Software manual
SCHUNK motion protocol

Motion Control Schunk V 2.11

Superior Clamping and Gripping

Imprint

Document number: 0389052

Version: 02.00 | 17/01/2022 | en

Dear Customer,

thank you for trusting our products and our family-owned company, the leading technology supplier of robots and production machines.

Our team is always available to answer any questions on this product and other solutions. Ask us questions and challenge us. We will find a solution!

Best regards,

Your SCHUNK team

Customer Management

Tel. +49-7133-103-2503
Fax +49-7133-103-2189

cmg@de.schunk.com

Please read the operating manual in full and keep it close to the product.

Table of Contents

1 General ... 7
- 1.1 Presentation of Warning Labels ... 7
- 1.2 Electrical connection ... 8
- 1.3 Display elements ... 9
- 1.4 Factory settings ... 9
- 1.5 Booting ... 9
- 1.6 SMP protocol for CAN/Profibus/USB ... 10
  - 1.6.1 Data format ... 10
  - 1.6.2 Data frame ... 12
  - 1.6.3 Special features in USB communication ... 13
  - 1.6.4 Special features with CAN ... 13
  - 1.6.5 Special features with Profibus ... 14
  - 1.6.6 Fragmentation ... 16
- 1.7 User administration ... 18
  - 1.7.1 User ... 18
  - 1.7.2 Profi ... 18
  - 1.7.3 Advanced ... 18
  - 1.7.4 Root ... 18
  - 1.7.5 Schunk ... 19
  - 1.7.6 Disabled ... 19
- 1.8 Pseudo-absolute encoder ... 19
  - 1.8.1 Prerequisites ... 19
  - 1.8.2 Function ... 19
- 1.9 Standstill commutation ... 21
  - 1.9.1 Prerequisites ... 21
  - 1.9.2 Function ... 21

2 Unit system ... 22

3 Commands ... 23
- 3.1 Movement ... 24
  - 3.1.1 CMD REFERENCE ... 24
  - 3.1.2 MOVE POS ... 25
  - 3.1.3 MOVE POS REL ... 27
  - 3.1.4 MOVE VEL ... 29
  - 3.1.5 SET TARGET VEL ... 30
  - 3.1.6 SET TARGET ACC ... 30
  - 3.1.7 SET TARGET JERK ... 31
  - 3.1.8 SET TARGET CUR ... 32
  - 3.1.9 SET TARGET POS ... 33
  - 3.1.10 SET TARGET POS REL ... 33
  - 3.1.11 CMD STOP ... 34
  - 3.1.12 CMD FAST STOP ... 35
- 3.2 Spontaneous messages ... 36
  - 3.2.1 CMD INFO ... 36

3.2.2 CMD MOVE BLOCKED ... 37
3.2.3 CMD POS REACHED ... 38
3.2.4 CMD ERROR ... 38
3.2.5 GET STATE ... 39
3.2.6 CMD TOGGLE IMPULSE MESSAGE ... 42
3.3 Settings ... 42
3.3.1 CALIB CURRENT ... 42
3.3.2 SET CONFIG EXT ... 43
3.3.3 GET CONFIG EXT ... 44
3.4 Other ... 45
3.4.1 CMD REBOOT ... 45
3.4.2 CHANGE USER ... 45
3.4.3 CMD GET DIO (0xE9) ... 45
3.5 Fragmentation ... 47
3.5.1 FRAG ACK (0x87) ... 47
3.5.2 FRAG START (0x84) ... 47
3.5.3 FRAG MIDDLE (0x85) ... 47
3.5.4 FRAG END (0x86) ... 48
4 Info and error messages ... 49
4.1 Troubleshooting ... 49
4.1.1 CMD ERROR ... 49
4.1.2 CMD WARNING ... 50
4.1.3 CMD INFO ... 50
4.1.4 CMD ACK ... 50
4.1.5 Detailed error information ... 50
4.2 Information codes and error codes ... 51
4.2.1 INFO BOOT ... 51
4.2.2 INFO NO RIGHTS ... 51
4.2.3 INFO UNKNOWN COMMAND ... 51
4.2.4 INFO FAILED ... 51
4.2.5 NOT REFERENCED ... 51
4.2.6 INFO SEARCH SINE VECTOR ... 51
4.2.7 INFO NO ERROR ... 52
4.2.8 INFO COMMUNICATION ERROR ... 52
4.2.9 INFO TIMEOUT ... 52
4.2.10 INFO WRONG DATA TYPE ... 52
4.2.11 INFO RESTART ... 52
4.2.12 INFO CHECKSUM ... 52
4.2.13 INFO VALUE LIMIT MAX ... 52
4.2.14 INFO VALUE LIMIT MIN ... 53
4.2.15 INFO MESSAGE LENGTH ... 53
4.2.16 INFO WRONG PARAMETER ... 53
4.2.17 INFO UNKNOWN PARAMETER ... 53
4.2.18 ERROR FILE NOT FOUND ... 53
4.2.19 ERROR FILE IS CORRUPT ... 53

4.2.20 ERROR FILE TYPE WRONG ... 53
4.2.21 ERROR FILE SYSTEM WRONG ... 53
4.2.22 ERROR FILE READ ... 53
4.2.23 ERROR FILE IS NOT CREATED ... 54
4.2.24 ERROR FILE WRITE ... 54
4.2.25 ERROR REBOOT ... 54
4.2.26 ERROR MOTOR PHASE ... 54
4.2.27 ERROR WRONG RAMP TYPE ... 54
4.2.28 ERROR WRONG DIRECTION ... 54
4.2.29 ERROR CONFIG MEMORY ... 54
4.2.30 ERROR SOFT LOW ... 54
4.2.31 ERROR SOFT HIGH ... 55
4.2.32 ERROR SERVICE ... 55
4.2.33 ERROR FAST STOP ... 55
4.2.34 ERROR TOW ... 55
4.2.35 ERROR VPC3 ... 55
4.2.36 ERROR FRAGMENTATION ... 55
4.2.37 ERROR COMMUTATION ... 56
4.2.38 ERROR I2T ... 56
4.2.39 ERROR CURRENT ... 56
4.2.40 ERROR TOO FAST ... 56
4.2.41 ERROR POS SYSTEM ... 56
4.2.42 ERROR RESOLVER CHECK FAILED ... 56
4.2.43 ERROR MATH ... 56
4.2.44 ERROR CALIB CURRENT ... 57
4.2.45 ERROR INITIALIZE ... 57
4.2.46 ERROR INTERNAL ... 57
4.2.47 ERROR CONNECTION TEMP LOW ... 57
4.2.48 ERROR CONNECTION TEMP HIGH ... 57
4.2.49 ERROR MOTOR TEMP LOW ... 58
4.2.50 ERROR MOTOR TEMP HIGH ... 58
4.2.51 ERROR TEMP LOW ... 58
4.2.52 ERROR TEMP HIGH ... 58
4.2.53 ERROR LOGIC LOW ... 58
4.2.54 ERROR LOGIC HIGH ... 58
4.2.55 ERROR MOTOR VOLTAGE LOW ... 59
4.2.56 ERROR MOTOR VOLTAGE HIGH ... 59
4.2.57 ERROR CABLE BREAK ... 59
4.2.58 ERROR LIFE SIGN ... 60
4.2.59 ERROR CUSTOM DEFINED ... 60
4.2.60 ERROR OVERSHOOT ... 60
4.2.61 ERROR HARDWARE VERSION ... 60
4.2.62 ERROR SOFTWARE VERSION ... 60

5 Configuration parameters ... 61

5.1 General ... 61
5.2 Parameterization with the help of a USB stick ... 62

5.2.1 Reading parameters 62
5.2.2 Write parameters 62
5.3 Parameter 62
5.3.1 Cyclical 62
5.3.2 Permanent 66
5.3.3 Asynchronous 102

6 Remarks 111
6.1 Quick test 111
6.2 Modules 111
6.3 Protocol 111
6.4 Profibus 111

7 Appendix 112
7.1 List of the enumerations 112
7.2 Information codes and error codes 115
7.3 Firmware update 117

1 General

1.1 Presentation of Warning Labels

To make risks clear, the following signal words and symbols are used for safety notes.

⚠ DANGER

Danger for persons!

Non-observance will inevitably cause irreversible injury or death.

⚠ WARNING

Dangers for persons!

Non-observance can lead to irreversible injury and even death.

⚠ CAUTION

Dangers for persons!

Non-observance can cause minor injuries.

CAUTION

Material damage!

Information about avoiding material damage.

1.2 Electrical connection

The module has separate input terminals for the motor voltageand the logic voltage (24V DC). We recommend you connect the two voltages separately. This ensures that the logic continues to work in the event of the motor voltage overloading and that the module status is always known. In modules with a motor voltage > 24V DC, a separate power supply is unavoidable in any case, since the logic voltage must always be in a range between 18V DC and 32V DC.

CAUTION

Permanent damage to the electronics possible!

- If the power supply is separated, you must make sure that potential equalization is carried out between the two supply voltages (join the grounds)
- Only the positive pole may be switched off; the motor GND must always remain connected.

NOTE

If the power supply for the logic and motor is separate, the supply voltage of the motor can be switched off from the control unit by means of relays, but the module remains active on the bus system.

When the power unit (motor) is supplied with power, use a power supply unit that is capable of supplying enough power to the respective module. Make sure there is adequate cable cross-section when cabling.

The voltage drop on the cable can be calculated using the following formula:

$\Delta U = (2 \times I \times l) / (\chi \times A)$ with:

I: Current input of load
l: Length of line
χ: electrical conductivity
Cu: $\chi = 56\ ((m) / (\Omega \times mm^2))$
Al: $\chi = 35\ ((m) / (\Omega \times mm^2))$
A: conductor diameter

1.3 Display elements

See the instructions for the respective module.

1.4 Factory settings

See the instructions for the respective module.

1.5 Booting

Default values for motions are predefined for the module as standard values. This allows the module to be commissioned directly without having to set the parameters beforehand. The following default values apply after the restart:

- "Target speed"
  as [%] of the maximum value -> 10%, ▶ 5.3.2.2.7 [🗋 73]
- "Target acceleration"
  as [%] of the maximum value -> 10%, ▶ 5.3.2.2.8 [🗋 73]
- "Target jerk"
  as [%] of the maximum value -> 50%, ▶ 5.3.2.2.9 [🗋 73]
- "Target current"
  Rated current, ▶ 5.3.2.2.6 [🗋 72]
- Spontaneous messages activated.
- User is set to "User" ▶ 1.7 [🗋 18]

1.6 SMP protocol for CAN/Profibus/USB

1.6.1 Data format

Data is sent from the modules in the Intel format (little-endian) and interpreted in this same format when being received.

Floating-point numbers

The IEEE 754 standard (floating-point numbers) was developed in the early 1980s in order to, among other things, attain consistent floating-point number representation using various computer architecture. This standard is adhered to if the parameters are sent as floating-point numbers to the module or are transmitted from the module to the control unit. A floating-point number is shown here with 32 bits.

Sign bit	Exponent	Mantissa (standardized)
1 bit (bit 32)	8 bits (bit 23.. bit 30)	23 bits (bit 1.. bit 22)
s	*e*	*f*

Since the mantissa is always standardized to "1", only the digits after the decimal point are saved. The leading "1" is not saved concurrently. A float value can be calculated as follows.

$(-1)^s \times 2^{e-127} \times (1.\,f\,)_{bin}$

Examples:

	Sign	Exponent	Mantissa
	1 bit	**8 bits**	**23 bits**
7/4	0	01111111	11000000000000000000000
−34.432175	1	10000100	00010011011101010001100
-959818	1	10010010	11010100101010010100000
0	0	00000000	00000000000000000000000
0	1	00000000	00000000000000000000000
$2^{-126} = 1.175 \times 10^{-38}$			
Smallest positive number	0	00000001	00000000000000000000000
$(2-2^{-23})\ 2^{127} = 3.403\times10^{38}$			
Largest positive number	0	11111110	11111111111111111111111
Infinite	0	11111111	11111111111111111111111
NaN	0	11111111	Not all "0" or "1"
$2^{-23} = 1.192 \times 10^{-7}$			
Smallest number to be resolved	0	01101000	00000000000000000000000
2^{-128}	0	00000000	01000000000000000000000

Two's complement

The two's complement is an option for representing negative numbers in the binary system. In the module, the two's complement is required for the representation of negative integers.

Positive numbers are provided in the two's complement with a leading 0 (sign bit) and otherwise not changed. Negative numbers are provided with a leading 1 as the sign bit and coded as follows: All digits in the relevant positive number are negated. 1 is added to the result. Exemplary conversion of the negative decimal number -4(dec) into the two's complement:

1. Ignore the sign and convert to the binary system: $4_{dec} = 00000100_{bin} = 0x\ 04_{hex}$
2. Invert, since negative: $11111011_{bin} = 0x\ FB_{hex}$
3. Add one, since negative: $11111011_{bin} + 00000001_{bin} = 11111100_{bin} = 0x\ FC_{hex} = -4_{dec}$

Somewhat more mathematical:
If *x* is a negative number, then *x* is calculated in the two's complement (x_z) with **n** places as follows:

$x_z = 2^n - |x|$

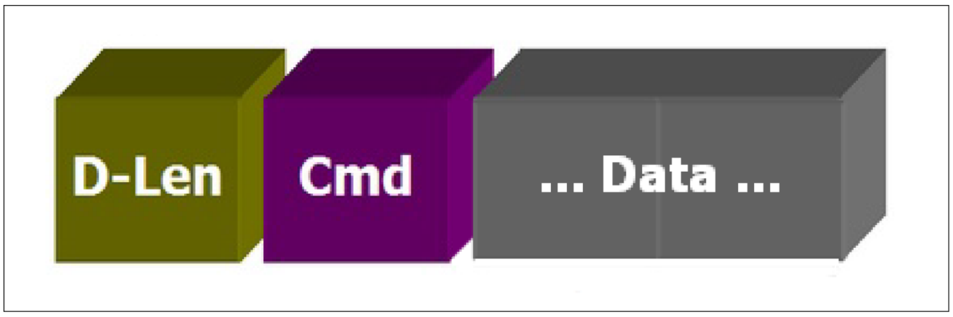
Accordingly, the following also applies:

$x_z + |x| = 2^n$

1.6.2 Data frame

The data frame of the Motion protocol always includes the following elements:

- D-Len (1 byte)
- COMMAND CODE (1 byte)



Data frame

D-Len (Data Length) specifies the number of subsequent items of user data including the command byte. D-Len is one byte wide, therefore a Motion protocol message can transfer a maximum of 254 data bytes.

The D-Len byte is always followed by the command code, consisting of one byte. If necessary, the command code is followed by the relevant parameters that are required. The command code can also be extended further using "Subcommand codes," if needed.

All commands sent are immediately confirmed by the module with a response (acknowledge). This response also uses the data frame described above. (D-Len, command code, parameters, if necessary). If the request has been successfully processed, D-Len always has a value that is not equal to "0x02". If the request failed, D-Len has the value "0x02".

The cause of the *failed request* is described in the two following bytes.

Furthermore, the modules have the capacity to send messages without a request having been made. The data frame is also adhered to in the event of such "spontaneous messages". A "spontaneous message" is triggered in the following events:

- When a serious error has occurred.
- When a motion has been concluded correctly.
- When there is a regular *status message*, if activated ▸ 3.2 [🗋 36].

1.6.3 Special features in USB communication

USB communication is only intended for parameterization and commissioning. For USB access, it is recommended to use the Motion Tool Schunk. (see "Schunk Motion Tool" software manual) zu verwenden.

1.6.4 Special features with CAN

As a message oriented bus system, CAN needs - in addition to the data frame - a unique identifier allocated to the message. The modules support the standard 11-bit identifier. The lower 8 bits are used here for the unique module ID. Therefore, a maximum of 255 modules can be addressed.

The remaining 3 bits are coded as follows:

- 0x03 - Module error message
- 0x05 - Message from the master to a module
- 0x07 - Response from the module

All other states are not used.

- A message to the module thus has the following identifier 0x5*XX*.
  (*XX* module address in hex representation)
- A message from the module has the identifier 0x7*XX*.
  (*XX* module address in hex representation)
- In the event of an error, the messages from the module to the master are sent with the identifier 0x3*XX*.
  (*XX* module address in hex representation)

A maximum of 8 bytes can be sent in a CAN message. Under certain circumstances, it will be necessary to pack a longer data frame (D-Len > 7) into several CAN messages. This is done by using the fragmentation protocol.
▸ 1.6.6 [🗋 16]

NOTE

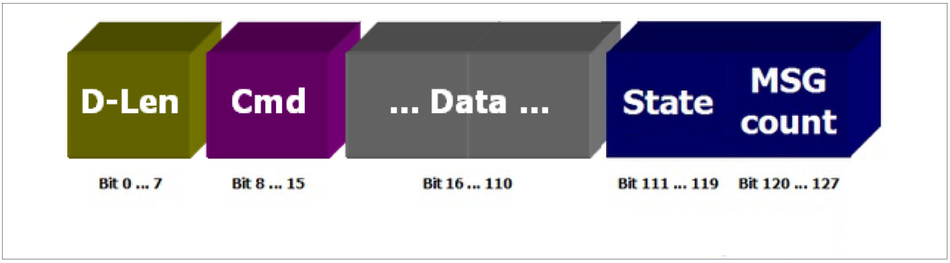
Fragmentation is normally not required, since all commands needed to operate the modules can be packed into one CAN message.

