | *Enum_CALL* | Output: Operator call.<br>At the end of a program segment, a contact is set, e.g. for an acoustic signal. This indicates to the operator that a certain program status has been reached, and operator action is required. Operator calling is used, if the program may only be continued after a check or some kind of operator action. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is switched by an operator call. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out.0 | r/w | base<br>1dP<br>2dP<br>3dP | 4171<br>12363<br>20555<br>28747 | 41110 | Float | -1999...9999 ☐ | Lower scaling limit of the analog output (corresponds to 0%). If current and voltage signals are used as output values, the display can be scaled to the output value in the Parameter Level. The output value of the lower scaling point is indicated in the respective electrical unit (mA / V). |
| Out.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4172<br>12364<br>20556<br>28748 | 41112 | Float | -1999...9999 ☐ | Upper scaling limit of the analog output (corresponds to 100%). If current or voltage signals are used as output values, scaling of the display can be applied to the output value by means of the Parameter Level. Definition of the upper output limit is done using the corresponding electrical value (mA / V). |
| O.Src | r/w | base<br>1dP<br>2dP<br>3dP | 4173<br>12365<br>20557<br>28749 | 41114 | Enum | *Enum_OSrc* | Signal source of the analog output. |

| | |
|---|---|
| 0 | not active |
| 1 | controller output y1 (cont.) |
| 2 | controller output y2 (cont.) |
| 3 | process value |
| 4 | effective setpoint Weff |
| 5 | Control deviation xw (process value - setpoint)= relative alarm<br>Note: Monitoring with the effective setpoint Weff. For example using a ramp it is the changing setpoint, not the target setpoint of the ramp. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out? | r | base<br>1dP<br>2dP<br>3dP | 4180<br>12372<br>20564<br>28756 | 41128 | Enum | *Enum_Ausgang* | Status of the digital output |

| | |
|---|---|
| 0 | off |
| 1 | on |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Do? | r/w | base<br>1dP<br>2dP<br>3dP | 4181<br>12373<br>20565<br>28757 | 41130 | Enum | *Enum_Ausgang* | Forcing of this digital output. Forcing involves the external operation of an output. The instrument has no influence on this output (use of free outputs by superordinate system). |

| | |
|---|---|
| 0 | off |
| 1 | on |

# 15 Out.1

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| F.Out? | r/w | base<br>1dP<br>2dP<br>3dP | 4182<br>12374<br>20566<br>28758 | 41132 | Float | 0...120 | ☐ | Forcing value of the analog output. Forcing involves the external operation of an output, i.e. the instrument has an influence on this output. (Used for the operation of free outputs e.g. by a supervisory PLC.) |

# 16 Out.2

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.Act | r/w | base<br>1dP<br>2dP<br>3dP | 4250<br>12442<br>20634<br>28826 | 41268 | Enum | *Enum_OAct* | Operating sense of the switching output.<br>Direct: Active function (e.g. limit value) switches the output ON;<br>Inverse: Active function (e.g. limit value) switches the output OFF. |

| | |
|---|---|
| 0 | Direct / Normally de-energized mode |
| 1 | inverse / normally closed |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4251<br>12443<br>20635<br>28827 | 41270 | Enum | *Enum_Y1* | Output function: Controller output Y1 |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4252<br>12444<br>20636<br>28828 | 41272 | Enum | *Enum_Y2* | Output function: Controller output Y2. Caution: Do not confuse the controller output Y2 with the parameter 'Fixed output Y2' ! |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y2. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4253<br>12445<br>20637<br>28829 | 41274 | Enum | *Enum_Lim1* | Output function: Signal limit 1 |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4254<br>12446<br>20638<br>28830 | 41276 | Enum | *Enum_Lim2* | Output function: Signal limit 2 |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 2. |

# 16 Out.2

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4255<br>12447<br>20639<br>28831 | 41278 | Enum | *Enum_Lim3* | Output function: Signal limit 3 |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an alarm from limit value 3. |
| LP.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4257<br>12449<br>20641<br>28833 | 41282 | Enum | *Enum_OUT_LPAL* | Output function: Signal Interruption alarm (LOOP)<br>The overall control loop is monitored and the process value has to change with an output signal of maximum value, else loop alarm is generated. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This loop alarm (= open loop alarm) is assigned to this output. |
| HC.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4258<br>12450<br>20642<br>28834 | 41284 | Enum | *Enum_OUT_HCAL* | Output function: Signal Heat current alarm. Either break (= current I < heating current limit) can be monitored or overload (= current I > heating current limit), dependent on configuration. |
| | | | | | | 0 | not active |
| | | | | | | 1 | The heating current alarm is assigned to this output. |
| HC.SC | r/w | base<br>1dP<br>2dP<br>3dP | 4259<br>12451<br>20643<br>28835 | 41286 | Enum | *Enum_HCSC* | Output function: Signal Solid-state relay (SSR) short circuit.<br>The short circuit alarm of the SSR is triggered, if a current is detected in the heating circuit, although the controller output is switched off. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an SSR fault. |
| timE | r/w | base<br>1dP<br>2dP<br>3dP | 4260<br>12452<br>20644<br>28836 | 41288 | Enum | *Enum_time* | Output function: Signal Timer running.<br>This message is generated by the setpoint processing, if a timer mode has been configured, and the time has elapsed. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by the timer status |
| t.End | writ | base<br>1dP<br>2dP<br>3dP | 4276<br>12468<br>20660<br>28852 | 41320 | Enum | *Enum_TEnd* | Output function: Signal Timer end.<br>This message is available when the timer has been completed (only when configured as a timer). |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by the message 'Timer end'. |
| P.End | r/w | base<br>1dP<br>2dP<br>3dP | 4261<br>12453<br>20645<br>28837 | 41290 | Enum | *Enum_PEnd* | Output function: Signal Program end.<br>This message is available when the program has been completed (only when configured as a program controller). |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by the message 'Program end'. |

# 16 Out.2

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.1 | r/w | base | 4262 | 41292 | Enum | *Enum_FAi1* | Output function: Signal INP1 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP1. |
| | | 1dP | 12454 | | | | |
| | | 2dP | 20646 | | | | |
| | | 3dP | 28838 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP1 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.2 | r/w | base | 4263 | 41294 | Enum | *Enum_FAi2* | Output function: Signal INP2 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP2. |
| | | 1dP | 12455 | | | | |
| | | 2dP | 20647 | | | | |
| | | 3dP | 28839 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP2 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.1 | r/w | base | 4265 | 41298 | Enum | *Enum_PrG1* | Output function: Signal programmer's control output no. 1.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | 1dP | 12457 | | | | |
| | | 2dP | 20649 | | | | |
| | | 3dP | 28841 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 1 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.2 | r/w | base | 4266 | 41300 | Enum | *Enum_PrG2* | Output function: Signal programmer's control output no 2.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | 1dP | 12458 | | | | |
| | | 2dP | 20650 | | | | |
| | | 3dP | 28842 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 2 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.3 | r/w | base | 4267 | 41302 | Enum | *Enum_PrG3* | Output function: Signal programmer's control output no. 3. T<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | 1dP | 12459 | | | | |
| | | 2dP | 20651 | | | | |
| | | 3dP | 28843 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 3 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.4 | r/w | base | 4268 | 41304 | Enum | *Enum_PrG4* | Output function: Signal programmer's control output no 4.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | 1dP | 12460 | | | | |
| | | 2dP | 20652 | | | | |
| | | 3dP | 28844 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 4 is assigned to this output. |