1.6.5 Special features with Profibus

There are the following special features with Profibus DPV0:

An *alternative Profibus protocol* can be selected

The alternative SDP Profibus protocol can be activated directly via the PLC during configuration of the module.
In the protocol described here, the maximum length of the data to be transmitted all at once from the master to the module is limited to 8 bytes. This can be used to fully operate a module. (A maximum 7 bytes are needed for a message from the master to the module)



Data frame for Profibus

The maximum length of the data sent from the module to the master (response) is limited to 16 bytes (GSD file). If you need to send/receive larger quantities of data, a *fragmentation* ▸ 1.6.6 [🗋 16] may be necessary. The longest message arising in normal operation from the module to the master can be accommodated in 16 bytes. There will be 2 bytes still remaining. Contained in these 2 bytes, which are always at the end of the Profibus message (byte 14, byte 15), are

1. the actual *status* ▸ 3.2.5 [🗋 39] of the module (byte 14) **and**
2. a so-called command counter (MsgCount) (byte 15)

.
If bytes 10-13 are not required, the actual position in the respective *unit system* ▸ 2 [🗋 22] can be read out here.

NOTE

Bytes 10-13 are only used in a response to a ▸ 3.2.5 [🗋 39] if all data (position, speed, and current) are requested.
Bytes 10-15 are used for data in fragmented messages.

NOTE

Only the upper 8 bits of the status word are written. The error code is not used. You can use the extended diagnosis under Profibus for this on the one hand, and on the other, the error code ▸ 4.1 [🗋 49] is displayed in the output data in the event of an error.
If a message is sent from the master to the module, then with Profibus the MsgCount is increased by 1 in addition to the response. This ensures that every request is confirmed despite possible spontaneous messages.

NOTE

A spontaneous message ▸ 3.2 [🗋 36] does not increase the MsgCount!

If, for example, you want to move to a position where the module is currently located, the module will signal "Command understood" and "Position reached" immediately in the next Profibus cycle. Since in some circumstances a control unit connected to the Profibus does not query the data in every Profibus cycle, the acknowledgment of (response to) the motion command could go missing. The MsgCount ensures that a confirmation of the request was received.

The *status byte* ▸ 3.2.5 [🗋 39] (byte 14) is used to constantly receive the current information on the status of the present module.

NOTE

The last bit for MsgCount can be evaluated as a toggle bit. (Module to master) During data transmission from the master to the module the byte 8 not used can be used as a toggle byte, or bit 63 as a toggle bit.

Groups are fully supported by the SYNC / FREEZE mechanism implemented in Profibus.

If consistent data transmission is not possible, then the following options can be used to operate the module:

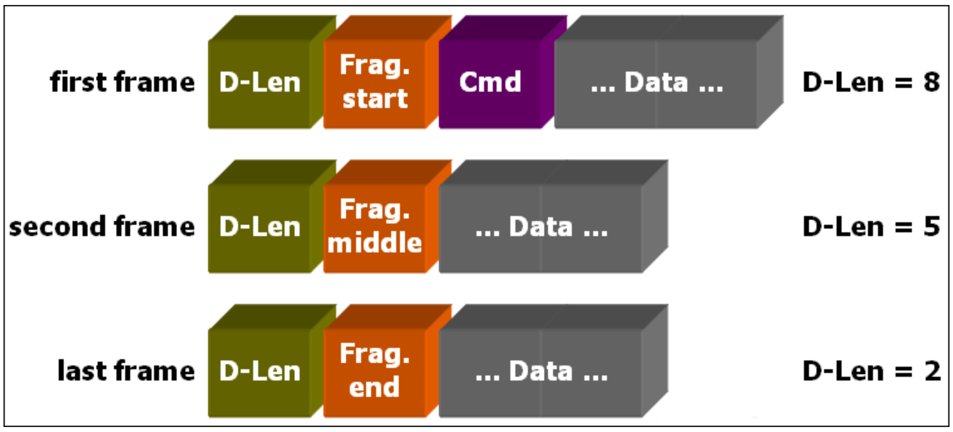
- Using the SYNC, UNSYNC mechanism.
- Set D-Len to "0". Populate all data and set the D-Len as soon as all data are present.

1.6.6 Fragmentation

NOTE

The fragmentation of messages is not required for normal operation!

Should a fragmentation of messages be required, this is done as follows:



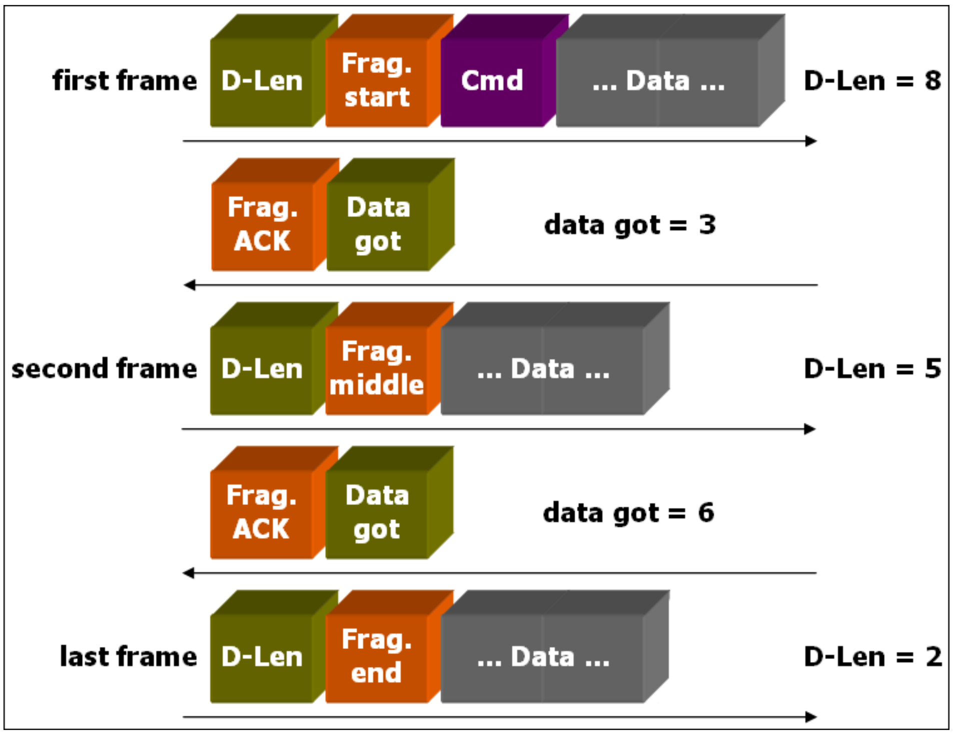
Fragmentation

The length of the user data still following is sent at the beginning of each message. Then a fragment identifier is sent. This fragment identifier is **not** recorded in the length byte (D-Len).

- FragStart -> *First fragment*
- FragMiddle -> *A middle fragment*
- FragEnd -> *Last fragment*

These individual fragments can thus be reassembled into a complete *data frame* which can then be interpreted.

Special feature with Profibus



Profibus fragmentation

Since, in Profibus, a "token" is constantly underway out of which the respective participants pick out the valid data for themselves or register the required data for the master, each fragment received must be confirmed using *"FRAG ACK"* and the D-Len byte for the received fragment. If a fragmented message is sent to the master, then the master has to confirm every single fragment using "FRAG ACK" and the D-Len byte for the received fragment so that the module can send the next fragment. If the master sends a fragmented message to the module, then it has to delay sending the following fragment until the module has confirmed the receipt of the fragment ("FRAG ACK" and the D-Len byte for the received fragment). The last fragment does not have to be confirmed.

1.7 User administration

The module is equipped with a user administration to specifically protect certain actions. The user rights can be changed via *"CHANGE USER"* ▶ 3.4.2 [🗋 45] or by the "User" parameter (0x7DDA) ▶ 5.3.3.33 [🗋 110].

1.7.1 User

Is the standard user who is always activated after the module is switched on. The user can fully operate the module. Parameterizing is only permitted for the most important *parameters* ▶ 5 [🗋 61].

1.7.2 Profi

Is the professional user having the full range of functions of the "user" and can also write other parameters. Incorrect parameterization can result in unanticipated module behavior. However, the module cannot be destroyed.

The standard password for the professional rights is "Schunk".

1.7.3 Advanced

Is the advanced user who has the full range of functions of the "Profi" and can also adjust further parameters.

CAUTION

Incorrect operation or incorrect parameterization can lead to the destruction of the electronics or the motor.

1.7.4 Root

Is the root user who has full access to the module. All parameters can be adjusted and other functions are accessible for test purposes.

CAUTION

Incorrect operation or incorrect parameterization can lead to the destruction of the electronics or the motor.

1.7.5 Schunk

Parameters with Schunk users can only be modified in the production facility or by qualified service personnel.

1.7.6 Disabled

Parameters with "Disabled user" cannot be modified.

1.8 Pseudo-absolute encoder

1.8.1 Prerequisites

The modules support the pseudo absolute encoder function if the following prerequisites are met:

- Brake
- FRAM *(Hardware version* odd-numbered)
- *Position measuring system* ▸ 5.3.2.5.2 [🗋 89] resolver
- or position measuring system Encoder with index track and
  - Motor type *DC* ▸ 5.3.2.2.3 [🗋 71]
  - or BLDC motor type
  - or PMSM motor type

1.8.2 Function

When the brake is applied, the current position is saved in a non-volatile memory. If the logic voltage is switched off, then an attempt is made to save the current position with the remaining residual energy.

Resolver

When the module is switched on again, the position saved beforehand will now be compared with a control value. Should this check be successful, then the position saved will be compared with the current position of the resolver. If these positions are also equal, then the module does not need to be re-referenced.

NOTE

If the resolver is turned by precisely one revolution when deenergized, then the displayed position will be faulty when reactivation takes place.

Encoder with index track

When the module is switched on again, the position saved beforehand will be compared with a control value. Should this check be successful, then the saved position will become the current position. The interval between the next index pulse is calculated at the same time.

In the first motion command that follows, the calculated interval is compared with the measured interval when the index pulse is reached. The module is deemed as referenced if the two values agree with each other. In addition, the index pulse must be reached within a certain period of time after the first motion command has been sent.

If an *error*
▸ 4.1 [🗋 49] occurs during the movement to the index pulse, then the referencing will be deleted.

NOTE

After a successful reference movement, the index pulse has to be run over at least once in order to activate the function.

NOTE

If the encoder is moved in a deenergized state, then it is possible that the module will carry out a motion to the next index pulse (no more than one motor revolution) after it has been switched on again with the incorrect position.

NOTE

If the encoder is turned by precisely one revolution when deenergized, then the displayed position will be faulty when reactivation takes place.

1.9 Standstill commutation

1.9.1 Prerequisites

- Motor type *DC* ▸ 5.3.2.2.3 [🗋 71]
- or BLDC motor type
- or PMSM motor type and
  - *Position measurement system* ▸ 5.3.2.5.2 [🗋 89] "Encoder with index track" and available Hall sensors
  - or resolver position measuring system

CAUTION

Direction of motion for block commutation and sine commutation must agree. If the directions of rotation are different, then the phases must be changed and the commutation table ▸ 5.3.2.2.14 [🗋 75] adapted.

1.9.2 Function

If all prerequisites are met, then the module will attempt to carry out the standstill commutation. In modules with absolute-value measuring systems, the sine commutation can be activated directly after switch-on, since the position of the "sine pointer" is known.
In modules with an encoder measuring system, the position of the "sine pointer" is not known until the index pulse has been reached. Therefore, the module is moved with block commutation up to the first index pulse (for the movement with block commutation, the force may be slightly lower) and then converted to sine commutation.

The position of the "sine pointer" to the index pulse is set or readjusted using the parameter *positioning offset* Positioning. If this value is set to "0", then a "sine pointer" - which is *saved* Positioning- is searched for in the next motion command by energizing the motor phases. The *referencing* ▸ 1.8 [🗋 19] is deleted in the process.

NOTE

The module should be able to move freely in all directions for the pointer search. The module is moved in a jerking fashion up to two motor revolutions. Communication with the module is not possible during this time.

2 Unit system

For linear and gripper products, the following unit system is specified:

- Position [mm]
- Speed [mm/s]
- Acceleration [mm/s^2]
- Jerk [mm/s^3]
- Current values [A]
- Times [s]

For rotational units, the following applies:

- Position [degree]
- Speed [degree/s]
- Acceleration [degree/s^2]
- Jerk [degree/s^3]
- Current values [A]
- Times [s]

3 Commands

NOTE

This chapter does not apply to the Schunk Drive Protocol (SDP).

All examples only list the data frame. The special features of the different bus systems are described starting in chapter ▸ 1.6.3 [🗋 13]. Several selected examples for the various bus systems are listed in the *Appendix* ▸ 7 [🗋 112].

NOTE

It is assumed for all listed examples that a gripper is connected which is moved in [mm].

In all examples, only the *necessary parameters* are listed, not the *optional parameters*. In the examples, "M" stands for master and "S" for slave (module).

3.1 Movement

3.1.1 CMD REFERENCE

COMMAND CODE: 0x92

DESCRIPTION: A reference movement is executed. The type of referencing is defined once in the *configuration parameter* ▶ 5.3.2.4.1 [🗋 82].

PARAMETERS (Master -> Slave): None

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful. The module executes the command.

EXAMPLE:

	D-Len	Cmd	Param
M->S	0x01	0x92	
S->M	0x03	0x92	0x4F 0x4B

MISCELLANEOUS: Spontaneous response possible. Depending on the *referencing type* ▶ 5.3.2.4.1 [🗋 82], this can result in *"CMD MOVE BLOCKED"*▶ 3.2.2 [🗋 37] or *"CMD POS REACHED"* ▶ 3.2.3 [🗋 38]. This depends on the *"MOVE ZERO AFTER REFERENCING"*▶ 5.3.2.4.4 [🗋 87] flag. A set flag triggers a positioning movement after the referencing => *"CMD POS REACHED"*▶ 3.2.3 [🗋 38]. During referencing, the *set parameters*▶ 5.3.2.4.5 [🗋 87] are accepted for the movements. After referencing, the values set last (prior to the reference movement) are re-established. Please also observe the special features of the respective *position measuring system* ▶ 5.3.2.5.2 [🗋 89].

NOTE

Under certain conditions, a referencing process successfully performed once is maintained after the operating voltages have been switched off. For more about this, see "Pseudo-absolute encoder" ▶ 1.8 [🗋 19]

NOTE

Prior to a reference movement, all workpieces must be removed for a gripper.

3.1.2 MOVE POS

COMMAND CODE: 0xB0

DESCRIPTION: Moves the module to a specified position. The position is set in the configured *unit system* ▸ 2 [🗋 22]. The positioning movement is based on the *configured motion profile* ▸ 5.3.2.5.11 [🗋 92].

PARAMETERS (Master -> Slave)*:

- *Position* ▸ 3.1.9 [🗋 33] (optional), in the configured unit system.
- *Speed* ▸ 3.1.5 [🗋 30] (optional) which is used for the positioning movement. Not relevant for the motion profile *"No ramp"* ▸ 5.3.2.5.11 [🗋 92].
- *Acceleration* ▸ 3.1.6 [🗋 30] (optional) which is used for the positioning movement. Not relevant for the motion profile *"No ramp"* ▸ 5.3.2.5.11 [🗋 92].
- *Current* ▸ 3.1.8 [🗋 32] (optional) which must not be exceeded in the positioning movement. If the controller structure *"CURRENT SPEED"* ▸ 5.3.2.3.15 [🗋 80] is active, then this value must be transferred (jerk is required). The value has to be at least "0", otherwise *"INFO WRONG PARAMETER"* ▸ 4.2.16 [🗋 53] will be generated.
- *Jerk* (optional) which is used for the positioning movement. Should the motion profile not be equal to *"jerk-limited"* ▸ 5.3.2.5.11 [🗋 92], then this value cannot also be transferred (*"INFO WRONG PARAMETER"* ▸ 4 [🗋 49]).

ANSWER (Slave -> Master): If possible, the time that the module is expected to need for the movement is returned. If a calculation of the time is not possible, then the request will be confirmed with "OK" (0x4F4B) in the event of success, and the module will execute the movement.