# 16 Out.2

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| CALL | r/w | base<br>1dP<br>2dP<br>3dP | 4269<br>12461<br>20653<br>28845 | 41306 | Enum | *Enum_CALL* | Output: Operator call.<br>At the end of a program segment, a contact is set, e.g. for an acoustic signal. This indicates to the operator that a certain program status has been reached, and operator action is required. Operator calling is used, if the program may only be continued after a check or some kind of operator action. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is switched by an operator call. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out? | r | base<br>1dP<br>2dP<br>3dP | 4280<br>12472<br>20664<br>28856 | 41328 | Enum | *Enum_Ausgang* | Status of the digital output |

| | |
|---|---|
| 0 | off |
| 1 | on |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Do? | r/w | base<br>1dP<br>2dP<br>3dP | 4281<br>12473<br>20665<br>28857 | 41330 | Enum | *Enum_Ausgang* | Forcing of this digital output. Forcing involves the external operation of an output. The instrument has no influence on this output (use of free outputs by superordinate system). |

| | |
|---|---|
| 0 | off |
| 1 | on |

# 17 Out.3

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.Act | r/w | base<br>1dP<br>2dP<br>3dP | 4350<br>12542<br>20734<br>28926 | 41468 | Enum | *Enum_OAct* | Operating sense of the switching output.<br>Direct: Active function (e.g. limit value) switches the output ON;<br>Inverse: Active function (e.g. limit value) switches the output OFF. |

| | |
|---|---|
| 0 | Direct / Normally de-energized mode |
| 1 | inverse / normally closed |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.tYP | r/w | base<br>1dP<br>2dP<br>3dP | 4370<br>12562<br>20754<br>28946 | 41508 | Enum | *Enum_OtYP* | Signal type selection OUT |

| | |
|---|---|
| 0 | relay/logic |
| 1 | 0 ... 20 mA continuous |
| 2 | 4 ... 20 mA continuous |
| 3 | 0...10 V continuous |
| 4 | 2...10 V continuous |
| 5 | Transmitter supply |

# 17 Out.3

## ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| Y.1 | r/w | base 4351<br>1dP 12543<br>2dP 20735<br>3dP 28927 | 41470 | Enum | *Enum_Y1* | Output function: Controller output Y1 |
| | | | | | 0 | not active |
| | | | | | 1 | This output provides the controller output Y1. |
| Y.2 | r/w | base 4352<br>1dP 12544<br>2dP 20736<br>3dP 28928 | 41472 | Enum | *Enum_Y2* | Output function: Controller output Y2. Caution: Do not confuse the controller output Y2 with the parameter 'Fixed output Y2' ! |
| | | | | | 0 | not active |
| | | | | | 1 | This output provides the controller output Y2. |
| Lim.1 | r/w | base 4353<br>1dP 12545<br>2dP 20737<br>3dP 28929 | 41474 | Enum | *Enum_Lim1* | Output function: Signal limit 1 |
| | | | | | 0 | not active |
| | | | | | 1 | This output is activated by an alarm from limit value 1. |
| Lim.2 | r/w | base 4354<br>1dP 12546<br>2dP 20738<br>3dP 28930 | 41476 | Enum | *Enum_Lim2* | Output function: Signal limit 2 |
| | | | | | 0 | not active |
| | | | | | 1 | This output is activated by an alarm from limit value 2. |
| Lim.3 | r/w | base 4355<br>1dP 12547<br>2dP 20739<br>3dP 28931 | 41478 | Enum | *Enum_Lim3* | Output function: Signal limit 3 |
| | | | | | 0 | not active |
| | | | | | 1 | This output is activated by an alarm from limit value 3. |
| LP.AL | r/w | base 4357<br>1dP 12549<br>2dP 20741<br>3dP 28933 | 41482 | Enum | *Enum_OUT_LPAL* | Output function: Signal Interruption alarm (LOOP) The overall control loop is monitored and the process value has to change with an output signal of maximum value, else loop alarm is generated. |
| | | | | | 0 | not active |
| | | | | | 1 | This loop alarm (= open loop alarm) is assigned to this output. |
| HC.AL | r/w | base 4358<br>1dP 12550<br>2dP 20742<br>3dP 28934 | 41484 | Enum | *Enum_OUT_HCAL* | Output function: Signal Heat current alarm. Either break (= current I < heating current limit) can be monitored or overload (= current I > heating current limit), dependent on configuration. |
| | | | | | 0 | not active |
| | | | | | 1 | The heating current alarm is assigned to this output. |

# 17 Out.3

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HC.SC | r/w | base<br>1dP<br>2dP<br>3dP | 4359<br>12551<br>20743<br>28935 | 41486 | Enum | *Enum_HCSC* | Output function: Signal Solid-state relay (SSR) short circuit.<br>The short circuit alarm of the SSR is triggered, if a current is detected in the heating circuit, although the controller output is switched off. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an SSR fault. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| timE | r/w | base<br>1dP<br>2dP<br>3dP | 4360<br>12552<br>20744<br>28936 | 41488 | Enum | *Enum_time* | Output function: Signal Timer running.<br>This message is generated by the setpoint processing, if a timer mode has been configured, and the time has elapsed. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the timer status |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| P.End | r/w | base<br>1dP<br>2dP<br>3dP | 4361<br>12553<br>20745<br>28937 | 41490 | Enum | *Enum_PEnd* | Output function: Signal Program end.<br>This message is available when the program has been completed (only when configured as a program controller). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Program end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4362<br>12554<br>20746<br>28938 | 41492 | Enum | *Enum_FAi1* | Output function: Signal INP1 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP1. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP1 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| t.End | r/w | base<br>1dP<br>2dP<br>3dP | 4376<br>12568<br>20760<br>28952 | 41520 | Enum | *Enum_TEnd* | Output function: Signal Timer end.<br>This message is available when the timer has been completed (only when configured as a timer). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Timer end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4363<br>12555<br>20747<br>28939 | 41494 | Enum | *Enum_FAi2* | Output function: Signal INP2 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP2. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP2 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4365<br>12557<br>20749<br>28941 | 41498 | Enum | *Enum_PrG1* | Output function: Signal programmer's control output no. 1.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 1 is assigned to this output. |

# 17 Out.3

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4366<br>12558<br>20750<br>28942 | 41500 | Enum | *Enum_PrG2* | Output function: Signal programmer's control output no 2.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | | | | | 0 | not active |
| | | | | | | 1 | Control output 2 is assigned to this output. |
| PrG.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4367<br>12559<br>20751<br>28943 | 41502 | Enum | *Enum_PrG3* | Output function: Signal programmer's control output no. 3. T<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | | | | | 0 | not active |
| | | | | | | 1 | Control output 3 is assigned to this output. |
| PrG.4 | r/w | base<br>1dP<br>2dP<br>3dP | 4368<br>12560<br>20752<br>28944 | 41504 | Enum | *Enum_PrG4* | Output function: Signal programmer's control output no 4.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |
| | | | | | | 0 | not active |
| | | | | | | 1 | Control output 4 is assigned to this output. |
| CALL | r/w | base<br>1dP<br>2dP<br>3dP | 4369<br>12561<br>20753<br>28945 | 41506 | Enum | *Enum_CALL* | Output: Operator call.<br>At the end of a program segment, a contact is set, e.g. for an acoustic signal. This indicates to the operator that a certain program status has been reached, and operator action is required. Operator calling is used, if the program may only be continued after a check or some kind of operator action. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is switched by an operator call. |
| Out.0 | r/w | base<br>1dP<br>2dP<br>3dP | 4371<br>12563<br>20755<br>28947 | 41510 | Float | -1999...9999 ☐ | Lower scaling limit of the analog output (corresponds to 0%). If current and voltage signals are used as output values, the display can be scaled to the output value in the Parameter Level. The output value of the lower scaling point is indicated in the respective electrical unit (mA / V). |
| Out.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4372<br>12564<br>20756<br>28948 | 41512 | Float | -1999...9999 ☐ | Upper scaling limit of the analog output (corresponds to 100%). If current or voltage signals are used as output values, scaling of the display can be applied to the output value by means of the Parameter Level. Definition of the upper output limit is done using the corresponding electrical value (mA / V). |