NOTE

If the motion profile is configured to "jump", then the jerk-limited motion profile is used. In motion profile "jump", the drive would execute vigorous movements.

EXAMPLE 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB0	0x00 0x00 0x20 0x41	Move to position 10.0 [mm]
S->M	0x05	0xB0	0xCD 0xCC 0x04 0x41	Expect to reach position in 8.3 [s]

EXAMPLE 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB0		Move to the position which was set last
S->M	0x05	0xB0	0xCD 0xCC 0x04 0x41	Expect to reach position in 8.3 [s]

MISCELLANEOUS: Spontaneous response when the position is reached *"CMD POS REACHED"* ▸ 3.2.3 [🗋 38] or when the positioning movement is blocked *"MOVE BLOCKED"* ▸ 3.2.2 [🗋 37].

All parameters must be transferred in the specified order. If just the current is to be specified, then it is vital that the position, speed, and acceleration are also specified. The following parameters do not have to be transferred as well. All parameters are retained up to the restart or until these parameters are changed.

3.1.3 MOVE POS REL

COMMAND CODE: 0xB8

DESCRIPTION: Moves the module a specified distance. The change of position is set in the configured *unit system* ▸ 2 [🗋 22]. The positioning movement is based on the *configured motion profile* ▸ 5.3.2.5.11 [🗋 92].

PARAMETERS (Master -> Slave)*:

- *Position change* ▸ 3.1.10 [🗋 33] (optional), in the configured unit system
- *Speed* ▸ 3.1.5 [🗋 30] (optional) which is used for the positioning movement. Not relevant for the motion profile *"No ramp"* ▸ 5.3.2.5.11 [🗋 92].
- *Acceleration* ▸ 3.1.6 [🗋 30] (optional) which is used for the positioning movement. Not relevant for the motion profile *"No ramp"* ▸ 5.3.2.5.11 [🗋 92].
- *Current* ▸ 3.1.8 [🗋 32] (optional) which must not be exceeded in the positioning movement. If the controller structure *"CURRENT SPEED"*▸ 5.3.2.3.15 [🗋 80] is active, then this value must be transferred (since jerk is required). The value has to be at least "0", otherwise *"INFO WRONG PARAMETER"* ▸ 4.2.16 [🗋 53] will be generated.
- *Jerk* (optional) which is used for the positioning movement. Should the motion profile not be equal to *"jerk-limited"* ▸ 5.3.2.5.11 [🗋 92], then this value cannot also be transferred (*"INFO WRONG PARAMETER"* ▸ 4 [🗋 49]).

NOTE

If the motion profile is configured to "jump", then the jerk-limited motion profile is used. In motion profile "jump", the drive would execute vigorous movements.

ANSWER (Slave -> Master): If possible, the time that the module is expected to need for the movement is returned. If a calculation of the time should not be possible, then the request will be confirmed with "OK" (0x4F4B) in the event of success, and the module will execute the movement.

EXAMPLE 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB8	0x00 0x00 0x20 0x41	Move further by 10.0[mm]
S->M	0x05	0xB8	0xCD 0xCC 0x04 0x41	Expect to need 8.3[sec] for this

EXAMPLE 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB8		Continue to move by the distance set last
S->M	0x05	0xB8	0xCD 0xCC 0x04 0x41	Expect to need 8.3[sec] for this

Other: Spontaneous response when the position is reached *"CMD POS REACHED"* ▸ 3.2.3 [🗋 38] or when the positioning movement is blocked *"MOVE BLOCKED"* ▸ 3.2.2 [🗋 37]. All parameters must be transferred in the specified order. If just the current is to be specified, then it is vital that the change of position, speed, and acceleration are also specified. The following parameters do not have to be transferred as well. All parameters are retained up to the restart or until these parameters are changed.

3.1.4 MOVE VEL

COMMAND CODE: 0xB5

DESCRIPTION: A speed movement is executed.

PARAMETERS (Master -> Slave):

- *Speed* ▸ 3.1.5 [🗋 30] in the configured *unit system* ▸ 2 [🗋 22].
- *Current* ▸ 3.1.8 [🗋 32] (optional) which must not be exceeded in the speed movement. Generates *"INFO WRONG PARAMETER"* ▸ 5.3.2.3.15 [🗋 80] in controller structure *"CURRENT SPEED"* ▸ 4 [🗋 49].

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful. The module executes the command.

EXAMPLE 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB5	0x9A 0x99 0x31 0x41	Execute speed movement with 11.1[(mm)/(s)]
S->M	0x03	0xB5	0x4F 0x4B	

EXAMPLE 2:

	D-Len	Cmd	Param	
M->S	0x05	0xB5		Execute speed movement with speed set last
S->M	0x03	0xB5	0x4F 0x4B	

MISCELLANEOUS: Other: Spontaneous message *"CMD MOVE BLOCKED"* ▸ 3.2.2 [🗋 37] is possible if the module was blocked during the movement.

3.1.5 SET TARGET VEL

COMMAND CODE: 0xA0

DESCRIPTION: This is used to set the velocity parameter.

PARAMETERS (Master -> Slave):

- *Speed* in the *specified unit system* ▸ 2 [🗋 22].

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x05	0xA0	0x33 0x33 0x43 0x41	Set speed to 12.2 [mm/s]
S->M	0x03	0xA0	0x4F 0x4B	

MISCELLANEOUS: This value is retained after it has been written successfully once until the module is restarted or this value is changed again. The value is not adopted until the next motion command and is not immediately active.

3.1.6 SET TARGET ACC

COMMAND CODE: 0xA1

DESCRIPTION: This is used to set the acceleration parameter.

PARAMETERS (Master -> Slave):

- *Acceleration* in the specified *unit system* ▸ 2 [🗋 22].

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x05	0xA1	0x00 0x00 0xF0 0xA2	Set the acceleration to 120 [mm/s^2]
S->M	0x03	0xA1	0x4F 0x4B	

MISCELLANEOUS: This value is retained after it has been written successfully once until the module is restarted or this value is changed again. The value is not adopted until the next motion command and is not immediately active.

3.1.7 SET TARGET JERK

COMMAND CODE: 0xA2

DESCRIPTION: This is used to set the jerk parameter.

PARAMETERS (Master -> Slave): *Jerk* in the specified *unit system* ▸ 2 [🗋 22].

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful

EXAMPLE:

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x05	0xA2	0x00 0x00 0x7A 0x44	Set jerk to 1000 [mm/s^3]
S->M	0x03	0xA2	0x4F 0x4B	

MISCELLANEOUS: This value is retained after it has been written successfully once until the module is restarted or this value is changed again. The value is not adopted until the next motion command and is not immediately active.

3.1.8 SET TARGET CUR

COMMAND CODE: 0xA3

DESCRIPTION: This is used to set the current parameter.

PARAMETERS (Master -> Slave): *Current* in the specified *unit system* ▶ 2 [🗋 22].

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x05	0xA3	0xCD 0xCC 0x2C 0x40	Set the current to 2.7[A]
S->M	0x03	0xA3	0x4F 0x4B	

MISCELLANEOUS: This value is retained after it has been set successfully once until the module is restarted or this value is changed again. The value is adopted immediately and is also immediately active.

NOTE

This can be used to change the target specifications for the current during the movement. Thus it is possible with grippers to "re-grip" an already gripped object (increase or reduce gripping force).

NOTE

Only considered for movements if the controller structure allows it. Controller structure ▶ 5.3.2.3.15 [🗋 80] "CURRENT SPEED" does not allow subordinate current control.

3.1.9 SET TARGET POS

COMMAND CODE: 0xA6

DESCRIPTION: The "position" parameter for the next *"MOVE POS"* ▸ 3.1.2 [🗋 25] command is set with this.

PARAMETERS (Master -> Slave):

- *Position* in the specified *unit system* ▸ 2 [🗋 22]

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x05	0xA6	0x00 0x00 0xc8 0x43	Set position to 400.0 [°]
S->M	0x03	0xA6	0x4F 0x4B	OK

MISCELLANEOUS: This value is retained after it has been written successfully once until the module is restarted or this value is changed again. The value is not adopted until the next motion command and is not immediately active.

3.1.10 SET TARGET POS REL

COMMAND CODE: 0xA7

DESCRIPTION: The "relative position" parameter for the next *"MOVE POS REL"* ▸ 3.1.3 [🗋 27] command is set with this.

PARAMETERS (Master -> Slave):

- *Relative position* in the specified *unit system* ▸ 2 [🗋 22].

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x05	0xA7	0x00 0x00 0xc8 0x43	Set relative position to 400.0 [°]
S->M	0x03	0xA7	0x4F 0x4B	OK

MISCELLANEOUS: This value is retained after it has been written successfully once until the module is restarted or this value is changed again. The value is not adopted until the next motion command and is not immediately active.

3.1.11 CMD STOP

COMMAND CODE: 0x91

DESCRIPTION: The module is braked and stopped in the current position. In modules with an appropriately *configured holding brake* , the brake will be activated, otherwise, the module will be controlled actively.

PARAMETERS (Master -> Slave): None

ANSWER (Slave -> Master): "OK" (0x4F4B) if successful

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x01	0x91		
S->M	0x03	0x91	0x4F 0x4B	

MISCELLANEOUS: If the command cannot be executed correctly, then it is answered with an *error message* ▸ 4 [🗋 49]. For example, the module is to remain in active control, but cannot do this because a serious error is present.

NOTE

In modules without a brake or appropriately configured holding brake ▸ 5.3.2.7.3 [🗋 95], the maximum permissible current for control is limited to the nominal current ▸ 5.3.2.2.6 [🗋 72] to prevent overheating of the motor. This is why the module may "stall".

3.1.12 CMD FAST STOP

COMMAND CODE: 0x90

DESCRIPTIONIf a brake is fitted and appropriately *configured* ▸ 5.3.2.7.3 [🗋 95], it is activated immediately and the motor phases are short circuited. The motor is "de-energized".

PARAMETERS (Master -> Slave): None

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x01	0x90		
S->M	0x02	0x88	0xD9	Fast stop executed

ANSWER (Slave -> Master): Error message *"ERROR FAST STOP"* ▸ 4 [🗋 49] is triggered.

MISCELLANEOUS: Can only be reset again by *"CMD ACK"*▸ 4.1 [🗋 49].

⚠ WARNING

Risk of injury!

In modules without a brake, the module may "stall", since the motor is de-energized during a fast stop.

NOTE

This type of stopping leads to heavy mechanical wear on the brake.

3.2 Spontaneous messages

The module can report independently for certain events. A so-called "spontaneous message" is triggered. These messages are sent via the standard *data frame*. (D-Len, CmdCode, parameters). Spontaneous messages can be *deactivated* ▸ 3.2.6 [🗋 42].

NOTE

In Profibus, the MsgCount is not increased for these types of messages, since no data has been requested by the control unit.

3.2.1 CMD INFO

COMMAND CODE: 0x8A

EXAMPLE:

	D-Len	Cmd	Param	
S->M	0x02	0x8A	0x10	*"INFO TIMEOUT"* ▸ 4.2.9 [🗋 52]

DESCRIPTION: The module sends an information message.

REMARK: "Info messages" are also sent when errors are rectified and the module is restarted.

(▸ 4.2.1 [🗋 51]▸ 4.2.7 [🗋 52])

3.2.2 CMD MOVE BLOCKED

COMMAND CODE: 0x93

DESCRIPTION: The current motion command was interrupted. The motor was briefly blocked or is still blocked.

ANSWER (Slave -> Master): Current *position* in the set *unit system* ▸ 2 [🗋 22]

EXAMPLE:

	D-Len	Cmd	Param	
S->M	0x05	0x93	0xA4 0x70 0x9D 0x3F	Stop at position 1.23[mm]

MISCELLANEOUS: The motor is deemed as blocked if all of the following prerequisites are met:

- The motor turns at a speed below the *motion threshold* ▸ 5.3.2.5.9 [🗋 92]
- The target current is reached (+/-15%)
- The parameterized time in *waitMoveBlocked*▸ 5.3.2.5.13 [🗋 93] has expired.

Also see the *"Motion blocked" flag*

NOTE

When the blocking is detected, the set current will be reduced by root(2) in sine commutation. (Form factor for effective value to peak value in sinusoidal currents).

NOTE

Absolute reliable detection as to whether an object has been gripped is not possible using this.

3.2.3 CMD POS REACHED

COMMAND CODE: 0x94

DESCRIPTION: A positioning movement has reached the target position.

ANSWER (Slave -> Master): *Position* (current) in the set *unit system* ▸ 2 [🗋 22]

EXAMPLE:

	D-Len	Cmd	Param	
S->M	0x05	0x94	0xCD 0xCC 0x2C 0x40	Has reached position 2.7 [mm].

MISCELLANEOUS: The module has stopped. If the brake is present, it will be engaged depending on the *configuration* ▸ 5.3.2.7.3 [🗋 95].

3.2.4 CMD ERROR

COMMAND CODE: 0x88

DESCRIPTION:
A serious error has occurred which makes user intervention necessary. These errors must be acknowledged with *"CMD ACK"*▸ 4.1.4 [🗋 50]. The module is not ready for operation. The power to the motor is switched off and the brake is applied. These error messages are sent regularly every 15 seconds from the module to the control unit until the error is acknowledged.

- CAN: The first three bits in the identifier result in "0x3"
- Profibus: An extended diagnosis is generated

ANSWER (Slave -> Master): *Error code*▸ 4 [🗋 49]

EXAMPLE:

	D-Len	Cmd	Param	
S->M	0x02	0x88	0xDE	The error *"ERROR CURRENT"* has occurred.

⚠ WARNING

Risk of injury! In modules without a brake, the module may "stall", since the motor is de-energized in the event of a serious error.

3.2.5 GET STATE

COMMAND CODE: 0x95

DESCRIPTION: Provides the status of the module and some further information, if required. The module can independently send this status at regular intervals.

PARAMETERS (Master -> Slave):

- None
  Module provides the data once. This can be used to switch off the previously set cyclical sending of data.
- *Time* (4 bytes)
  The module independently transmits its status at the entered time interval (in the respective *unit system*).
- *Time* (4 bytes) *Data* (1 byte)
  The module independently transmits its status at the entered time interval (in the respective *unit system* 18).
  The "Mode" code is used to parameterize which data is also to be provided in addition to the status:
  Bit 1 (0x01): Position
  Bit 2 (0x02): Speed
  Bit 3 (0x04): Current

ANSWER (Slave -> Master): Optional data (observe the "Mode" code), then the status (2 bytes), which is structured as follows:

Referenced	Bit 1	0x01
Movement	Bit 2	0x02
Program sequence	Bit 3	0x04
Warning	Bit 4	0x08
Error	Bit 5	0x10
Brake	Bit 6	0x20
Motion blocked	Bit 7	0x40
Position reached	Bit 8	0x80

- Bit 1: The module is referenced.
- Bit 2: The module moves.
- Bit 3: The module is in program mode (an internal sequential program is active).
- Bit 4: There is a *warning*.
- Bit 5: There is an *error*.
- Bit 6: The brake is activated.
- Bit 7: A *movement was interrupted*.
- Bit 8: The *target position Position was reached*.
- In the case of an error, bits 9-16 also contain the *error code*.

EXAMPLE 1:

	D-Len	Cmd	Param	
M->S	0x06	0x95	0x00 0x00 0x80 0x3F 0x07	Status information is to be provided cyclically by the module every second. Aside from status bits, the position, speed, and current are also sent.
S->M	0x0F	0x95	0xD6 0xA3 0x70 0x41 0x56 0xC9 0x41 0x40 0x3C 0x41 0xEB 0x3E 0x03 0x00	cyclically every second Position: 0xD6..0x41, speed: 0x56..0x40, current: 0x3C..0x3E; Module is in motion and referenced (0x03); no error (0x00)

EXAMPLE 2:

	D-Len	Cmd	Param	
M->S	0x01	0x95		Query status information once. The parameters queried last will also be sent.
S->M	0x0F	0x95	0x53 0x63 0xB7 0x41 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x61 0xD9	Position: 0x53..0x41, speed: 0x00..0x00, current: 0x00..0x00; Module referenced, motion ended, brake actuated (0x61); fast stop actuated (0xD9)

EXAMPLE 3:

	D-Len	Cmd	Param	
M->S	0x06	0x95	0x00 0x00 0x00 0x00 0x01	Query status information once. Aside from status bits, the position is also sent.
S->M	0x07	0x95	0x00 0x00 0x00 0x00 0x20 0x00	Position: 0x00..0x00; module not referenced, brake actuated (0x20); no error (0x00)

MISCELLANEOUS: If you need to receive the position, speed, and current under CAN in one message, then the *fragmentation protocol* must be used. Under Profibus, all information is accommodated in one Profibus message. A "Data" data code set once is retained and thus does not need to be reset each time. If the module is switched on, then the data code is set to "0x07"; all possible status information is therefore transferred completely.

NOTE

If all parameters (position, speed, current) have been transferred, only the lower 8 bits of the status are displayed under Profibus. These now come to be in byte 14, where the status is always provided up-to-date Profibus specifically 15. The MsgCount that overwrites the upper 8 bits of the status word follows in byte 15.

NOTE

Under Profibus, the current position is transferred in bytes 10-13 as standard. If GET STATE is used to request all parameters, then the up-to-date position in bytes 10-13 is overwritten with the up-to-date value of the current. This is why it is recommended you set under Profibus the "Mode" byte to a maximum of "0x06" (speed and current will be transferred).

NOTE

Under Profibus, one should be careful when calling up "automatically". Under certain circumstances, it may be more favorable here to "poll" the data; in particular when the FREEZE mechanism is used.

3.2.6 CMD TOGGLE IMPULSE MESSAGE

COMMAND CODE: 0xE7

DESCRIPTION: This can be used to activate or deactivate spontaneous messages.

ANSWER (Slave -> Master): If the command is confirmed with "ON" (0x4F 0x4E), then spontaneous messages are active. If the command is confirmed with "OFF" (0x4F 0x46 0x46), then spontaneous messages are deactivated.

EXAMPLE:

	D-Len	Cmd	Param	
M->S	0x01	0xE7		
S->M	0x04	0xE7	0x4F 0x46 0x46	Spontaneous message deactivated.

MISCELLANEOUS: Spontaneous messages are always activated in the restart of the module.

3.3 Settings

3.3.1 CALIB CURRENT

COMMAND CODE: 0x8F

DESCRIPTION: The current sensors will be compared. (Offset has been set).

ANSWER (Slave -> Master): Offset values for phases A, B and C as Int16.

MISCELLANEOUS: The determined values are automatically accepted by the EEPROM ▸ 5.3.2.2.17 [🗋 76]. If the third current sensor is deactivated ▸ 5.3.2.2.16 [🗋 75], the offset for phase C is always set to "2048". If the comparison fails, a *ERROR CALIB CURENT* ▸ 4.2.44 [🗋 57] is triggered

NOTE

This command requires *Advanced rights* ▸ 5.3.3.33 [🗋 110].

NOTE

An uncalibrated module does not behave as expected, for example. dirty ride, current values does not make sense, movements are possible only in one direction.

3.3.2 SET CONFIG EXT

COMMAND CODE: 0x83

DESCRIPTION: Individual configuration parameters in the module are set and *saved permanently* ▶ 5.3 [🗋 62].

- Only one, single configuration parameter can be set specifically per command.
- The configuration parameters are identified via a 16-bit wide configuration code .
- The significance of a configuration code depends on the device connected. For more on this, see the relevant supplementary instructions.
- The value of a configuration parameter to be set can be transferred in various data formats, depending on the device (see definition of "data type" below).

PARAMETERS (Master -> Slave):

- Configuration code (2 bytes)
- Data type (1 byte)
- Configuration parameter (n bytes)

The "Configuration code" ▶ 5.3 [🗋 62] identifies the configuration parameter to be set. The definition of the data type, ▶ 7.1 [🗋 112].

The returned values of "Configuration code" ▶ 5.3 [🗋 62] match the requested values and thus allow an assignment of the response to the request.

MISCELLANEOUS: The supplied software tool can be used for making it easier to set the configuration parameters. (see "Schunk Motion Tool" software manual) verwendet werden.

NOTE

Some configuration parameters can only be changed if the module has stopped and the control is deactivated. Therefore, trigger fast stop ▶ 3.1.12 [🗋 35] beforehand, if necessary.

NOTE

Some configuration parameters cannot be changed. If "SET CONFIG EXT" is used to access such "Read-only" data, then a response is made with error code "INFO NO RIGHTS" 0x03.

3.3.3 GET CONFIG EXT

COMMAND CODE: 0x82

DESCRIPTION: Certain configuration parameters can be read out from the module.

- Only one, single configuration parameter can be queried specifically per command.
- The configuration parameters are identified via a 16-bit wide configuration code ▸ 5.3 [🗋 62].
- The value of a returned configuration parameter can be transferred in various *data formats* ▸ 3.3.2 [🗋 43], depending on the device.

PARAMETERS (Master -> Slave):

- Configuration code (2 bytes)

The "Configuration code" identifies the configuration parameter requested.

ANSWER (Slave -> Master):

- Configuration code (2 bytes)
- Data type (1 byte)
- Configuration parameter (n bytes)

The returned values of "Configuration code'' match the requested values and thus allow an assignment of the response to the request. The "Data type" is defined according to *SET CONFIG EXT*▸ 3.3.2 [🗋 43], ▸ 7.1 [🗋 112]. The n amount for bytes in the configuration parameter depends on the "Data type". With the "DT_STRING" data type, the number of the following ASCII characters is contained in the first byte of the configuration parameter.

MISCELLANEOUS: The supplied *software tool* can be used for making it easier to set the configuration parameters. (see "Schunk Motion Tool" software manual) verwendet werden.

3.4 Other

3.4.1 CMD REBOOT

COMMAND CODE: 0xE0

DESCRIPTION: The module is restarted.

PARAMETERS (Master -> Slave): None

ANSWER (Slave -> Master): The command is confirmed with "OK" (0x4F4B). The module then restarts and indicates *"INFO BOOT"* ▸ 4 [🗋 49] after a successful reboot.

3.4.2 CHANGE USER

COMMAND CODE: 0xE3

DESCRIPTION: The current user of the module is changed.

PARAMETERS (Master -> Slave):

- None
  User of *"User"* ▸ 5.3.3.33 [🗋 110] is set.
- *Password*
  or the respective user. ANSWER (Slave -> Master): The command is always confirmed with "OK (0x4F4B)". The current user (0 = User, 1 = Diag, 2 = Profi, 3 = Advanced) is attached (1 byte).

MISCELLANEOUS: If an incorrect password is entered, then "User" is always set. "User" is always active when restarting a module.

3.4.3 CMD GET DIO (0xE9)

COMMAND CODE: 0xE9
DESCRIPTION: Digital inputs/outputs can be read.
PARAMETERS (Master -> Slave):

3 types of requests are possible which differentiate in the number of transmitted parameters and in the type and number of requested digital inputs/outputs. Differing responses are also returned by the module depending on the type (1-3) of request.

Type 1: The current status of all digital inputs is read as a bit vector. (The outputs cannot be read with type 1 requests).:

PARAMETERS (Master -> Slave):

- None

ANSWER (Slave -> Master):

- *Bit vector* (4 bytes)

Type 2: A specific digital input or output is read. Which input or output is to be read is determined by the "Address" parameter. The "status" of the digital input or output addressed is transferred in bit 0 for the transmitted data byte in the response:

PARAMETERS (Master -> Slave):

- *Address* (4 bytes)

ANSWER (Slave -> Master):

- *Status* (1 byte)

Type 3: A sequential set of digital inputs or outputs is read. Which inputs or outputs are to be read is determined by the "Address" and "Length" parameters. Beginning with the input or output at "Address", as many inputs/outputs are read as the "Length" parameter specifies. In the response, the status of the digital inputs or outputs addressed is transferred as a bit vector in the transmitted "States" n data bytes. The number of transferred data bytes depends on the requested "Length" and is calculated as follows: "(Length + 7) / 8", where "/" designates the integer division (division without remainder). Thus, for the "Length" of 1-8, the response constitutes one byte; for the "Length" of 9-16, the response constitutes two bytes, etc.:

PARAMETERS (Master -> Slave):

- *Address* (4 bytes)
- Length (1 byte)

ANSWER (Slave -> Master):

- Condition (n bytes)

REMARK

- Type 1 requests can only read inputs (not outputs).
- Which addresses are valid for type 2 or 3 depends on the module. For more on this, see the respective supplementary instructions.

Example:

Type 1	**D-Len**	**Cmd**	**Param**	
M->S	0x01	0xE9		Type 1: Read all inputs
S->M	0x05	0xE9	0x01 0x02 0x04 0x08	Type 1: Inputs 1, 10, 19 and 28 are set, all other inputs are deleted

3.5 Fragmentation

3.5.1 FRAG ACK (0x87)

COMMAND CODE: 0x87

DESCRIPTION: Confirmation of a correctly processed fragment.

PARAMETERS (Master -> Slave): *D-Len code for received fragment* (UInt16) if the control unit confirms the fragment to the module.

ANSWER (Slave -> Master): *D-Len code for received fragment* (UInt16) if the module confirms the fragment to the control unit.

EXAMPLE: See *Selected examples* 130.

NOTE

This command is only required for Profibus 18) if a fragmentation of messages is necessary.

3.5.2 FRAG START (0x84)

COMMAND CODE 0x84

DESCRIPTION: Indicates in a fragmented message that it is the first fragment.

PARAMETERS (Master -> Slave): None

ANSWER (Slave -> Master): None

EXAMPLE: See *Selected examples* 130.

MISCELLANEOUS: It is written directly after the D-Len byte. However, it is not included in D-Len since it only serves as a marker.

3.5.3 FRAG MIDDLE (0x85)

COMMAND CODE: 0x85

DESCRIPTION: Indicates in a fragmented message that it is a middle fragment.

PARAMETERS (Master -> Slave): None

ANSWER (Slave -> Master): None

EXAMPLE: See *Selected examples* 130.

MISCELLANEOUS: It is written directly after the D-Len byte. However, it is not included in D-Len since it only serves as a marker.

3.5.4 FRAG END (0x86)

COMMAND CODE 0x86

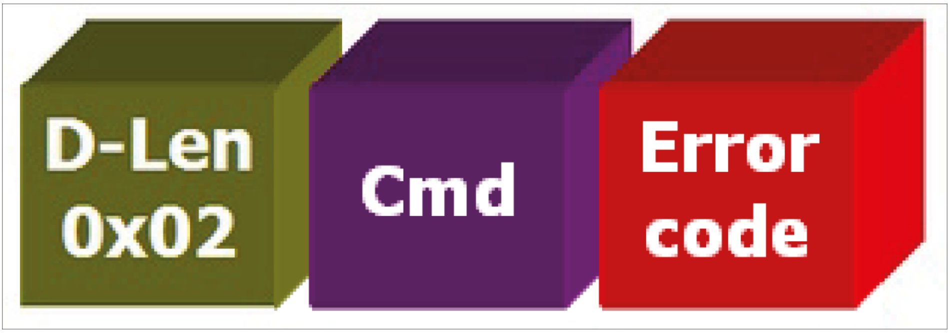
DESCRIPTION: Indicates in a fragmented message that it is the last fragment.

PARAMETERS (Master -> Slave): None

ANSWER (Slave -> Master): None

EXAMPLE: See *Selected examples* 130.
Other: It is written directly after the D-Len byte. However, it is not included in D-Len since it only serves as a marker.

4 Info and error messages



Error message

In the event of an error, D-Len always has exactly the value "2" in the data frame from the module to the control unit.

Either the error command is in the command byte or one of the following "error commands", ▸ 4.1 [🗋 49].

The parameter byte contains information on the cause of the error.

4.1 Troubleshooting

4.1.1 CMD ERROR

COMMAND CODE: 0x88

DESCRIPTION:
A serious error has occurred which makes user intervention necessary. These errors must be acknowledged with *"CMD ACK"*▸ 4.1.4 [🗋 50]. The module is not ready for operation. The power to the motor is switched off and the brake is applied. These error messages are sent regularly every 15 seconds from the module to the control unit until the error is acknowledged.

All *error codes* ▸ 4.2 [🗋 51] as an overview.

- CAN: The first three bits in the identifier result in "0x3"
- Profibus: An extended diagnosis is generated

⚠ WARNING

Risk of injury!

In modules without a brake, the module may "stall", since the motor is de-energized in the event of a serious error.

4.1.2 CMD WARNING

COMMAND CODE: 0x89

DESCRIPTION:

- Software limit ranges were transgressed. A fast stop is initially triggered at this point and must be acknowledged. The module is then partially ready for operation. (Traveling out of the software limit range is permitted). If the module is moved out of the software limit range, then the warning will be deleted.
- *Maximum temperature limit*▸ 5.3.2.1.13 [🗋 69] was exceeded. The module will shut down in 1[min] if the temperature is not lowered.

These error messages are regularly sent from the module to the control unit (every 30 seconds). All *error codes* ▸ 4.2 [🗋 51] as an overview.

- CAN: The first three bits in the identifier result in "0x3"
- Profibus: An extended diagnosis is generated

4.1.3 CMD INFO

COMMAND CODE: 0x8A

EXAMPLE:

	D-Len	Cmd	Param	
S->M	0x02	0x8A	0x10	*"INFO TIMEOUT"* ▸ 4.2.9 [🗋 52]

DESCRIPTION: The module sends an information message.

REMARK: "Info messages" are also sent when errors are rectified and the module is restarted.

(▸ 4.2.1 [🗋 51]▸ 4.2.7 [🗋 52])

4.1.4 CMD ACK

COMMAND CODE: 0x8B

DESCRIPTION: Acknowledgment of an active error message.

EXAMPLES:

OK

OK + no error

OK + no error + <error>

4.1.5 Detailed error information

Important information about the last occurring error can be queried for service via the *errorDetail* ▸ 5.3.3.5 [🗋 103], *errorLine* ▸ 5.3.3.4 [🗋 103] and *errorFile* ▸ 5.3.3.6 [🗋 103] parameters.

4.2 Information codes and error codes

4.2.1 INFO BOOT

The module has booted successfully.

Code:	**0x0001**

This is triggered after the module has been switched on and undergone a successful initialization. If this message occurs during operation, the logic voltage must be checked. It is also possible that there is a defective power drive.

4.2.2 INFO NO RIGHTS

The appropriate rights to execute the command are missing.

Code:	**0x03**

4.2.3 INFO UNKNOWN COMMAND

The sent command is not recognized.

Code:	**0x04**

4.2.4 INFO FAILED

The command has failed.

Code:	**0x05**

All of the parameters are correct, but the execution of the command is not possible at this time due to other reasons, e. g. the module is in emergency stop mode.

4.2.5 NOT REFERENCED

The module is not referenced and can therefore not execute the command.

Code:	**0x06**

A referencing process is necessary in order to carry out a positioning movement.

4.2.6 INFO SEARCH SINE VECTOR

Code:: 0x0007

A search is running for the space vector for the sine commutation. 60% of the maximum current is used for the phases.

Code:	**0x0007**

If the standstill commutation is not active and the commutation type is set to PMSM, a movement command for the space vector is carried out once after the power up and before the execution, ▸ 1.9 [🗋 21] and ▸ 5.3.2.2.3 [🗋 71].

4.2.7 INFO NO ERROR

There are no other error messages pending.

Code:	**0x0008**

If no error or warning is present, or if the module has moved out of the software limit stops, this info message is displayed.

4.2.8 INFO COMMUNICATION ERROR

An error has occurred in communication.

Code:	**0x09**

The connection of the communication cable and external influences on the communication cable must be checked.

4.2.9 INFO TIMEOUT

A timeout occurred during communication.

Code:	**0x10**

The data could not be sent, or more data was anticipated and was not received on time.

4.2.10 INFO WRONG DATA TYPE

The datatype does not match the parameter.

Code:	**0x12**

4.2.11 INFO RESTART

The module or control unit was restarted.

The module is not ready for operation. The message must be acknowledged; for acknowledgment the bits "Stop" and "Fast stop" must be set to "1", .

Code:	**0x13**

4.2.12 INFO CHECKSUM

The checksum is incorrect, the data is invalid.

Code:	**0x19**

4.2.13 INFO VALUE LIMIT MAX

The specified value exceeds the maximum permitted set value.

Code:	**0x1B**

4.2.14 INFO VALUE LIMIT MIN

The specified value falls below the minimum permitted set value.

Code:	**0x1C**

4.2.15 INFO MESSAGE LENGTH

The length command does not match the data received.

Code:	**0x1D**

4.2.16 INFO WRONG PARAMETER

One of the specified parameter values is outside of the permitted range.

Code:	**0x1E**

If a parameter value is found to be not permitted, all old parameter values will be retained, even if the other parameter values should lie in the valid range.

4.2.17 INFO UNKNOWN PARAMETER

The parameter requested is unknown.

Code:	**0x23**

4.2.18 ERROR FILE NOT FOUND

The file to be edited is not on the USB stick or the USB stick is defective.

Code:	**0x60**

4.2.19 ERROR FILE IS CORRUPT

The file to be edited on the USB stick is corrupted.

Code:	**0x61**

4.2.20 ERROR FILE TYPE WRONG

The data type is not correct.

Code:	**0x62**

- Create a new file.

4.2.21 ERROR FILE SYSTEM WRONG

The file system from the USB stick has an error.