# 17 Out.3

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.Src | r/w | base | 4373 | 41514 | Enum | *Enum_OSrc* | Signal source of the analog output. |
| | | 1dP | 12565 | | | | |
| | | 2dP | 20757 | | | | |
| | | 3dP | 28949 | | | | |

| | |
|---|---|
| 0 | not active |
| 1 | controller output y1 (cont.) |
| 2 | controller output y2 (cont.) |
| 3 | process value |
| 4 | effective setpoint Weff |
| 5 | Control deviation xw (process value - setpoint)= relative alarm<br>Note: Monitoring with the effective setpoint Weff. For example using a ramp it is the changing setpoint, not the target setpoint of the ramp. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out? | r | base | 4380 | 41528 | Enum | *Enum_Ausgang* | Status of the digital output |
| | | 1dP | 12572 | | | | |
| | | 2dP | 20764 | | | | |
| | | 3dP | 28956 | | | | |

| | |
|---|---|
| 0 | off |
| 1 | on |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Do? | r/w | base | 4381 | 41530 | Enum | *Enum_Ausgang* | Forcing of this digital output. Forcing involves the external operation of an output. The instrument has no influence on this output (use of free outputs by superordinate system). |
| | | 1dP | 12573 | | | | |
| | | 2dP | 20765 | | | | |
| | | 3dP | 28957 | | | | |

| | |
|---|---|
| 0 | off |
| 1 | on |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Out? | r/w | base | 4382 | 41532 | Float | 0...120 ☐ | Forcing value of the analog output. Forcing involves the external operation of an output, i.e. the instrument has an influence on this output. (Used for the operation of free outputs e.g. by a supervisory PLC.) |
| | | 1dP | 12574 | | | | |
| | | 2dP | 20766 | | | | |
| | | 3dP | 28958 | | | | |

# 18 Out.4

## • ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.Act | r/w | base<br>1dP<br>2dP<br>3dP | 4450<br>12642<br>20834<br>29026 | 41668 | Enum | *Enum_OAct* | Operating sense of the switching output.<br>Direct: Active function (e.g. limit value) switches the output ON;<br>Inverse: Active function (e.g. limit value) switches the output OFF. |

| | |
|---|---|
| 0 | Direct / Normally de-energized mode |
| 1 | inverse / normally closed |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4451<br>12643<br>20835<br>29027 | 41670 | Enum | *Enum_Y1* | Output function: Controller output Y1 |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4452<br>12644<br>20836<br>29028 | 41672 | Enum | *Enum_Y2* | Output function: Controller output Y2. Caution: Do not confuse the controller output Y2 with the parameter 'Fixed output Y2' ! |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y2. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4453<br>12645<br>20837<br>29029 | 41674 | Enum | *Enum_Lim1* | Output function: Signal limit 1 |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4454<br>12646<br>20838<br>29030 | 41676 | Enum | *Enum_Lim2* | Output function: Signal limit 2 |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 2. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4455<br>12647<br>20839<br>29031 | 41678 | Enum | *Enum_Lim3* | Output function: Signal limit 3 |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an alarm from limit value 3. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| LP.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4457<br>12649<br>20841<br>29033 | 41682 | Enum | *Enum_OUT_LPAL* | Output function: Signal Interruption alarm (LOOP)<br>The overall control loop is monitored and the process value has to change with an output signal of maximum value, else loop alarm is generated. |

| | |
|---|---|
| 0 | not active |
| 1 | This loop alarm (= open loop alarm) is assigned to this output. |

# 18 Out.4

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HC.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4458<br>2650<br>0842<br>9034 | 1684 | Enum | *Enum_OUT_HCAL* | Output function: Signal Heat current alarm. Either break (= current I < heating current limit) can be monitored or overload (= current I > heating current limit), dependent on configuration. |

| | |
|---|---|
| 0 | not active |
| 1 | The heating current alarm is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| HC.SC | r/w | base<br>1dP<br>2dP<br>3dP | 4459<br>2651<br>0843<br>9035 | 1686 | Enum | *Enum_HCSC* | Output function: Signal Solid-state relay (SSR) short circuit.<br>The short circuit alarm of the SSR is triggered, if a current is detected in the heating circuit, although the controller output is switched off. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by an SSR fault. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| timE | r/w | base<br>1dP<br>2dP<br>3dP | 4460<br>2652<br>0844<br>9036 | 1688 | Enum | *Enum_time* | Output function: Signal Timer running.<br>This message is generated by the setpoint processing, if a timer mode has been configured, and the time has elapsed. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the timer status |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| P.End | r/w | base<br>1dP<br>2dP<br>3dP | 4461<br>2653<br>0845<br>9037 | 1690 | Enum | *Enum_PEnd* | Output function: Signal Program end.<br>This message is available when the program has been completed (only when configured as a program controller). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Program end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| t.End | r/w | base<br>1dP<br>2dP<br>3dP | 4476<br>2668<br>0860<br>9052 | 1720 | Enum | *Enum_TEnd* | Output function: Signal Timer end.<br>This message is available when the timer has been completed (only when configured as a timer). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Timer end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4462<br>2654<br>0846<br>9038 | 1692 | Enum | *Enum_FAi1* | Output function: Signal INP1 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP1. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP1 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4463<br>2655<br>0847<br>9039 | 1694 | Enum | *Enum_FAi2* | Output function: Signal INP2 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP2. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP2 fault'. |

# 18 Out.4

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4465<br>12657<br>20849<br>29041 | 41698 | Enum | *Enum_PrG1* | Output function: Signal programmer's control output no. 1.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 1 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4466<br>12658<br>20850<br>29042 | 41700 | Enum | *Enum_PrG2* | Output function: Signal programmer's control output no 2.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 2 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4467<br>12659<br>20851<br>29043 | 41702 | Enum | *Enum_PrG3* | Output function: Signal programmer's control output no. 3. T<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 3 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.4 | r/w | base<br>1dP<br>2dP<br>3dP | 4468<br>12660<br>20852<br>29044 | 41704 | Enum | *Enum_PrG4* | Output function: Signal programmer's control output no 4.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 4 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| CALL | r/w | base<br>1dP<br>2dP<br>3dP | 4469<br>12661<br>20853<br>29045 | 41706 | Enum | *Enum_CALL* | Output: Operator call.<br>At the end of a program segment, a contact is set, e.g. for an acoustic signal. This indicates to the operator that a certain program status has been reached, and operator action is required. Operator calling is used, if the program may only be continued after a check or some kind of operator action. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is switched by an operator call. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out? | r | base<br>1dP<br>2dP<br>3dP | 4480<br>12672<br>20864<br>29056 | 41728 | Enum | *Enum_Ausgang* | Status of the digital output |

| | |
|---|---|
| 0 | off |
| 1 | on |

# 18 Out.4

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Do? | r/w | base<br>1dP<br>2dP<br>3dP | 4481<br>12673<br>20865<br>29057 | 41730 | Enum | *Enum_Ausgang* | Forcing of this digital output. Forcing involves the external operation of an output. The instrument has no influence on this output (use of free outputs by superordinate system). |

| | |
|---|---|
| 0 | off |
| 1 | on |

# 19 Out.5

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.Act | r/w | base<br>1dP<br>2dP<br>3dP | 4550<br>12742<br>20934<br>29126 | 41868 | Enum | *Enum_OAct* | Operating sense of the switching output.<br>Direct: Active function (e.g. limit value) switches the output ON;<br>Inverse: Active function (e.g. limit value) switches the output OFF. |

| | |
|---|---|
| 0 | Direct / Normally de-energized mode |
| 1 | inverse / normally closed |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| O.tYP | r/w | base<br>1dP<br>2dP<br>3dP | 4570<br>12762<br>20954<br>29146 | 41908 | Enum | *Enum_OtYP* | Signal type selection OUT |

| | |
|---|---|
| 0 | relay/logic |
| 1 | 0 ... 20 mA continuous |
| 2 | 4 ... 20 mA continuous |
| 3 | 0...10 V continuous |
| 4 | 2...10 V continuous |
| 5 | Transmitter supply |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4551<br>12743<br>20935<br>29127 | 41870 | Enum | *Enum_Y1* | Output function: Controller output Y1 |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y1. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Y.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4552<br>12744<br>20936<br>29128 | 41872 | Enum | *Enum_Y2* | Output function: Controller output Y2. Caution: Do not confuse the controller output Y2 with the parameter 'Fixed output Y2' ! |

| | |
|---|---|
| 0 | not active |
| 1 | This output provides the controller output Y2. |

# 19 Out.5

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4553<br>12745<br>20937<br>29129 | 41874 | Enum | *Enum_Lim1* | Output function: Signal limit 1 |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an alarm from limit value 1. |
| Lim.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4554<br>12746<br>20938<br>29130 | 41876 | Enum | *Enum_Lim2* | Output function: Signal limit 2 |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an alarm from limit value 2. |
| Lim.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4555<br>12747<br>20939<br>29131 | 41878 | Enum | *Enum_Lim3* | Output function: Signal limit 3 |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an alarm from limit value 3. |
| LP.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4557<br>12749<br>20941<br>29133 | 41882 | Enum | *Enum_OUT_LPAL* | Output function: Signal Interruption alarm (LOOP) The overall control loop is monitored and the process value has to change with an output signal of maximum value, else loop alarm is generated. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This loop alarm (= open loop alarm) is assigned to this output. |
| HC.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4558<br>12750<br>20942<br>29134 | 41884 | Enum | *Enum_OUT_HCAL* | Output function: Signal Heat current alarm. Either break (= current I < heating current limit) can be monitored or overload (= current I > heating current limit), dependent on configuration. |
| | | | | | | 0 | not active |
| | | | | | | 1 | The heating current alarm is assigned to this output. |
| HC.SC | r/w | base<br>1dP<br>2dP<br>3dP | 4559<br>12751<br>20943<br>29135 | 41886 | Enum | *Enum_HCSC* | Output function: Signal Solid-state relay (SSR) short circuit.<br>The short circuit alarm of the SSR is triggered, if a current is detected in the heating circuit, although the controller output is switched off. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an SSR fault. |
| timE | r/w | base<br>1dP<br>2dP<br>3dP | 4560<br>12752<br>20944<br>29136 | 41888 | Enum | *Enum_time* | Output function: Signal Timer running.<br>This message is generated by the setpoint processing, if a timer mode has been configured, and the time has elapsed. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by the timer status |

# 19 Out.5

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| P.End | r/w | base<br>1dP<br>2dP<br>3dP | 4561<br>12753<br>20945<br>29137 | 41890 | Enum | *Enum_PEnd* | Output function: Signal Program end.<br>This message is available when the program has been completed (only when configured as a program controller). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Program end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4562<br>12754<br>20946<br>29138 | 41892 | Enum | *Enum_FAi1* | Output function: Signal INP1 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP1. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP1 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| t.End | r/w | base<br>1dP<br>2dP<br>3dP | 4576<br>12768<br>20960<br>29152 | 41920 | Enum | *Enum_TEnd* | Output function: Signal Timer end.<br>This message is available when the timer has been completed (only when configured as a timer). |

| | |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Timer end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4563<br>12755<br>20947<br>29139 | 41894 | Enum | *Enum_FAi2* | Output function: Signal INP2 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP2. |

| | |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP2 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4565<br>12757<br>20949<br>29141 | 41898 | Enum | *Enum_PrG1* | Output function: Signal programmer's control output no. 1.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 1 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4566<br>12758<br>20950<br>29142 | 41900 | Enum | *Enum_PrG2* | Output function: Signal programmer's control output no 2.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 2 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4567<br>12759<br>20951<br>29143 | 41902 | Enum | *Enum_PrG3* | Output function: Signal programmer's control output no. 3. T<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 3 is assigned to this output. |

# 19 Out.5

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.4 | r/w | base<br>1dP<br>2dP<br>3dP | 4568<br>12760<br>20952<br>29144 | 41904 | Enum | *Enum_PrG4* | Output function: Signal programmer's control output no 4.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| | |
|---|---|
| 0 | not active |
| 1 | Control output 4 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| CALL | r/w | base<br>1dP<br>2dP<br>3dP | 4569<br>12761<br>20953<br>29145 | 41906 | Enum | *Enum_CALL* | Output: Operator call.<br>At the end of a program segment, a contact is set, e.g. for an acoustic signal. This indicates to the operator that a certain program status has been reached, and operator action is required. Operator calling is used, if the program may only be continued after a check or some kind of operator action. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is switched by an operator call. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out.0 | r/w | base<br>1dP<br>2dP<br>3dP | 4571<br>12763<br>20955<br>29147 | 41910 | Float | -1999...9999 ☐ | Lower scaling limit of the analog output (corresponds to 0%). If current and voltage signals are used as output values, the display can be scaled to the output value in the Parameter Level. The output value of the lower scaling point is indicated in the respective electrical unit (mA / V). |
| Out.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4572<br>12764<br>20956<br>29148 | 41912 | Float | -1999...9999 ☐ | Upper scaling limit of the analog output (corresponds to 100%). If current or voltage signals are used as output values, scaling of the display can be applied to the output value by means of the Parameter Level. Definition of the upper output limit is done using the corresponding electrical value (mA / V). |
| O.Src | r/w | base<br>1dP<br>2dP<br>3dP | 4573<br>12765<br>20957<br>29149 | 41914 | Enum | *Enum_OSrc* | Signal source of the analog output. |

| | |
|---|---|
| 0 | not active |
| 1 | controller output y1 (cont.) |
| 2 | controller output y2 (cont.) |
| 3 | process value |
| 4 | effective setpoint Weff |
| 5 | Control deviation xw (process value - setpoint)= relative alarm<br>Note: Monitoring with the effective setpoint Weff. For example using a ramp it is the changing setpoint, not the target setpoint of the ramp. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out? | r | base<br>1dP<br>2dP<br>3dP | 4580<br>12772<br>20964<br>29156 | 41928 | Enum | *Enum_Ausgang* | Status of the digital output |

| | |
|---|---|
| 0 | off |
| 1 | on |

## 19 Out.5

### Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| F.Do? | r/w | base 4581<br>1dP 12773<br>2dP 20965<br>3dP 29157 | 41930 | Enum | *Enum_Ausgang* | Forcing of this digital output. Forcing involves the external operation of an output. The instrument has no influence on this output (use of free outputs by superordinate system). |
| | | | | | 0 | off |
| | | | | | 1 | on |
| F.Out? | r/w | base 4582<br>1dP 12774<br>2dP 20966<br>3dP 29158 | 41932 | Float | 0...120 ☐ | Forcing value of the analog output. Forcing involves the external operation of an output, i.e. the instrument has an influence on this output. (Used for the operation of free outputs e.g. by a supervisory PLC.) |

## 20 Out.6

### ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| O.Act | r/w | base 4650<br>1dP 12842<br>2dP 21034<br>3dP 29226 | 42068 | Enum | *Enum_OAct* | Operating sense of the switching output.<br>Direct: Active function (e.g. limit value) switches the output ON;<br>Inverse: Active function (e.g. limit value) switches the output OFF. |
| | | | | | 0 | Direct / Normally de-energized mode |
| | | | | | 1 | inverse / normally closed |
| Y.1 | r/w | base 4651<br>1dP 12843<br>2dP 21035<br>3dP 29227 | 42070 | Enum | *Enum_Y1* | Output function: Controller output Y1 |
| | | | | | 0 | not active |
| | | | | | 1 | This output provides the controller output Y1. |
| Y.2 | r/w | base 4652<br>1dP 12844<br>2dP 21036<br>3dP 29228 | 42072 | Enum | *Enum_Y2* | Output function: Controller output Y2. Caution: Do not confuse the controller output Y2 with the parameter 'Fixed output Y2' ! |
| | | | | | 0 | not active |
| | | | | | 1 | This output provides the controller output Y2. |
| Lim.1 | r/w | base 4653<br>1dP 12845<br>2dP 21037<br>3dP 29229 | 42074 | Enum | *Enum_Lim1* | Output function: Signal limit 1 |
| | | | | | 0 | not active |
| | | | | | 1 | This output is activated by an alarm from limit value 1. |