Code:	**0x64**

- Check whether the USB drive is formatted with FAT16 or FAT32.

4.2.22 ERROR FILE READ

A reading error has occurred during the reading of the file.

Code:	**0x65**

4.2.23 ERROR FILE IS NOT CREATED

No file could be created.

Code:	**0x66**

- Check whether the USB stick is defective or write-protected.

4.2.24 ERROR FILE WRITE

A writing error has occurred during the writing of the file.

Code:	**0x67**

4.2.25 ERROR REBOOT

A parameter was written which is needed by a reboot.

Code:	**0x7C**

- Switch the module off and on.

4.2.26 ERROR MOTOR PHASE

One motor phase is not properly connected.

Code:	**0x7D**

4.2.27 ERROR WRONG RAMP TYPE

No valid motion profile has been selected for the positioning movement.

Code:	**0xC8**

4.2.28 ERROR WRONG DIRECTION

The module moves in the wrong direction for the check of the pseudo-absolute encoder

Code:	**0xD1**

- Check the sine pointer.

4.2.29 ERROR CONFIG MEMORY

The configuration range is incorrect. Data could not be written to EEPROM or EEPROM is defective.

Code:	**0xD2**

4.2.30 ERROR SOFT LOW

The module has exceeded the lower software limit range.

Code:	**0xD5**

- If necessary, acknowledge the error and move the module out of the software end stop with a movement command.

4.2.31 ERROR SOFT HIGH

The module has exceeded the upper software limit range.

Code:	**0xD6**

- If necessary, acknowledge the error and move the module out of the software end stop with a movement command.

4.2.32 ERROR SERVICE

An error has occurred that can only be remedied by SCHUNK.

Code:	**0xD8**

The detailed error information can be used by SCHUNK Service to localize the error precisely, ▸ 4.1.5 [🗋 50].

Contact SCHUNK's service and provide the following data:

- Module type
- Serial number of the module
- Description of how the error occurred

4.2.33 ERROR FAST STOP

A fast stop was triggered, .

The module is not ready for operation. No error condition was triggered. The message must be acknowledged.

Code:	**0xD9**

4.2.34 ERROR TOW

A towing error has occurred.

Code:	**0xDA**

- Reduce the load.
- Check the "towing error" parameter.

4.2.35 ERROR VPC3

The controller works incorrectly or is faulty.

Code:	**0xDB**

4.2.36 ERROR FRAGMENTATION

An error has occurred in the fragmentation protocol. Data packets have been lost.

Code:	**0xDC**

4.2.37 ERROR COMMUTATION

The motor cannot commutate.

Code:	**0xDD**

If this error occurs, the commutation mode is selected incorrectly. In case of block commutation the hall sensors are defective or not connected. With regard to sine commutation, there is an error in the position measuring system.

4.2.38 ERROR I2T

An I^2T error has occurred.

Code:	**0xDF**

- Reduce load of motor.

4.2.39 ERROR CURRENT

The maximum current was exceeded.

Code:	**0xDE**

- Reduce load of motor.

4.2.40 ERROR TOO FAST

The maximum speed was exceeded.

Code:	**0xE4**

4.2.41 ERROR POS SYSTEM

The position measuring system is not working correctly.

Code:	**0xE5**

- Check configuration of the module.

4.2.42 ERROR RESOLVER CHECK FAILED

A parameter for the resolver setting is incorrect.

Code:	**0xEB**

4.2.43 ERROR MATH

A mathematical error has occurred, e. g. division by zero.

Code:	**0xEC**

Usually, a configuration parameter is incorrect, resulting in the value range being exceeded. In most cases, a controller parameter is set incorrectly.

The detailed error information can be used by SCHUNK Service to localize the error precisely, ▸ 4.1.5 [🗋 50].

4.2.44 ERROR CALIB CURRENT

The measured values of the current sensors are beyond the tolerance limits.

Code:	**0xEE**

- Calibrate the module.
  - If the error occurs again, the current measuring is defective.

4.2.45 ERROR INITIALIZE

The module could not be properly initialized.

Code:	**0xE0**

- Check configuration parameters.

The detailed error information can be used by SCHUNK Service to localize the error precisely, ▸ 4.1.5 [🗋 50].

4.2.46 ERROR INTERNAL

An internal error has occurred.

Code:	**0xE1**

The firmware is in an undefined status.

Contact SCHUNK's service and provide the following data:

- Module type
- Serial number of the module
- Description of how the error occurred

4.2.47 ERROR CONNECTION TEMP LOW

The temperature of the communication board dropped below the minimal permissible level.

Code:	**0x6A**

- Warm up the module.

4.2.48 ERROR CONNECTION TEMP HIGH

The maximum permissible temperature of the communication board was exceeded.

Code:	**0x6B**

- Allow the module to cool down.
- Reduce the load.

4.2.49 ERROR MOTOR TEMP LOW

The temperature of the motor dropped below the minimal permissible level.

Code:	**0x6C**

- Warm up the module.

4.2.50 ERROR MOTOR TEMP HIGH

Code:: 0x6D

The temperature of the motor has exceeded the maximum permissible level.

Code:	**0x6D**

- Allow the module to cool down.
- Reduce the load.

4.2.51 ERROR TEMP LOW

The temperature value for the main board dropped below the minimal permissible temperature range.

Code:	**0x70**

- Warm up the module.

4.2.52 ERROR TEMP HIGH

The temperature value for the main board exceeded the maximum permissible temperature range.

Code:	**0x71**

- Allow the module to cool down.
- Reduce the load.

4.2.53 ERROR LOGIC LOW

The logic voltage is below the limit values.

Code:	**0x72**

- Check the logic voltage.

4.2.54 ERROR LOGIC HIGH

The logic voltage is above the limit values.

Code:	**0x73**

- Check the logic voltage.

4.2.55 ERROR MOTOR VOLTAGE LOW

The motor voltage is under the limit values.

Code:	**0x74**

- Check the motor voltage.
  - If necessary, the power supply unit for the motor voltage might be underdimensioned or the voltage supply cables to the module are not dimensioned correctly.

NOTE

MotorVoltageLow is a fatal error when the module is moved.

4.2.56 ERROR MOTOR VOLTAGE HIGH

The motor voltage is above the limit values.

Code:	**0x75**

NOTE

If this error occurs repeatedly, the module is disabled and can only be put into operation again by SCHUNK.

- Check the motor voltage.
  - If necessary, an external brake chopper might have to be used.

4.2.57 ERROR CABLE BREAK

Communication to the control was interrupted.

Code:	**0x76**

NOTE

The error is only displayed once communication has been re-established.

- Check communication cables
  - The communication cable is defective.

4.2.58 ERROR LIFE SIGN

Timeout of the internal module communication due to an internal error.

Code:	**0x7A**

- Module must be restarted.

Possible causes:

- External interference present on the logic voltage, observe the specifications for the basic data described in the product-specific installation and operating instruction.
  Additional possible solution: Separate supply of logic and power voltage via separate power supply units.
- Malfunction on the bus line, observe the specifications on bus physics of the PROFIBUS User Organisation e.V. (PNO).
  Possible solution: Reduce baud rate.

4.2.59 ERROR CUSTOM DEFINED

An error has occurred in a customer-defined function.

Code:	**0x7B**

4.2.60 ERROR OVERSHOOT

The module has overshot the target position.

Code:	**0x82**

- Increase the specified current.
  - The current needed for deceleration is too low.
- Check the controller parameters.

4.2.61 ERROR HARDWARE VERSION

The hardware of the different components do not match. One of the files saved on the USB stick can not be edited with the available hardware.

Code:	**0x83**

4.2.62 ERROR SOFTWARE VERSION

The software of the different components does not match. One of the files saved on the USB stick can not be edited with the available software version.

Code:	**0x84**

5 Configuration parameters

5.1 General

Access to the respective parameters is controlled in the module's firmware. There are various users here who are each protected by passwords. A change of the user is possible using the *"CHANGE USER"* ▸ 3.4.2 [🗋 45] command or by modifying the parameter *"user" (0x5610)* ▸ 5.3.3.33 [🗋 110]. The module recognizes the following users:

1. User:
   Standard user. This is active when the module is switched on. It has very limited parameterizing capacities for the module, but can fully operate it.
2. Profi:
   This can change all important parameters. Incorrect parameterization can result in unanticipated module behavior. However, the module cannot be destroyed.
3. Advanced:
   This can change all important parameters. Incorrect parameterization can lead to the destruction of the module!
4. Root:
   Access to all parameters. Incorrect parameterization can lead to the destruction of the module!
5. Schunk:
   parameter can only be changed in the production facility.
6. Disabled:
   parameter cannot be modified.

The following value ranges are used:

- MAX_BOOL -> 1
- MAX_INT8 -> 27
- MAX_INT16 -> 32767
- MAX_INT32 -> 2147483647
- MAX_UINT8 -> 255
- MAX_UINT16 -> 65535
- MAX_UINT32 -> 4294967295
- MAX_CHAR -> 0xFF
- MAX_ENUM -> 0xFFFF
- MAX_FLOAT -> 3.40282347E+38
- MIN_BOOL -> 0
- MIN INT8 -> (-MAX INT8 - 1)
- MIN INT16 -> (-MAX INT16 - 1)
- MIN INT32 -> (-MAX INT32 - 1)

- MIN_UINT8 -> 0
- MIN_UINT16 -> 0
- MIN_UINT32 -> 0
- MIN_CHAR -> 0
- MIN_ENUM -> 0
- MIN_FLOAT -> 1.17549435E-38

5.2 Parameterization with the help of a USB stick

5.2.1 Reading parameters

- Activate ECM/V6R USB flash drive (see hardware guide)
- Activate USB flash drive (see hardware guide)
- Idxx_???.sav file is created (xx = hexadecimal ID number of the module, ??? serial number)
- Open Idxx_???.sav file on the PC

5.2.2 Write parameters

- Create the IDxx Set.par file with the MTS Tool (xx = module ID number or "xx" if the parameter file on the module is to be written with different IDs)
- Save file on the USB flash drive (if need be, rename)
- Insert ECM/V6R USB flash drive (see hardware guide)
- Activate USB flash drive (see hardware guide)
- Activate parameter writing (see hardware guide)
- Idxx_???.log file is created (xx = ID number of the module, ??? serial number)
- Open Idxx_???.sav file on the PC (a protocol will be shown of which errors occurred during the parameterization)

5.3 Parameter

5.3.1 Cyclical

In this section, all data is listed which is cyclically exchanged with the module. In principle, access to this data is possible acyclically, but it is not recommended.

5.3.1.1 ControlWord

PARAMETER CODE: 0x7D01

Read - write access rights: *USER* - *USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type:(UINT16) MIN_UINT16 - MAX_UINT16

5.3.1.2 DigitalOut

PARAMETER CODE: 0x7D02

Read - write access rights: *USER* - *USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

5.3.1.3 DesiredPosition

PARAMETER CODE: 0x7D03

Read - write access rights: *USER* - *USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.4 DesiredVel

PARAMETER CODE: 0x7D04

Read - write access rights: *USER* - *USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.5 DesiredCur

PARAMETER CODE: 0x7D05

Read - write access rights: *USER - USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.6 DesiredAcc

PARAMETER CODE: 0x7D06

Read - write access rights: *USER - USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.7 DesiredJerk

PARAMETER CODE: 0x7D07

Read - write access rights: *USER - USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.8 DesiredPrePosition

PARAMETER CODE: 0x7D08

Read - write access rights: *USER - USER*

DESCRIPTION: can only be accessed with Schunk Drive Protocol (SDP) for
Profibus/PROFINET.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.9 StatusWord

PARAMETER CODE: 0x7D09

Read - write access rights: *USER - Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.1.10 ActiveCopy

PARAMETER CODE: 0x7D0A

Read - write access rights: *USER* - *Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

5.3.1.11 DigitalIn

PARAMETER CODE: 0x7D0B

Read - write access rights: *USER* - *Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

5.3.1.12 ActPosition

PARAMETER CODE: 0x7D0C

Read - write access rights: *USER* - *Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.13 ActVel

PARAMETER CODE: 0x7D0D

Read - write access rights: *USER* - *Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.14 ActCur

PARAMETER CODE: 0x7D0E

Read - write access rights: *USER* - *Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.1.15 ErrorCode

PARAMETER CODE: 0x7D0F

Read - write access rights: *USER - Disabled*

DESCRIPTION: can only be accessed with *Schunk Drive Protocol (SDP)*.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

5.3.2 Permanent

All parameters are listed here that are permanently saved in the internal EEPROM of the module.

Access is only possible via:

- MTS
- USB stick ▸ 5.2 [🗋 62]
- ProfiBus DPV1 (asynchronous data exchange)

5.3.2.1 Device

Serial number

PARAMETER CODE: 0x7D73

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the module

Data type: (UINT32) 0 - MAX_UINT32

Actual gripper

PARAMETER CODE: 0x7D74

Read - write access rights:

USER - PROFI

DESCRIPTION: Shows whether the module is a gripper.

Data type: (BOOL) MIN-BOOL - MAX_BOOL

Invert direction of rotation of motor

PARAMETER CODE: 0x7D76

Read - write access rights:

USER - PROFI

DESCRIPTION: The direction of rotation of the motor is defined.

NOTE

Incorrect configuration may lead to unexpected results. (The module rotates at an unexpectedly high speed!)

Data type: (BOOL) false - true

Invert position measurement

PARAMETER CODE: 0x7D77

Read - write access rights:

USER - PROFI

DESCRIPTION: The direction of measurement for the position measuring system is defined. Interchanged A and B encoder tracks can be exchanged using software.

NOTE

Incorrect configuration may lead to expected results. (The module rotates at an unexpectedly high speed!)

Data type: (BOOL) false - true

NOTE

If the directions of rotation of the motor and the position measuring system are inverted at the same time, then a "right-turning" module can be configured from a "left-turning" one, or a "positive" closing gripper can be configured from a "positive" opening gripper.

Endless

PARAMETER CODE: 0x7D78

Read - write access rights:

USER - PROFI

DESCRIPTION: The axis can turn endlessly.

Data type: (BOOL) false - true

NOTE

This option is not recommended for grippers!

Digital Inputs

PARAMETER CODE: 0x7D79

Read - write access rights:

USER - PROFI

DESCRIPTION: Using digital inputs.

Data type: DIGITAL_IN_NORMAL - MAX_DIGITAL_IN_TYPE, ▶ 7.1 [🗋 112]

Digital outputs

PARAMETER CODE: 0x7D7A

Read - write access rights:

USER - PROFI

DESCRIPTION: Using digital outputs.

Data type: DIGITAL_OUT_NORMAL - MAX_DIGITAL_OUT_TYPE, ▶ 7.1 [🗋 112]

Min. position

PARAMETER CODE: 0x7D7B

Read - write access rights:

USER - PROFI

DESCRIPTION: Minimum position allowed (software limit range). Will be ignored if *"Endless"* ▶ 5.3.2.1.5 [🗋 67] is set. Is used for *Referencing* ▶ 5.3.2.4.1 [🗋 82] with "stroke control".

Data type: (FLOAT) -MAX_FLOAT - maxPosition

Max. position

PARAMETER CODE: 0x7D7C

Read - write access rights:

USER - PROFI

DESCRIPTION: Maximum position allowed (software limit range). Will be ignored if *"Endless"*▶ 5.3.2.1.5 [🗋 67] is set. Is used for *Referencing* ▶ 5.3.2.4.1 [🗋 82] with "stroke control".

Data type: (FLOAT) minPosition - MAX_FLOAT

Min. motor temperature

PARAMETER CODE: 0x7D7F

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Minimum permissible working temperature of the motor (in case motor temperature sensor is connected). If the temperature value falls below the working temperature, there will be an error message
▸ 4.2.49 [🗋 58].

Data type: (FLOAT) -MAX_FLOAT - maxTempMotor

Max. motor temperature

PARAMETER CODE: 0x7D80

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible working temperature of the motor (in case motor temperature sensor is connected). If the temperature value exceeds the working temperature, there will be an error message
▸ 4.2.50 [🗋 58].

Data type: (FLOAT) -MAX_FLOAT - maxTempBoard2

Min. main board temperature

PARAMETER CODE: 0x7D7D

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Minimum permissible working temperature. If the temperature value falls below the working temperature, there will be an error message ▸ 4.2.51 [🗋 58].

Data type: (FLOAT) -MAX_FLOAT - maxTempBoard1

Max. main board temperature

PARAMETER CODE: 0x7D7E

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible working temperature. If the temperature value exceeds the working temperature, there will be an warning. There will be an error message ▸ 4.2.52 [🗋 58] if the temperature does not drop within 1[min].

Data type: (FLOAT) minTemoBoard1 - MAX_FLOAT

Min. OPT temperature Comm.