# 20 Out.6

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Lim.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4654<br>12846<br>21038<br>29230 | 42076 | Enum | *Enum_Lim2* | Output function: Signal limit 2 |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an alarm from limit value 2. |
| Lim.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4655<br>12847<br>21039<br>29231 | 42078 | Enum | *Enum_Lim3* | Output function: Signal limit 3 |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an alarm from limit value 3. |
| LP.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4657<br>12849<br>21041<br>29233 | 42082 | Enum | *Enum_OUT_LPAL* | Output function: Signal Interruption alarm (LOOP)<br>The overall control loop is monitored and the process value has to change with an output signal of maximum value, else loop alarm is generated. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This loop alarm (= open loop alarm) is assigned to this output. |
| HC.AL | r/w | base<br>1dP<br>2dP<br>3dP | 4658<br>12850<br>21042<br>29234 | 42084 | Enum | *Enum_OUT_HCAL* | Output function: Signal Heat current alarm. Either break (= current I < heating current limit) can be monitored or overload (= current I > heating current limit), dependent on configuration. |
| | | | | | | 0 | not active |
| | | | | | | 1 | The heating current alarm is assigned to this output. |
| HC.SC | r/w | base<br>1dP<br>2dP<br>3dP | 4659<br>12851<br>21043<br>29235 | 42086 | Enum | *Enum_HCSC* | Output function: Signal Solid-state relay (SSR) short circuit.<br>The short circuit alarm of the SSR is triggered, if a current is detected in the heating circuit, although the controller output is switched off. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by an SSR fault. |
| timE | r/w | base<br>1dP<br>2dP<br>3dP | 4660<br>12852<br>21044<br>29236 | 42088 | Enum | *Enum_time* | Output function: Signal Timer running.<br>This message is generated by the setpoint processing, if a timer mode has been configured, and the time has elapsed. |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by the timer status |
| P.End | r/w | base<br>1dP<br>2dP<br>3dP | 4661<br>12853<br>21045<br>29237 | 42090 | Enum | *Enum_PEnd* | Output function: Signal Program end.<br>This message is available when the program has been completed (only when configured as a program controller). |
| | | | | | | 0 | not active |
| | | | | | | 1 | This output is activated by the message 'Program end'. |

# 20 Out.6

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| t.End | r/w | base<br>1dP<br>2dP<br>3dP | 4676<br>12868<br>21060<br>29252 | 42120 | Enum | *Enum_TEnd* | Output function: Signal Timer end.<br>This message is available when the timer has been completed (only when configured as a timer). |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | This output is activated by the message 'Timer end'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4662<br>12854<br>21046<br>29238 | 42092 | Enum | *Enum_FAi1* | Output function: Signal INP1 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP1. |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP1 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| FAi.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4663<br>12855<br>21047<br>29239 | 42094 | Enum | *Enum_FAi2* | Output function: Signal INP2 fault.<br>The fail signal is generated, if a fault occurs at the analog Input INP2. |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | This output sends the error message 'INP2 fault'. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.1 | r/w | base<br>1dP<br>2dP<br>3dP | 4665<br>12857<br>21049<br>29241 | 42098 | Enum | *Enum_PrG1* | Output function: Signal programmer's control output no. 1.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | Control output 1 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.2 | r/w | base<br>1dP<br>2dP<br>3dP | 4666<br>12858<br>21050<br>29242 | 42100 | Enum | *Enum_PrG2* | Output function: Signal programmer's control output no 2.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | Control output 2 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.3 | r/w | base<br>1dP<br>2dP<br>3dP | 4667<br>12859<br>21051<br>29243 | 42102 | Enum | *Enum_PrG3* | Output function: Signal programmer's control output no. 3. T<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | Control output 3 is assigned to this output. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| PrG.4 | r/w | base<br>1dP<br>2dP<br>3dP | 4668<br>12860<br>21052<br>29244 | 42104 | Enum | *Enum_PrG4* | Output function: Signal programmer's control output no 4.<br>A control output is one of the four digital signals that can be operated segment-wise by a program. |

| Value | Description |
|---|---|
| 0 | not active |
| 1 | Control output 4 is assigned to this output. |

# 20 Out.6

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| CALL | r/w | base<br>1dP<br>2dP<br>3dP | 4669<br>12861<br>21053<br>29245 | 42106 | Enum | *Enum_CALL* | Output: Operator call.<br>At the end of a program segment, a contact is set, e.g. for an acoustic signal. This indicates to the operator that a certain program status has been reached, and operator action is required. Operator calling is used, if the program may only be continued after a check or some kind of operator action. |

| | |
|---|---|
| 0 | not active |
| 1 | This output is switched by an operator call. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Out? | r | base<br>1dP<br>2dP<br>3dP | 4680<br>12872<br>21064<br>29256 | 42128 | Enum | *Enum_Ausgang* | Status of the digital output |

| | |
|---|---|
| 0 | off |
| 1 | on |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| F.Do? | r/w | base<br>1dP<br>2dP<br>3dP | 4681<br>12873<br>21065<br>29257 | 42130 | Enum | *Enum_Ausgang* | Forcing of this digital output. Forcing involves the external operation of an output. The instrument has no influence on this output (use of free outputs by superordinate system). |

| | |
|---|---|
| 0 | off |
| 1 | on |

# 21 PAr.2

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Pb12 | r/w | base<br>1dP<br>2dP<br>3dP | 5030<br>13222<br>21414<br>29606 | 42828 | Float | 0,1...9999 | ☐ | Proportional band 1 (heating) in engineering unit (e.g. °C) of the 2nd parameter set. The Pb defines the ratio between output value and control deviation. The smaller the value of Pb is, the stronger is the control response for a specific control deviation. Too large and too small values for Pb lead to process oscillations (hunting). |
| Pb22 | r/w | base<br>1dP<br>2dP<br>3dP | 5031<br>13223<br>21415<br>29607 | 42830 | Float | 0,1...9999 | ☐ | Proportional band 2 (cooling) in engineering unit (e.g. °C) of the 2nd parameter set. The Pb defines the ratio between output value and control deviation. The smaller the value of Pb is, the stronger is the control response for a specific control deviation. Too large and too small values for Pb lead to process oscillations (hunting). |

## 21 PAr.2

### • PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| ti22 | r/w | base 5033<br>1dP 13225<br>2dP 21417<br>3dP 29609 | 42834 | Float | 0...9999 | ☑ | Integral action time 2 (cooling) [s]. 2nd parameter set. T is the time constant of the integral portion. The smaller Ti is, the faster is the response of the integral action. Ti too small: Control tends to oscillate.<br>Ti too large: Control is sluggish and needs a long time to line out. |
| ti12 | r/w | base 5032<br>1dP 13224<br>2dP 21416<br>3dP 29608 | 42832 | Float | 0...9999 | ☑ | Integral action time 1 (heating) [s]. 2nd parameter set. T is the time constant of the integral portion. The smaller Ti is, the faster is the response of the integral action. Ti too small: Control tends to oscillate.<br>Ti too large: Control is sluggish and needs a long time to line out. |
| td12 | r/w | base 5034<br>1dP 13226<br>2dP 21418<br>3dP 29610 | 42836 | Float | 0...9999 | ☑ | Derivative action time 1 (heating) [s], 2nd parameter set Td is the time constant of the derivative portion. The faster the process value changes, and the larger the value of Td is, the stronger will be the derivative action. Td too small: Very little derivative action.<br>Td too large: Control tends to oscillate. |
| td22 | r/w | base 5035<br>1dP 13227<br>2dP 21419<br>3dP 29611 | 42838 | Float | 0...9999 | ☑ | Derivative action time 2 (cooling) [s], 2nd parameter set. Td is the time constant of the derivative portion. The faster the process value changes, and the larger the value of Td is, the stronger will be the derivative action.<br>Td too small: Very little derivative action.<br>Td too large: Control tends to oscillate. |