PARAMETER CODE: 0x7D81

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Minimum permissible working temperature of the second temperature sensor. If the temperature value falls below the working temperature, there will be an error message ▸ 4.2.47 [🗋 57].

Data type: (FLOAT) -MAX_FLOAT - maxTempBoard2

Max. OPT temperature Comm.

PARAMETER CODE: 0x7D82

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible working temperature of the second temperature sensor. If the temperature value exceeds the working temperature, there will be an error message ▸ 4.2.48 [🗋 57].

Data type: (FLOAT) minTempBoard2 - MAX_FLOAT

5.3.2.2 Motor

Serial number

PARAMETER CODE: 0x7D1E

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the installed motor.

Can only be written when the motor is de-energized.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

Voltage

PARAMETER CODE: 0x7D1F

Read - write access rights:

USER - ROOT

DESCRIPTION: Rated voltage of the motor. This can be used to calculate both the derived brake control PWM (duty cycle) as well as the maximum permissible PWM for test purposes. This value is also needed for automatic controller configuration.

Can only be written when the motor is de-energized.

Data type: (UINT8) 24 or 48

CAUTION

An incorrect entry can lead to the destruction of the electronics.

Type

PARAMETER CODE: 0x7D20

Read - write access rights:

USER - ADVANCED

DESCRIPTION: The motor type can be selected.

- DC (0x00): Brush-type DC motor
- BLDC (0x01): Electronically commutated brushless DC motor with block commutation.
- PMSM (0x02): Electronically commutated brushless DC motor with sine commutation.

Can only be written when the motor is de-energized.

Data type: (ENUM) ▸ 7.1 [🗋 112] MOTOR_DC - MAX_MOTORTYPE -1

CAUTION

An incorrect entry can lead to the destruction of the electronics.

I2T

PARAMETER CODE: 0x7D21

Read - write access rights:

USER - ADVANCED

DESCRIPTION: I^2T monitoring can be activated. An I^2T *error* ▸ 4.2.38 [🗋 56] will be triggered if the load is too high.
With I^2T monitoring, it is assumed that the maximum current may be present for three seconds (corresponds to 100%). If a value of < 100% is entered, the time will be shortened accordingly (I^2T monitoring is triggered earlier).

Can only be written when the motor is de-energized.

Data type: (UINT8) 10 - 100

Max. current

PARAMETER CODE: 0x7D22

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible current that may briefly flow through the motor's phases. If this current is exceeded for a long period of time (several ms), then a fast stop will be triggered with the error message ERROR CURRENT ▸ 4.2.39 [🗋 56].

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - maxMeasureCurrent

CAUTION

An incorrect entry can lead to the destruction of the electronics.

Nom. Current

PARAMETER CODE: 0x7D23

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Current that may flow permanently through the motor's phases. If this is exceeded for a long period of time, then an I^2T *error* ▸ 4.2.38 [🗋 56] will be triggered.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - maxCur

CAUTION

An incorrect entry can lead to the destruction of the electronics.

Max. speed

PARAMETER CODE: 0x7D24

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible speed of the system (on the drive side).

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

Max. acceleration

PARAMETER CODE: 0x7D25

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible acceleration of the system (output side).

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

Max. jerk

PARAMETER CODE: 0x7D26

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permissible jerk of the system (output side). The jerk is the temporal alteration of acceleration. This parameter is only evaluated if *a positioning movement with jerk limitation* ▸ 5.3.2.5.11 [🗋 92] is executed.

Data type: (FLOAT) 0 - MAX_FLOAT

Pole pairings

PARAMETER CODE: 0x7D27

Read - write access rights:

USER - ROOT

DESCRIPTION: Electric pole pairings of the motor. Only required for brushless DC motors. Has effects on the calculation of speeds, positions, and commutation patterns.

Can only be written when the motor is de-energized.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Terminal resistance

PARAMETER CODE: 0x7D28

Read - write access rights:

USER - ROOT

DESCRIPTION: Terminal resistance is needed both for test functions to limit maximum currents as well as for automatic controller configuration.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

CAUTION

An incorrect entry can lead to the destruction of the electronics.

Inductance

PARAMETER CODE: 0x7D29

Read - write access rights:

USER - ROOT

DESCRIPTION: Terminal inductance is needed for automatic controller configuration.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

CAUTION

An incorrect entry can lead to the destruction of the electronics.

Motor constant

PARAMETER CODE: 0x7D2A

Read - write access rights:

USER - ROOT

DESCRIPTION: The motor constant is needed for the system settings.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

Commutation table

PARAMETER CODE: 0x7D2B

Read - write access rights:

USER - ROOT

DESCRIPTION: The Hall sensor table valid for the unit is set here for block commutation using Hall sensors. If the entry is incorrect, the motor will not move at all or only generates very little torque.

Can only be written when the motor is de-energized.

Data type: (UINT8) 0 - 5

Metering current range

PARAMETER CODE: 0x7D2C

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Maximum metering current range for the current sensor used internally.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

Max. measurement discrepancies

PARAMETER CODE: 0x7D2D

Read - write access rights:

USER - ROOT

DESCRIPTION: 3. Current sensor needed. Maximum permitted deviation according to A + B + C = 0. If the deviation is greater than the set value, a *ERROR MOTOR PHASE* ▸ 4.2.26 [🗋 54] will be produced.

Can only be written when the motor is de-energized.

NOTE

The value "-1" deactivates the third current sensor (not supported by all hardware versions). => A motor phase break or motor phase short circuit is not recognized.

NOTE

If the third current sensor has been activated, a calibration ▸ 3.3.1 [🗋 42] of the current sensors is needed.

Data type: (FLOAT) 0 - MAX_FLOAT

Offset phase A

PARAMETER CODE: 0x7D2E

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Can be written via *Calibration wizard* ▸ 3.3.1 [🗋 42] with advanced rights. Zero point alignment for the first current sensor. This value should vary in a range between 1700 - 2200. Otherwise it is thought that the hardware may be defective.

Can only be written when the motor is de-energized.

Data type: (UINT16) 1500 - 2600

NOTE

An incorrect value may lead to unpredictable behavior in the drive (only travels in one direction, jerks badly, overspeed).

Offset phase B

PARAMETER CODE: 0x7D2F

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Can be written via calibration wizard with advanced rights. : Zero point alignment for the second current sensor. This value should vary in a range between 1700 - 2200. Otherwise it is thought that the hardware may be defective.

Can only be written when the motor is de-energized.

Data type: (UINT16) 1500 - 2600

NOTE

An incorrect value may lead to unpredictable behavior in the drive (only travels in one direction, jerks badly, overspeed).

Offset phase C

PARAMETER CODE: 0x7D30

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Can be written via *Calibration wizard* ▶ 3.3.1 [🗋 42] with advanced rights. : Zero point alignment for the third current sensor. This value should vary in a range between 1700 - 2200. Otherwise it is thought that the hardware may be defective.

Can only be written when the motor is de-energized.

If the hardware does not support third current sensors, then this value is always written as 2048.

Data type: (UINT16) 1500 - 2600

NOTE

An incorrect value may lead to unpredictable behavior in the drive (only travels in one direction, jerks badly, overspeed).

5.3.2.3 Controller

KR current

PARAMETER CODE: 0x7D4B

Read - write access rights:

USER - PROFI

DESCRIPTION: Proportional part of current controller. With *current limiting control* ▶ 5.3.2.3.15 [🗋 80], this is the proportional term for the current limiting control

Data type: (FLOAT) 0 - MAX_FLOAT

TN current

PARAMETER CODE: 0x7D4C

Read - write access rights:

USER - PROFI

DESCRIPTION: Integral term of the current controller. This is not required for *current limiting control* ▶ 5.3.2.3.15 [🗋 80].

Data type: (FLOAT) 0 - MAX_FLOAT

TD current

PARAMETER CODE: 0x7D4D

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Differential part of the current controller.

Data type: (FLOAT) 0 - MAX_FLOAT

KC current

PARAMETER CODE: 0x7D4E

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Correction factor of the current controller for integral part (anti-windup).

Data type: (FLOAT) 0 - MAX_FLOAT

KR speed

PARAMETER CODE: 0x7D4F

Read - write access rights:

USER - PROFI

DESCRIPTION: Proportional term of speed controller

Data type: (FLOAT) 0 - MAX_FLOAT

TN speed

PARAMETER CODE: 0x7D50

Read - write access rights:

USER - PROFI

DESCRIPTION: Integral term of speed controller

Data type: (FLOAT) 0 - MAX_FLOAT

TD speed

PARAMETER CODE: 0x7D51

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Differential part of the speed controller.

Data type: (FLOAT) 0 - MAX_FLOAT

KC speed

PARAMETER CODE: 0x7D52

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Correction factor of the current controller for integral part (anti-windup).

Data type: (FLOAT) 0 - MAX_FLOAT

KR position

PARAMETER CODE: 0x7D53

Read - write access rights:

USER - PROFI

DESCRIPTION: Proportional part of position controller.

Data type: (FLOAT) 0 - MAX_FLOAT

TN position

PARAMETER CODE: 0x7D54

Read - write access rights:

USER - PROFI

DESCRIPTION: Integral part of the position controller.

Data type: (FLOAT) 0 - MAX_FLOAT

TD position

PARAMETER CODE: 0x7D55

Read - write access rights:

USER - PROFI

DESCRIPTION: Differential part of the position controller.

Data type: (FLOAT) 0 - MAX_FLOAT

KC position

PARAMETER CODE: 0x7D56

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Correction factor of the current controller for integral part (anti-windup).

Data type: (FLOAT) 0 - MAX_FLOAT

Position deviation

PARAMETER CODE: 0x7D59

Read - write access rights:

USER - PROFI

DESCRIPTION: Position window in which position control is canceled (depending on *brake configuration* ▶ 5.3.2.7.3 [🗋 95]), control is continued or the brake is activated or Position reached is signaled.

Data type: (FLOAT) 0 - MAX_FLOAT

Max. overshoot

PARAMETER CODE: 0x7D5A

Read - write access rights:

USER - ADVANCED

DESCRIPTION: If the module moves via this position window during a positioning movement, then the error message *ERROR OVERSHOOT* ▶ 4.2.60 [🗋 60] will be generated. This value must be set greater than the maximum permissible *position variation* ▶ 5.3.2.3.13 [🗋 80].

Data type: (FLOAT) 0 - MAX_FLOAT

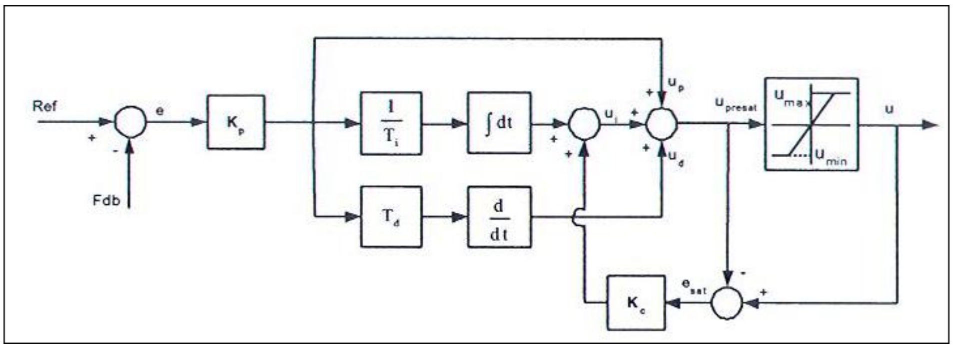
Structure

PARAMETER CODE: 0x7D5B

Read - write access rights:

PROFI - PROFI

DESCRIPTION:



Controllers, structure

All controllers are done as PID controllers with anti-windup functionality. The complete parameter set is only to be accessed via the root rights in SCHUNK mode.

The following parameters can be set:

- KR: proportional part of the respective controller
- TN: integral part of the respective controller
- TD: differential part of the respective controller

- KC: correction factor for the integral part
- Current - Speed (0x00)
  Current control and speed control operate independently of each other.
- Cascade (0x01)
  Position, speed, and current controllers are cascade connected => Current controlled (set current is not to be exceeded) positioning or speed movements are possible (e.g. no pre-positioning required for a gripping process). In this mode, the specified current ("SET TARGET CURRENT" ▸ 3.1.8 [🗋 32]) is not exceeded in all motion types.
- Speed with current limiting (0x02)
  Current control is not active. The specified current ("SET TARGET CURRENT" ▸ 3.1.8 [🗋 32]) is limited for speed or position movements. In contrast to the cascade, the current is not controlled, but limited (current limiting control).
- Speed with PWM limiting (0x02)
  Current control is not active. In speed or positioning movements, the PWM's duty cycle is limited. The ratio of current to duty cycle is calculated using the *terminal resistance* ▸ 5.3.2.2.11 [🗋 74] of the motor.

NOTE

Since the PWM's duty cycle is directly limited (voltage limitation), it is possible that the drive no longer reaches its full speed, which is why positioning movements could possibly take considerably longer than anticipated.

NOTE

If the controller structure is changed, the controller parameters might have to be adapted.

Data type: (ENUM) ▸ 7.1 [🗋 112] CTRL_CURRENT_SPEED - MAX_CTRL_MODE-1

5.3.2.4 Referencing

Type

PARAMETER CODE: 0x7D41

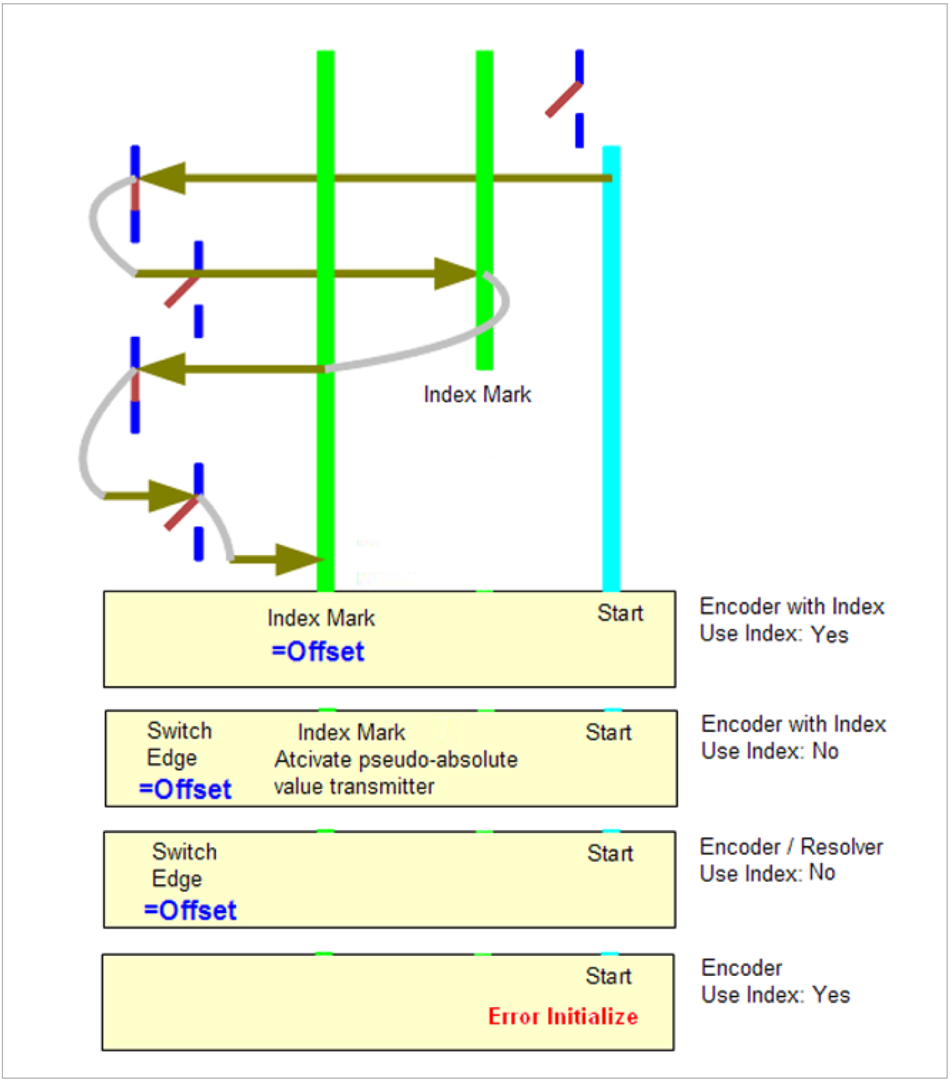
Read - write access rights:

USER - PROFI

DESCRIPTION: eReferenceType
The type of referencing can be defined. Observe the "Positioning type" ▸ 5.3.2.5.2 [🗋 89] section when using the encoder with index track.

Internal switch, left (0x00) / right (0x01)
The internal reference switch is used for referencing. The direction of motion when the reference switch is active is determined by direction "left" or "right".