## 22 ProG

### • ConF

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| t.bAS | r/w | base 6030<br>1dP 14222<br>2dP 22414<br>3dP 30606 | 44828 | Enum | *Enum_tbAS* | Definition of the programmer's time base in hours using minutes, or in minutes using seconds. |

| | |
|---|---|
| 0 | Hours [hh] : Minutes [mm] |
| 1 | Minutes [mm] : Seconds [ss] |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| Pr.no | r/w | base<br>1dP<br>2dP<br>3dP | 6000<br>14192<br>22384<br>30576 | 44768 | Enum | *Enum_PrgNoPar* | Program number (nominal). The program number (nominal) determines which program is to be started next. Running programs are not affected. The selected program is only started after a reset or restart. |

| | |
|---|---|
| 1 | prog. 01 |
| 2 | prog. 02 |
| 3 | prog. 03 |
| 4 | prog. 04 |
| 5 | prog. 05 |
| 6 | prog. 06 |
| 7 | prog. 07 |
| 8 | prog. 08 |
| 9 | prog. 09 |
| 10 | prog. 10 |
| 11 | prog. 11 |
| 12 | prog. 12 |
| 13 | prog. 13 |
| 14 | prog. 14 |
| 15 | prog. 15 |
| 16 | prog. 16 |

Prog Parameters below with base addresses 6100 to 6166 apply to the currently selected program.
To access these parameters for specific unselected programs, offset the addresses by +100 for each successive program.
*For example, parameter 6100 for the current selected program would be 6200 for program 1, 6300 for program 2 and 7700 for program 16.*

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| b.Lo | r/w | base<br>1dP<br>2dP<br>3dP | 6100<br>14292<br>22484<br>30676 | 44968 | Float | 0...9999 | ☐ | Lower bandwidth limit. The bandwidth monitor is valid for all segments of an individual program. If the bandwidth is exceeded, the programmer is stopped. The program continues, if the process value returns within the defined monitoring limits. |
| b.Hi | r/w | base<br>1dP<br>2dP<br>3dP | 6101<br>14293<br>22485<br>30677 | 44970 | Float | 0...9999 | ☐ | Upper bandwidth limit. The bandwidth monitor is valid for all segments of an individual program. If the bandwidth is exceeded, the programmer is stopped. The program continues, if the process value returns within the defined monitoring limits. |
| d.00 | r/w | base<br>1dP<br>2dP<br>3dP | 6134<br>14326<br>22518<br>30710 | 45036 | Enum | *ENUM_Spuren* | | Reset value for control outputs 1...4. A program can control up to four digital signals: the control outputs 1...4 The reset value of the control output contains the combination of these signals, which are output together with the controller's internal setpoint, if the programmer is not active. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProoG

## PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| tYPE | r/w | base 6135<br>1dP 14327<br>2dP 22519<br>3dP 30711 | 45038 | Enum | *Enum_SegTyp* | Type of segment 1. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time).Note: The 1st segment cannot be configured as the end segment. |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| SP | r/w | base 6102<br>1dP 14294<br>2dP 22486<br>3dP 30678 | 44972 | Float | -1999...9999 ☐ | End setpoint of segment 1. This is the target setpoint that is reached at the end of the first segment. The target setpoint is approached from the previous valid setpoint (when starting the 1st segment, matching to process value!). When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base 6103<br>1dP 14295<br>2dP 22487<br>3dP 30679 | 44974 | Float | 0...9999 ☐ | Segment time/gradient 1. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base | 6136 | 45040 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 1. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |
| | | 1dP | 14328 | | | | |
| | | 2dP | 22520 | | | | |
| | | 3dP | 30712 | | | | |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProoG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6137<br>14329<br>22521<br>30713 | 45042 | Enum | *Enum_SegTyp* | Segment type of segment 2. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6104<br>14296<br>22488<br>30680 | 44976 | Float | -1999...9999 ☐ | End setpoint of segment 2. This is the target setpoint that is reached at the end of the second segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6105<br>14297<br>22489<br>30681 | 44978 | Float | 0...9999 ☐ | Segment time/gradient 2. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6138<br>14330<br>22522<br>30714 | 45044 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 2. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6139<br>14331<br>22523<br>30715 | 45046 | Enum | *Enum_SegTyp* | Segment type of segment 3. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6106<br>14298<br>22490<br>30682 | 44980 | Float | -1999...9999 ☐ | End setpoint of segment 3. This is the target setpoint that is reached at the end of the third segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6107<br>14299<br>22491<br>30683 | 44982 | Float | 0...9999 ☐ | Segment time/gradient 3. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base | 6140 | 45048 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 3. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |
| | | 1dP | 14332 | | | | |
| | | 2dP | 22524 | | | | |
| | | 3dP | 30716 | | | | |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6141<br>14333<br>22525<br>30717 | 45050 | Enum | *Enum_SegTyp* | Segment type of segment 4. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6108<br>14300<br>22492<br>30684 | 44984 | Float | -1999...9999 ☐ | End setpoint of segment 4. This is the target setpoint that is reached at the end of the fourth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6109<br>14301<br>22493<br>30685 | 44986 | Float | 0...9999 ☐ | Segment time/gradient 4. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6142<br>14334<br>22526<br>30718 | 45052 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 4. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

## 22 ProG

### PArA

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| tYPE | r/w | base 6143<br>1dP 14335<br>2dP 22527<br>3dP 30719 | 45054 | Enum | *Enum_SegTyp* | Segment type of segment 5. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|
| SP | r/w | base 6110<br>1dP 14302<br>2dP 22494<br>3dP 30686 | 44988 | Float | -1999...9999 ☐ | End setpoint of segment 5. This is the target setpoint that is reached at the end of the fifth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base 6111<br>1dP 14303<br>2dP 22495<br>3dP 30687 | 44990 | Float | 0...9999 ☐ | Segment time/gradient 5. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProoG

## PArA

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6144<br>14336<br>22528<br>30720 | 45056 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 5. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6145<br>14337<br>22529<br>30721 | 45058 | Enum | *Enum_SegTyp* | Segment type of segment 6. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6112<br>14304<br>22496<br>30688 | 44992 | Float | -1999...9999 ☐ | End setpoint of segment 6. This is the target setpoint that is reached at the end of the sixth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6113<br>14305<br>22497<br>30689 | 44994 | Float | 0...9999 ☐ | Segment time/gradient 6. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6146<br>14338<br>22530<br>30722 | 45060 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 6. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6147<br>14339<br>22531<br>30723 | 45062 | Enum | *Enum_SegTyp* | | Segment type of segment 7. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6114<br>14306<br>22498<br>30690 | 44996 | Float | -1999...9999 | ☐ | End setpoint of segment 7. This is the target setpoint that is reached at the end of the seventh segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6115<br>14307<br>22499<br>30691 | 44998 | Float | 0...9999 | ☐ | Segment time/gradient 7. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6148<br>14340<br>22532<br>30724 | 45064 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 7. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6149<br>14341<br>22533<br>30725 | 45066 | Enum | *Enum_SegTyp* | Segment type of segment 8. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6116<br>14308<br>22500<br>30692 | 45000 | Float | -1999...9999 ☐ | End setpoint of segment 8. This is the target setpoint that is reached at the end of the eighth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6117<br>14309<br>22501<br>30693 | 45002 | Float | 0...9999 ☐ | Segment time/gradient 8. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## • PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6150<br>14342<br>22534<br>30726 | 45068 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 8. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6151<br>14343<br>22535<br>30727 | 45070 | Enum | *Enum_SegTyp* | Segment type of segment 9. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6118<br>14310<br>22502<br>30694 | 45004 | Float | -1999...9999 ☐ | End setpoint of segment 9. This is the target setpoint that is reached at the end of the ninth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6119<br>14311<br>22503<br>30695 | 45006 | Float | 0...9999 ☐ | Segment time/gradient 9. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6152<br>14344<br>22536<br>30728 | 45072 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 9. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6153<br>14345<br>22537<br>30729 | 45074 | Enum | *Enum_SegTyp* | Segment type of segment 10. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6120<br>14312<br>22504<br>30696 | 45008 | Float | -1999...9999 ☐ | End setpoint of segment 10. This is the target setpoint that is reached at the end of the tenth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6121<br>14313<br>22505<br>30697 | 45010 | Float | 0...9999 ☐ | Segment time/gradient 10. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6154<br>14346<br>22538<br>30730 | 45076 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 10. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