External switch IN1, left (0x02) / right (0x03)
An external reference switch (IN1) is used for referencing. The direction of motion when the reference switch is active is determined by direction "left" or "right".



Referencing with switch

NOTE

Ensure that the proximity switch remains in either status for at least 200 ms before switching. (Adjust referencing speed, adjust control cams).

NOTE

It is recommended to perform a "basic referencing" in the application after installation. It might also be necessary to perform a "basic referencing" if the position or load is changed during referencing. Otherwise an *ERROR REFERENCED* ▸ 4.2.5 [🗋 51] might occur often

Speed, left (0x04) / right (0x05)
A speed movement is executed for referencing. If the module moves to a fixed stop, then this will be recognized as a reference point. The direction of rotation is defined via "left" or "right".

NOTE

This type of referencing is only recommended if a fixed end stop is available.

Speed with stroke monitoring, left (0x06) / right (0x07)
In addition to the procedure described above, a movement is made to the opposite fixed stop after the first fixed stop has been approached. The traveled distance must be greater than the difference in the *software limit ranges* ▸ 5.3.2.1.8 [🗋 68] => Referencing successful

NOTE

This type of referencing is only recommended if a fixed end stop is available.

Current, left (0x08) / right (0x09)
A current movement is executed. The current is increased until the module moves. If the current exceeds the *max. reference current* ▸ 5.3.2.4.5 [🗋 87], then it is assumed that a fixed stop has been reached which is recognized as a reference point.

NOTE

This type of referencing is only recommended if a fixed end stop is available.

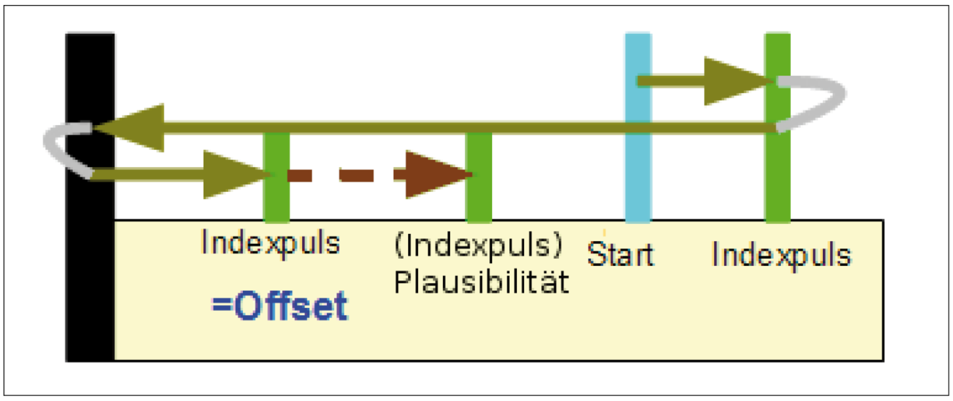
NOTE

Jamming, sluggishness in the mechanical system, or a "forgotten" workpiece can also lead to the rated current being exceeded. This would then also be interpreted as a fixed end stop, even if there are none present.

Current with stroke monitoring, left (0x0A) / right (0x0B)
In addition to the procedure described above, a movement is made to the opposite fixed end stop after the first fixed end stop has been approached. The traveled distance must be greater than the difference in the *software stops* ▸ 5.3.2.1.8 [🗋 68] => Referencing successful

NOTE

This type of referencing is only recommended if a fixed end stop is available.



Referencing up to stop with activated index track

None (0x0C)
After the *CMD REFERENCE* command has been sent, the current position will be seen as the reference position.

Data type: (ENUM) ▸ 7.1 [🗋 112] REF_SWITCH_INTERN_LEFT - MAX_REFTYPE-1

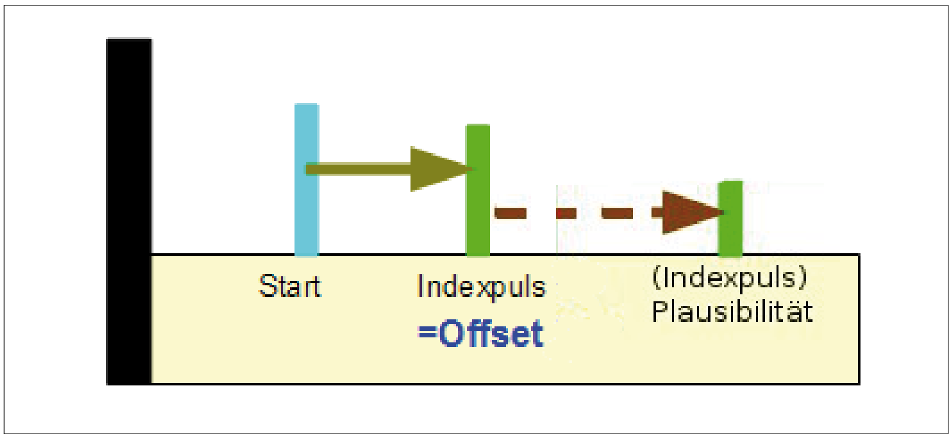
Usage index

PARAMETER CODE: 0x7D42

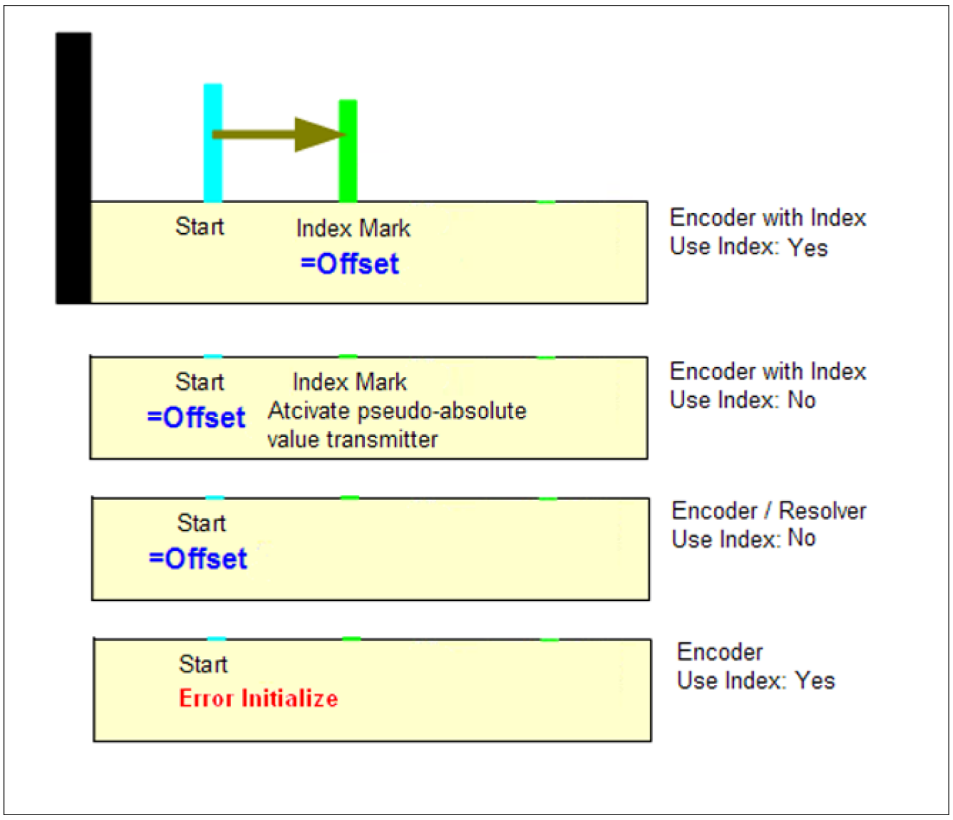
Read - write access rights:

USER - PROFI

DESCRIPTION: Specifies whether the index track of an encoder is to be evaluated during referencing.



Referencing of "none" with activated index track



Referencing of "none"

NOTE

Should be referenced with index pulse. If the index pulse is in a poor position, it can happen that the positions differ by one motor rotation each after repeated referencing. Remedy: Shift the reference mark a bit. This applies to all reference marks except "internal switch" and "external switch".

Data type: (BOOL) false - true

Distance to the index

PARAMETER CODE: 0x7D43

Read - write access rights:

USER - PROFI

DESCRIPTION: If the parameter "Distance to index" is set to "0", then the distance from the reference event (switching detected) to the index pulse is measured and saved during the reference movement. For subsequent reference movements, this is measured again and compared with the saved value. If the values are within an internally set *tolerance range* ▶ 5.3.2.4.10 [🗋 88], then reference can be successfully completed. Furthermore, an index pulse at an "unfavorable" point (index pulse shortly before or after switching) can be "corrected".

NOTE

If the distance between index pulse and reference event exceeded the permissible tolerance▶ 5.3.2.4.10 [🗋 88], then referencing will be aborted and "ERROR REFERENCED" ▶ 4.2.5 [🗋 51] displayed. It is necessary to measure the distance from the reference event to the index pulse again. To do this, the parameter "Distance to index" ▶ 5.3.2.4.3 [🗋 86] must be set to "0".

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

Approach 0 after referencing

PARAMETER CODE: 0x7D44

Read - write access rights:

USER - PROFI

DESCRIPTION: The position "0.0" is approached after successful referencing.

Data type: (BOOL) false - true

Max. reference current

PARAMETER CODE: 0x7D45

Read - write access rights:

USER - PROFI

DESCRIPTION: Specified current in [%] of the *motor's rated current* ▸ 5.3.2.4.5 [🗋 87]. The reference current does not exceed the specified value.

Data type: (UINT8) 0 - 200

velocity

PARAMETER CODE: 0x7D46

Read - write access rights:

USER - PROFI

DESCRIPTION: Specification of speed for reference movements with internal or external reference switch, and speed reference runs.

Data type: (FLOAT) 0 - maxVel

acceleration

PARAMETER CODE: 0x7D47

Read - write access rights:

USER - PROFI

DESCRIPTION: Specification of acceleration for reference movements with internal or external reference switch, and speed reference runs.

Data type: (FLOAT) 0 - maxAcc

Offset

PARAMETER CODE: 0x7D48

Read - write access rights:

USER - PROFI

DESCRIPTION: Position offset after successful referencing (zero offset)

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

Timeout

PARAMETER CODE: 0x7D49

Read - write access rights:

USER - PROFI

DESCRIPTION: Maximum time that the reference movement may take. If the time is exceeded, the drive will be de-energized and an *error message*
▶ 4.2.5 [🗋 51] comes

Data type: (FLOAT) 0 - MAX_FLOAT

Max. distance switch

PARAMETER CODE: 0x7D4A

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permitted distance from the reference event (switch flack detected) to the index pulse. If one encoder tick is specified. Also see ▶ 5.3.2.4.3 [🗋 86].

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.2.5 Positioning

Serial number

PARAMETER CODE: 0x7D5F

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the position measuring system.

Can only be written when the motor is de-energized.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

Type

PARAMETER CODE: 0x7D60

Read - write access rights:

USER - ADVANCED

DESCRIPTION: The type of measuring system is defined.

Can only be written when the motor is de-energized.

A restart of the module is required.

- Encoder
  Measuring system encoder without index track.
- Encoder index
  Encoder with index track. For the *reference movements* ▸ 5.3.2.4.1 [🗋 82], the index track is expanded depending on the *configuration* ▸ 5.3.2.4.2 [🗋 85]. To ensure that an index track can be found at the correct location, it is possible that with some referencing types the drive moves back and forth several times with short movements or makes small movements in the "wrong" direction. The modules also move in referencing type "None", because a search is made for the next index pulse. Also see *Pseudo-absolute encoder* ▸ 1.8 [🗋 19].
- Resolver
  Resolver system in which the excitation current can be set. Observe *Amplitude* ▸ 5.3.2.5.5 [🗋 90] and *Frequency* ▸ 5.3.2.5.6 [🗋 91].

NOTE

In motors with a resolver, "rattling" may occur in *"Positioning ramp type"* ▸ 5.3.2.5.11 [🗋 92] "jerk-limited". Another "Positioning ramp type" ▸ 5.3.2.5.11 [🗋 92] has to be selected in that case.

NOTE

A restart of the module is required.

- Differential encoder
  Differential encoder without index track

- Index encoder, differential
  Differential encoder with index track. For the reference movements, the index track is expanded depending on the *configuration* ▸ 5.3.2.4.2 [🗋 85]. To ensure that an index track can be found at the correct location, it is possible that with some referencing types the drive moves back and forth several times with short movements or makes small movements in the "wrong" direction. The modules also move in referencing type "None", because a search is made for the next index pulse. Also see ▸ 1.8 [🗋 19]

Data type: *(ENUM)* ▸ 7.1 [🗋 112] POS_SYSTEM_ENCODER - MAX_POS_SYSTEM_TYPE-1

Installation position

PARAMETER CODE: 0x7D61

Read - write access rights:

USER - ADVANCED

DESCRIPTION: The installation position of the position measuring system.

- On the drive side: The position measuring system is mounted directly on the drive end.
- Output side: The position measuring system is mounted directly on the output end.
- Between gear ratios: The position measuring system is mounted in the middle of the gear.

Data type: *(ENUM)* ▸ 7.1 [🗋 112] MOUNT_INPUT_SIDE - MAX_POS_MOUNT-1

Ticks per revolution

PARAMETER CODE: 0x7D62

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Ticks per revolution (4x evaluation)

Can only be written when the motor is de-energized.

Data type: (UINT16) 512 - MAX_UINT16

Exciter amplitude

PARAMETER CODE: 0x7D63

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Amplitude of the input voltage at the exciter coil [%]. Measurement must be determined. The output voltage at the receiver coils must not get into saturation. Data:

Data type: (UINT8) 0 - 100

Exciter frequency

PARAMETER CODE: 0x7D64

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Frequency of the voltage at the exciter coil in [kHz]. Permitted values: 8 [kHz], 4 [kHz], 2 [kHz], 1 [kHz].

Can only be written when the motor is de-energized.

Data type: *(ENUM)* ▸ 7.1 [🗋 112] RESOLVER_FRQ_1kHz - MAX_RESOLVER_FRQ-1

ADC offset

PARAMETER CODE: 0x7D65

Read - write access rights:

USER - ADVANCED

DESCRIPTION: ADC offset by the input signal for "centering". Only required for cos/sin measuring systems and resolvers.

Can only be written when the motor is de-energized.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

Offset

PARAMETER CODE: 0x7D68

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Twisting the position measuring system opposite the motor phases. Configured in the preset *unit system* ▸ 2 [🗋 22]. Under certain circumstances, this value is determined automatically. Also see standstill commutation ▸ 1.9 [🗋 21]. If this value is set at "0", a new search is made for the *"sine pointer"*. The *referencing* is deleted in the process.

Can only be written when the motor is de-energized.

Data type: (FLOAT) - MAX_FLOAT - MAX_FLOAT

NOTE

The module should be able to move freely in all directions for the pointer search. The module is moved in a jerking fashion up to two motor revolutions. Communication with the module is not possible during this time. This process may require up to 30 seconds.

Motion threshold

PARAMETER CODE: 0x7D69

Read - write access rights:

USER - PROFI

DESCRIPTION: The value in [%] of the *maximum speed* ▸ 5.3.2.2.7 [🗋 73]. If this value is not reached, then the module is treated as stationary. See *Motion tool, Schunk status display* (see "Schunk Motion Tool" software manual).

Can only be written when the motor is de-energized.

Data type: (FLOAT) 1 - 100

Position waiting time reached

PARAMETER CODE: 0x7D6A

Read - write access rights:

USER - PROFI

DESCRIPTIONOnly valid for Schunk Drive Protocol (SDP) for Profibus/PROFINET. For this, the resetting of the *"Position reached"* flag can be delayed for a defined time in [sec.]. It is recommended to set this value somewhat larger than the PLC cycle.

Data type: (FLOAT) 0 - MAX_FLOAT

Position ramp

PARAMETER CODE: 0x7D6B

Read - write access rights:

USER - PROFI

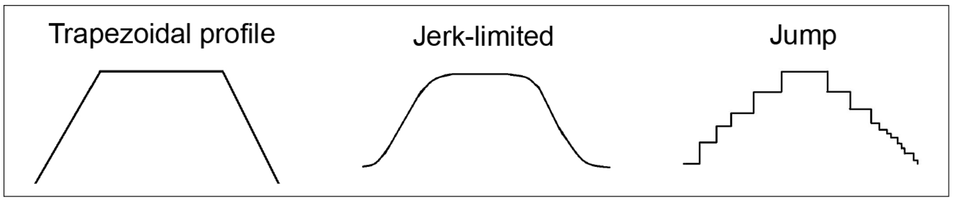
DESCRIPTION: The ramp type for the positioning movement is entered.

Can only be written when the motor is de-energized.