## 22 ProG

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6155<br>14347<br>22539<br>30731 | 45078 | Enum | *Enum_SegTyp* | Segment type of segment 11. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6122<br>14314<br>22506<br>30698 | 45012 | Float | -1999...9999 ☐ | End setpoint of segment 11. This is the target setpoint that is reached at the end of the eleventh segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6123<br>14315<br>22507<br>30699 | 45014 | Float | 0...9999 ☐ | Segment time/gradient 11. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6156<br>14348<br>22540<br>30732 | 45080 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 11. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

## 22 ProG

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6157<br>14349<br>22541<br>30733 | 45082 | Enum | *Enum_SegTyp* | Segment type of segment 12. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6124<br>14316<br>22508<br>30700 | 45016 | Float | -1999...9999 ☐ | End setpoint of segment 12. This is the target setpoint that is reached at the end of the twelfth segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6125<br>14317<br>22509<br>30701 | 45018 | Float | 0...9999 ☐ | Segment time/gradient 12. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProoG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base | 6158 | 45084 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 12. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |
| | | 1dP | 14350 | | | | |
| | | 2dP | 22542 | | | | |
| | | 3dP | 30734 | | | | |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6159<br>14351<br>22543<br>30735 | 45086 | Enum | *Enum_SegTyp* | Segment type of segment 13. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6126<br>14318<br>22510<br>30702 | 45020 | Float | -1999...9999 ☐ | End setpoint of segment 13. This is the target setpoint that is reached at the end of the 13th segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6127<br>14319<br>22511<br>30703 | 45022 | Float | 0...9999 ☐ | Segment time/gradient 13. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

## 22 ProG

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6160<br>14352<br>22544<br>30736 | 45088 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 13. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

## 22 ProG

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6161<br>14353<br>22545<br>30737 | 45090 | Enum | *Enum_SegTyp* | Segment type of segment 14. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6128<br>14320<br>22512<br>30704 | 45024 | Float | -1999...9999 ☐ | End setpoint of segment 14. This is the target setpoint that is reached at the end of the 14th segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6129<br>14321<br>22513<br>30705 | 45026 | Float | 0...9999 ☐ | Segment time/gradient 14. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6162<br>14354<br>22546<br>30738 | 45092 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 14. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6163<br>14355<br>22547<br>30739 | 45094 | Enum | *Enum_SegTyp* | Segment type of segment 15. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| | |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6130<br>14322<br>22514<br>30706 | 45028 | Float | -1999...9999 ☐ | End setpoint of segment 15. This is the target setpoint that is reached at the end of the 15th segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6131<br>14323<br>22515<br>30707 | 45030 | Float | 0...9999 ☐ | Segment time/gradient 15. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

## 22 ProG

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6164<br>14356<br>22548<br>30740 | 45096 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 15. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## PArA

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| tYPE | r/w | base<br>1dP<br>2dP<br>3dP | 6165<br>14357<br>22549<br>30741 | 45098 | Enum | *Enum_SegTyp* | Segment type of segment 16. The segment type defines the setpoint behaviour for this segment. The setpoint can be held constant or be changed with a ramp or a step function. Continuation to next segment is automatic or manual (define a hold time). |

| Value | Description |
|---|---|
| 0 | time to setpoint |
| 1 | rate to setpoint |
| 2 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. |
| 3 | step to setpoint |
| 4 | time to setpoint and wait |
| 5 | rate to setpoint and wait |
| 6 | The final setpoint of the previous segment is kept constant for the duration 'Pt'. At the end of a segment, the programmer enters the Stop mode (Run LED is off), and can be restarted by pressing the Start/Stop key (more than 3 s), via the interface, or a digital input. |
| 7 | step to setpoint and wait |
| 8 | The last segment in a program is the end segment. When the end segment has been reached, the last setpoint is maintained. |
| 9 | timer |
| 10 | timer and hold period. |

| Name | r/w | Adr. | Integer | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 6132<br>14324<br>22516<br>30708 | 45032 | Float | -1999...9999 ☐ | End setpoint of segment 16. This is the target setpoint that is reached at the end of the 16th segment. The target setpoint is approached from the previous valid setpoint. When the program is completed, the controller continues with the last target setpoint reached. |
| Pt | r/w | base<br>1dP<br>2dP<br>3dP | 6133<br>14325<br>22517<br>30709 | 45034 | Float | 0...9999 ☐ | Segment time/gradient 16. The duration of a segment can be defined directly, or by using the segment time and the setpoint difference (SP – segment starting setpoint). Whether the setting is for segment time or the gradient, is defined by means of the segment type parameter (tYPE). |

# 22 ProG

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| d.Out | r/w | base<br>1dP<br>2dP<br>3dP | 6166<br>14358<br>22550<br>30742 | 45100 | Enum | *ENUM_Spuren* | Control outputs 1...4 - 16. A program can control up to four digital signals: the control outputs 1...4. A combination of these signals can be assigned to every segment, whereby the signals are operated while the segment is running. For access to the controller's outputs, the signals must be assigned accordingly. |

| | |
|---|---|
| 0 | 0-0-0-0 |
| 1 | 1-0-0-0 |
| 2 | 0-1-0-0 |
| 3 | 1-1-0-0 |
| 4 | 0-0-1-0 |
| 5 | 1-0-1-0 |
| 6 | 0-1-1-0 |
| 7 | 1-1-1-0 |
| 8 | 0-0-0-1 |
| 9 | 1-0-0-1 |
| 10 | 0-1-0-1 |
| 11 | 1-1-0-1 |
| 12 | 0-0-1-1 |
| 13 | 1-0-1-1 |
| 14 | 0-1-1-1 |
| 15 | 1-1-1-1 |

# 22 ProG

## Signal

| Name | r/w | Adr.Integer | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|
| St.Prog | r | base 6050<br>1dP 14242<br>2dP 22434<br>3dP 30626 | 44868 | Int | 0...255 | ☐ | The programmer's status contains bit-wise coded data, e.g. which point of the program sequence the program has reached. |
| | | | | | Bit 0,1,2 Type of segment<br>0: rising<br>1: falling<br>2: hold (dwell)<br>Bit 3 Program 'Run'<br>Bit 4 Program 'End'<br>Bit 5 Program 'Reset'<br>Bit 6 Program 'StartFlankMissing'<br>Bit 7 Program 'BandHold + FailHold'<br>Bit 8 Program active | | |
| SP.Pr | r | base 6051<br>1dP 14243<br>2dP 22435<br>3dP 30627 | 44870 | Float | -1990...9999 | ☐ | The programmer's setpoint is displayed as the effective setpoint while the program is running. |
| T1.Pr | r | base 6052<br>1dP 14244<br>2dP 22436<br>3dP 30628 | 44872 | Float | 0...9999 | ☐ | Only with a running program. The net (elapsed) time of the programmer is shown in a simplified form as time elapsed since program start.Caution: Stop times are not counted! If the first segment is defined as a gradient, the program starts at the process value, whereby the offset is defined as the time that the controller would have needed with the gradient beginning at the setpoint valid at program start. |
| T3.Pr | r | base 6053<br>1dP 14245<br>2dP 22437<br>3dP 30629 | 44874 | Float | 0...9999 | ☐ | Only with running program. The remaining programmer time is given by the sum of the currently running segment plus the times of the remaining program segments (without hold times). |
| T2.Pr | r | base 6054<br>1dP 14246<br>2dP 22438<br>3dP 30630 | 44876 | Float | 0...9999 | ☐ | Only while program is running. The net segment time corresponds to the elapsed segment time.Caution: Stop times are not counted! If the first segment has bee defined as a gradient, the start commences at process value, and the offset specified for the first segment corresponds to the time that the controller would have required with a gradient beginning at the actual process value when the program was started. |
| T4.Pr | r | base 6055<br>1dP 14247<br>2dP 22439<br>3dP 30631 | 44878 | Float | 0...9999 | ☐ | Only with running program. The remaining time of the running program segment (without hold times). |
| SG.Pr | r | base 6056<br>1dP 14248<br>2dP 22440<br>3dP 30632 | 44880 | Int | 0...16 | ☐ | A program consists of one or more segments which are arranged and defined by means of the segment numbers. By means of the segment number(s), the program can be changed quickly and specifically at the required point. |
| Pr.SG | r/w | base 6060<br>1dP 14252<br>2dP 22444<br>3dP 30636 | 44888 | Int | 1...16 | ☑ | Segment number for Preset. Preset involves starting the selected program with a different segment than the normal (1st) start segment. The starting setpoint of the preset segment becomes effective immediately, i.e. the program is not started. To use the Preset function, the programmer must be in the Stop or Reset state. |

## 22 ProG

### Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Pr.EF | r | base<br>1dP<br>2dP<br>3dP | 6057<br>14249<br>22441<br>30633 | 44882 | Int | 0...16 | ☐ | Number of the active program. The program remains active until a reset or a new start is triggered. |
| SP.En | r | base<br>1dP<br>2dP<br>3dP | 6058<br>14250<br>22442<br>30634 | 44884 | Float | -1999...9999 | ☐ | The segment end set-point of the active segment is displayed. If programmer is in reset, the internal set-point is displayed. |

## 23 SEtP

### PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| SP.LO | r/w | base<br>1dP<br>2dP<br>3dP | 3100<br>11292<br>19484<br>27676 | 38968 | Float | -1999...9999 | ☐ | Lower setpoint limit. The setpoint is raised to this value automatically, if a lower setpoint is adjusted.<br>BUT: The (safety) setpoint W2 is not restricted by the setpoint limits!<br>The setpoint reserve for the step function is 10% of SPHi - SPLo. |
| SP.Hi | r/w | base<br>1dP<br>2dP<br>3dP | 3101<br>11293<br>19485<br>27677 | 38970 | Float | -1999...9999 | ☐ | Upper setpoint limit. The setpoint is reduced to this value automatically, if a higher setpoint is adjusted.<br>BUT: The (safety) setpoint W2 is not restricted by the setpoint limits!<br>The setpoint reserve for the step function is 10% of SPHi - SPLo. |
| SP.2 | r/w | base<br>1dP<br>2dP<br>3dP | 3102<br>11294<br>19486<br>27678 | 38972 | Float | -1999...9999 | ☐ | Second (safety) setpoint. Ramp function as with other setpoints (effective, external). However, SP2 is not restricted by the setpoint limits. |
| r.SP | r/w | base<br>1dP<br>2dP<br>3dP | 3103<br>11295<br>19487<br>27679 | 38974 | Float | 0,01...9999 | ☑ | Setpoint gradient [/min] or ramp. Max. rate of change in order to avoid step changes of the setpoint. The gradient acts in the positive and negative directions.<br>Note for self-tuning: with activated gradient function, the setpoint gradient is started from the process value, so that there is no sufficient setpoint reserve. |
| t.SP | r/w | base<br>1dP<br>2dP<br>3dP | 3104<br>11296<br>19488<br>27680 | 38976 | Float | 0...9999 | ☐ | The timer (preset) value is entered in minutes with one decimal digit (0,1 minute = 6 seconds). With an activated timer, the preset value is displayed automatically in the extended Operating Level, where it can be changed by means of the parameter t.ti. |
| SP.bo | r/w | base<br>1dP<br>2dP<br>3dP | 3105<br>11297<br>19489<br>27681 | 38978 | Float | -1999...9999 | ☐ | Boost increase. Increases the setpoint SP for the duration t.bo by the amount SP.bo. The boost function causes a brief setpoint increase, which is used e.g. to clear blocked channels ('frozen' material) in a hot-runner system. |
| t.bo | r/w | base<br>1dP<br>2dP<br>3dP | 3106<br>11298<br>19490<br>27682 | 38980 | Float | 0...9999 | ☐ | Duration of the boost increase in minutes. When the boost time t.bo has elapsed, the controller switches back to the standard setpoint SP. The boost function causes a brief setpoint increase, which is used e.g. to clear blocked channels ('frozen' material) in a hot-runner system. |

# 23 SEtP

## PArA

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| Y.St | r/w | base<br>1dP<br>2dP<br>3dP | 502<br>13215<br>21407<br>29599 | 42814 | Float | -120...120 | ☐ | Reduced output value for start-up [%]. The start-up function is a protective function, e.g. with hot runner control. To prevent destruction of high-performance heating elements, they must be heated slowly to remove any humidity. With activated start-up function, the controller maintains the reduced starting temperature for a defined dwell period. Subsequently, the controller switches over to the main setpoint. |
| SP.St | r/w | base<br>1dP<br>2dP<br>3dP | 3107<br>11299<br>19491<br>27683 | 38982 | Float | -1999...9999 | ☐ | Setpoint for start-up function. The start-up function is a protective function, e.g. with hot runner control. To prevent destruction of high-performance heating elements, they must be heated slowly to remove any humidity. With activated start-up function, the controller maintains the reduced starting temperature for a defined dwell period. Subsequently, the controller switches over to the main setpoint. |
| t.St | r/w | base<br>1dP<br>2dP<br>3dP | 3108<br>11300<br>19492<br>27684 | 38984 | Float | 0...9999 | ☐ | Start-up dwell period [min]. The start-up function is a protective function, e.g. with hot runner control. To prevent destruction of high-performance heating elements, they must be heated slowly to remove any humidity. With activated start-up function, the controller maintains the reduced starting temperature for a defined dwell period. Subsequently, the controller switches over to the main setpoint. |

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| SP.EF | r | base<br>1dP<br>2dP<br>3dP | 3170<br>11362<br>19554<br>27746 | 39108 | Float | -1999...9999 | ☐ | Effective setpoint. The value reached at the end of setpoint processing, after taking W2, external setpoint, gradient, boost function, programmer settings, start-up function, and limit functions into account. Comparison with the effective process value leads to the control deviation, from which the necessary controller response is derived. |
| Diff | r | base<br>1dP<br>2dP<br>3dP | 3171<br>11363<br>19555<br>27747 | 39110 | Float | -1999...9999 | ☐ | Difference between the effective setpoint and setpoint 2 |
| SP | r/w | base<br>1dP<br>2dP<br>3dP | 3180<br>11372<br>19564<br>27756 | 39128 | Float | -1999...9999 | ☐ | Setpoint for the interface (without the additional function 'Controller off'). SetpInterface acts on the internal setpoint before the setpoint processing stage. Note: The value in RAM is always updated. To protect the EEPROM, storage of the value in the EEPROM is timed (at least one value per half hour). |
| SP.d | r/w | base<br>1dP<br>2dP<br>3dP | 3181<br>11373<br>19565<br>27757 | 39130 | Float | -1999...9999 | ☐ | The effective setpoint is shifted by this value. In this way the setpoints of several controllers can be shifted together, regardless of the individually adjusted effective setpoints. |

# 23 SEtP

## Signal

| Name | r/w | Adr.Integer | | real | Typ | Value/off | | Description |
|---|---|---|---|---|---|---|---|---|
| t.ti | r/w | base | 3182 | 39132 | Float | 0...9999 | ☐ | Current timer count in minutes. Count-down timer. The run time is only visible, if the timer is active. Configuration in the extended Operating Level. |
| | | 1dP | 11374 | | | | | |
| | | 2dP | 19566 | | | | | |
| | | 3dP | 27758 | | | | | |

# 24 Tool

## ConF

| Name | r/w | Adr.Integer | | real | Typ | Value/off | Description |
|---|---|---|---|---|---|---|---|
| U.LinT | r/w | base | 634 | 34036 | Enum | *Enum_Unit* | Engineering unit of linearization table (temperature). |
| | | 1dP | 8826 | | | | |
| | | 2dP | 17018 | | | | |
| | | 3dP | 25210 | | | | |

| | |
|---|---|
| 0 | without unit |
| 1 | °C |
| 2 | °F |