- Trapezoid
  A trapezoid is taken as the basis for the calculation of the motion profile.
- Jerk-limited (0x01)
  A path with jerk limitation is calculated for positioning movements *"MOVE POS"* ▸ 3.1.2 [🗋 25] and *"MOVE POS REL"* ▸ 3.1.3 [🗋 27]. The parameter "jerk-limited" (*"SET TARGET JERK" 35* ▸ 3.1.7 [🗋 31]) is used for this ramp type.
- Trapezoid SRU
  A trapezoid is taken as the basis for the calculation of the motion profile. The traveling time is not calculated. (Change-over points are controlled according to positions)

- Jump
  A path profile is not calculated here, but rather the position jump directly specified. Internal path planning is switched off. Depending on the interpolation interval of the external interpolator, it may be necessary to adapt the *controller parameters* ▸ 5.3.2.3.4 [🗋 78].

Data type: *(ENUM)* ▸ 7.1 [🗋 112] RAMP_TRAPEZ - MAX_RAMP_MODE-1



Towing error

PARAMETER CODE: 0x7D6C

Read - write access rights:

USER - PROFI

DESCRIPTION: Towing error. This value must not be exceeded during a positioning movement. Exceeding this value results in an error message ("ERROR TOW" ▸ 4.2.34 [🗋 55]).

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

Motion waiting time blocked

PARAMETER CODE: 0x7D6D

Read - write access rights:

USER - PROFI

DESCRIPTION: Time in [sec] which must elapse to trigger the *CMD MOVE BLOCKED* ▸ 3.2.2 [🗋 37] flag or command. This value must be smaller than half the *brake timeout* ▸ 5.3.2.7.4 [🗋 96] for configured *brake* ▸ 5.3.2.7.2 [🗋 95] and *"normal"* ▸ 5.3.2.7.3 [🗋 95] brake use. (The "blocking detection" will otherwise interfere directly after the start of a movement command.)

Data type: (FLOAT) 0 - MAX_FLOAT

5.3.2.6 Gear

Serial number

PARAMETER CODE: 0x7D37

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the gear.

Can only be written when the motor is de-energized.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

Ratio 1

PARAMETER CODE: 0x7D38

Read - write access rights:

USER - PROFI

DESCRIPTION: Gear ratio or reduction ratio from motor to output.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

Ratio 2

PARAMETER CODE: 0x7D39

Read - write access rights:

USER - PROFI

DESCRIPTION: Gear ratio or reduction ratio from position measuring system to output. This is only required if the *position measuring system* ▸ 5.3.2.5.3 [🗋 90] is installed between the gearing.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

5.3.2.7 Brake

Serial number

PARAMETER CODE: 0x7D3C

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Serial number of the brake.

Can only be written when the motor is de-energized.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

Type

PARAMETER CODE: 0x7D3D

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Brake type. The brake voltage is adjusted per software using the configured motor voltage and the brake type.

Can only be written when the motor is de-energized.

Data type: (ENUM) ▸ 7.1 [🗋 112] BRAKE_TYPE_NONE - MAX_BRAKETYPE-1

⚠ DANGER

Serious injury or death due to brake not operating properly is possible

A configuration error leads to a non-functioning brake.

- Ensure that the configuration is free of errors

Brake usage

PARAMETER CODE: 0x7D3E

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Use of the brake.

Can only be written when the motor is de-energized.

- Is not used (0x00)

  The brake is not used. Only engaged in the event of power failure. It is released once, immediately after the module has been started.
- Only in case of error (0x01)

  The brake is only engaged in case of error and is released for the first movement command. Otherwise, the drive is permanently controlled.
- Normal (0x02)

  The brake is engaged in the event of an error and at the end of the movement.

If a brake is configured, then the *pseudo-absolute encoder* ▸ 1.8 [🗋 19] may be active if other conditions are met.

Data type: (ENUM) BRAKE_USE_NO USE - MAX_BRAKE_USAGE-1

Timeout

PARAMETER CODE: 0x7D3F

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Time which the brake needs to build up or down the magnetic field.

Can only be written when the motor is de-energized.

Data type: (FLOAT) 0 - MAX_FLOAT

5.3.2.8 Voltage

Min. motor voltage

PARAMETER CODE: 0x7D32

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Minimum permitted motor voltage. If this value is not met, a *low-voltage error* ▶ 4.2.55 [🗋 59] will be triggered.

Data type: (FLOAT) 10 - motorMaxVolt

Max. motor voltage

PARAMETER CODE: 0x7D33

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permitted motor voltage. If this value is exceeded, a *high-voltage error* ▶ 4.2.56 [🗋 59] will be triggered. If this error occurs repeatedly, the module is blocked and can only be put back into operation by a service employee.

Data type: (FLOAT) motorMinVolt - 72

NOTE

An external brake chopper might have to be used.

Min. logic voltage

PARAMETER CODE: 0x7D34

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Minimum permitted logic voltage. If this value is not met, a low-voltage error will be ▸ 4.2.53 [🗋 58] triggered.

Data type: (FLOAT) 5 - logicMaxVolt

Max. logic voltage

PARAMETER CODE: 0x7D35

Read - write access rights:

USER - ADVANCED

DESCRIPTION: Maximum permitted logic voltage. If this value is exceeded, a *high-voltage error* ▸ 4.2.54 [🗋 58] will be triggered.

Data type: (FLOAT) logicMinVolt - 30

5.3.2.9 Info

EEPROM version

PARAMETER CODE: 0x7D9B

Read - write access rights:

USER - ROOT

DESCRIPTION: Version number of the EEPROM data set.

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

EEPROM CRC

PARAMETER CODE: 0x7D9C

Read - write access rights:

USER - ROOT

DESCRIPTION: Check sum via all EEPROM data.

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

Data CRC

PARAMETER CODE: 0x7D9D

Read - write access rights:

USER - ROOT

DESCRIPTION: Check sum via all EEPROM data which is not module-specific.

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

Error code 0

PARAMETER CODE: 0x7DA0

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Error that occurred last. (nth error)

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 1

PARAMETER CODE: 0x7DA1

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Error that occurred second from the last. (nth-1 error)

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 2

PARAMETER CODE: 0x7DA2

Read - write access rights:

PROFI - ROOT

DESCRIPTION: Error that occurred third from the last. (nth-2 error)

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 3

PARAMETER CODE: 0x7DA3

Read - write access rights:

PROFI - ROOT

DESCRIPTION: n-3 error.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 4
PARAMETER CODE: 0x7DA4
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-4 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 5
PARAMETER CODE: 0x7DA5
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-5 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 6
PARAMETER CODE: 0x7DA6
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-6 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 7
PARAMETER CODE: 0x7DA7
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-7 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 8
PARAMETER CODE: 0x7DA8
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-8 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 9
PARAMETER CODE: 0x7DA9
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-9 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 10
PARAMETER CODE: 0x7DAA
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-10 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 11
PARAMETER CODE: 0x7DAB
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-11 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 12
PARAMETER CODE: 0x7DAC
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-12 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 13
PARAMETER CODE: 0x7DAD
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-13 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 14
PARAMETER CODE: 0x7DAE
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-14 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 15
PARAMETER CODE: 0x7DAF
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-15 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 16
PARAMETER CODE: 0x7DB0
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-16 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 17
PARAMETER CODE: 0x7DB1
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-17 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 18
PARAMETER CODE: 0x7DB2
Read - write access rights:
PROFI - ROOT
DESCRIPTION: n-18 error.
Data type: (UINT8) MIN_UINT8 - MAX_UINT8

Error code 19

PARAMETER CODE: 0x7DB3

Read - write access rights:

PROFI - ROOT

DESCRIPTION: n-19 error.

Data type: (UINT8) MIN_UINT8 - MAX_UINT8

5.3.3 Asynchronous

5.3.3.1 Current main board temperature

PARAMETER CODE: 0x7DB9

Read - write access rights:

PROFI - Disabled

DESCRIPTION: measured temperature of the first temperature sensor.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.3.2 Current motor temperature

PARAMETER CODE: 0x7DBA

Read - write access rights:

PROFI - Disabled

DESCRIPTION: measured temperature of the motor temperature sensor.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.3.3 Current OPT temperature Comm.

PARAMETER CODE: 0x7DBB

Read - write access rights:

PROFI - Disabled

DESCRIPTION: measured temperature of the second temperature sensor.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.3.4 Error line

PARAMETER CODE: 0x7DBC

Read - write access rights:

USER - Disabled

DESCRIPTION: Code line in which the last error was triggered.

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.5 Error value

PARAMETER CODE: 0x7DBD

Read - write access rights:

PROFI - Disabled

DESCRIPTION: Detail which led to the last error.

(Important information for service purposes.)

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.3.6 Error file

PARAMETER CODE: 0x7DBE

Read - write access rights:

USER - Disabled

DESCRIPTION: File name in which the last error was triggered.

(Important information for service purposes.)

Data type: (CHAR_ARRAY) -

5.3.3.7 Firmware type

PARAMETER CODE: 0x7DBF

Read - write access rights:

USER - Disabled

DESCRIPTION: Software type of the module

(Important information for service purposes.)

Data type: *(ENUM)* ▸ 7.1 [🗋 112] SW_TYPE_PTA - MAX_SW_TYPE-1

5.3.3.8 Order number

PARAMETER CODE: 0x7DC0

Read - write access rights:

USER - Disabled

DESCRIPTION: Order number of the module

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.9 Module type

PARAMETER CODE: 0x7DC1

Read - write access rights:

USER - Disabled

DESCRIPTIONModule type

(Important information for service purposes.)

Data type: (CHAR_ARRAY) -

5.3.3.10 Creation date of the main-board firmware

PARAMETER CODE: 0x7DC2

Read - write access rights:

USER - Disabled

DESCRIPTION: Compiler date of the software on the main board.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.11 Creation time of the main-board firmware

PARAMETER CODE: 0x7DC3

Read - write access rights:

USER - Disabled

DESCRIPTION: Compiler time of the software on the main board.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.12 Hardware version on the main board

PARAMETER CODE: 0x7DC4

Read - write access rights:

USER - Disabled

DESCRIPTIONHardware version on the main board

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.13 Firmware version on the main board

PARAMETER CODE: 0x7DC5

Read - write access rights:

USER - Disabled

DESCRIPTION: Software version on the main board.

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.14 Creation date OPT firmware Comm.

PARAMETER CODE: 0x7DC6

Read - write access rights:

USER - Disabled

DESCRIPTION: Compiler date of the software on the communication board.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.15 Creation time OPT firmware Comm.

PARAMETER CODE: 0x7DC7

Read - write access rights:

USER - Disabled

DESCRIPTION: Compiler time of the software on the communication board.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.16 Hardware OPT version Comm.

PARAMETER CODE: 0x7DC8

Read - write access rights:

USER - Disabled

DESCRIPTIONHardware version on the communication board

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.17 Firmware OPT version Comm.

PARAMETER CODE: 0x7DC9

Read - write access rights:

USER - Disabled

DESCRIPTION: Software version on the communication board.

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.18 OPT serial number Comm.

PARAMETER CODE: 0x7DCA

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the communication board.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.19 Creation date OPT 1 firmware

PARAMETER CODE: 0x7DCB

Read - write access rights:

USER - Disabled

DESCRIPTION: Compiler date of the software on the expansion board.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.20 Creation time OPT 1 firmware

PARAMETER CODE: 0x7DCC

Read - write access rights:

USER - Disabled

DESCRIPTION: Compiler time of the software on the expansion board.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.21 Hardware OPT 1 version

PARAMETER CODE: 0x7DCD

Read - write access rights:

USER - Disabled

DESCRIPTION: Hardware version on the expansion board

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.22 Firmware OPT 1 version

PARAMETER CODE: 0x7DCE

Read - write access rights:

USER - Disabled

DESCRIPTION: Software version on the expansion board

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.23 OPT 1 serial number

PARAMETER CODE: 0x7DCF

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the expansion board.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.24 Creation date OPT 2 firmware

PARAMETER CODE: 0x7DD0

Read - write access rights:

PROFI - Disabled

DESCRIPTION: Compiler date of the software on the expansion board 2.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.25 Creation time OPT 2 firmware

PARAMETER CODE: 0x7DD1

Read - write access rights:

PROFI - Disabled

DESCRIPTION: Compiler time of the software on the expansion board 2.

(Important information for service purposes.)

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.26 Hardware OPT 2 version

PARAMETER CODE: 0x7DD2

Read - write access rights:

USER - Disabled

DESCRIPTION: Hardware version on the expansion board 2.

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.27 Firmware OPT 2 version

PARAMETER CODE: 0x7DD3

Read - write access rights:

USER - Disabled

DESCRIPTION: Software version on the expansion board 2.

(Important information for service purposes.)

Data type: (UINT16) MIN_UINT16 - MAX_UINT16

5.3.3.28 OPT 2 serial number

PARAMETER CODE: 0x7DD4

Read - write access rights:

USER - SCHUNK

DESCRIPTION: Serial number of the expansion board 2.

Data type: (UINT32) MIN_UINT32 - MAX_UINT32

5.3.3.29 Motor voltage

PARAMETER CODE: 0x7DD6

Read - write access rights:

USER - Disabled

DESCRIPTION: Current measured motor voltage [V].

Data type: (FLOAT) 0 - MAX_FLOAT

5.3.3.30 Logic voltage

PARAMETER CODE: 0x7DD7

Read - write access rights:

USER - Disabled

DESCRIPTION: Current measured logic voltage [V].

Data type: (FLOAT) 0 - MAX_FLOAT

5.3.3.31 Max. software end stop

PARAMETER CODE: 0x7DD8

Read - write access rights:

PROFI - PROFI

DESCRIPTION: Temporary software end stop, max. value The software end stops can be changed with this without saving them permanently in the EEPROM.

Data type: (FLOAT) -MAX_FLOAT - MAX_FLOAT

5.3.3.32 Min. software end stop

PARAMETER CODE: 0x7DD9

Read - write access rights:

PROFI - PROFI

DESCRIPTION: Temporary software end stop, min. value The software end stops can be changed with this without saving them permanently in the EEPROM.

Data type: (FLOAT) -MAX_FLOAT - MAXFLOAT

5.3.3.33 User

PARAMETER CODE: 0x7DDA

See the chapter „User administration" ▸ 1.7 [🗋 18]

DESCRIPTION: With read access the current user is returned. With write access a string with the matching password must be transferred.

Data type: *(ENUM)* ▸ 7.1 [🗋 112] USER_USER - MAX_USER-1

6 Remarks

6.1 Quick test

If neither the USB device nor the USB host is active, a quick test can be performed using the "Test 1" and "Test 2" switches (see hardware instructions). The following commands are supported:

- Test1 - OFF Test2 - OFF : Wait for new command
- Test1 - ON Test2 - OFF: *CMD ACK* ▸ 4.1.4 [🗋 50]
- Test1 - OFF Test2 - ON: Relative travel by 1.0 [mm] or 1.0 [degree] with 10% maximum speed, 10% maximum acceleration, 50% nominal current and 50% maximum jerk.
- Test1- ON Test2- ON: *CMD REFERENCE*

To end the test mode, the module must be restarted (switch the logic voltage off and on again).

6.2 Modules

Description of connections to the module	Since there are a wide variety of *Schunk* modules with different connection assignments, please refer to the supplied documentation for your module for more on this.
The module cannot reference from every position	The maximum permitted reference current is not enough to move the module. Increase the *max. reference current* ▸ 5.3.2.4.5 [🗋 87].

6.3 Protocol

Fragmentation is not possible	The entire module can be fully operated without data fragmentation being necessary. This applies to all supported communication interfaces.

6.4 Profibus

Data cannot be transmitted consistently	1. Set D-Len (first byte) to 0x00, populate all data and, finally, set D-Len to the required value. 2. Use the SYNC, FREEZE mechanism.

7 Appendix

7.1 List of the enumerations

Value	MotorType	BrakeType	BrakeUsage
0	MOTOR DC	BRAKE TYPE NONE	BAKE USE NO USE
1	MOTOR BLDC	BRAKE TYPE MAGNETIC 12V	BRAKE USE ERROR ONLY
2	MOTOR PMSM	BRAKE TYPE MAGNETIC 24V	BRAKE USE NORMAL
3	not used	BRAKE TYPE MAGNETIC 48V	MAX BRAKE USAGE
4	not used		
5	MAX MOTOR TYPE		

Value	ReferenceType	User	Controller Mode
0	REF SWITCH, INTERNAL LEFT	User	CTRL CURRENT SPEED
1	REF SWITCH, INTERNAL RIGHT	reserved	CTRL CASCADE
2	REF SWITCH EXTERN LEFT	PROFI	CTRL SPEED CUR LIMIT
3	REF SWITCH EXTERN RIGHT	Advanced	CTRL SPPED PWM LIMIT
4	REF VEL LEFT	Root	CTRL POS CASCADE
5	REF VEL RIGH	reserved	MAX CTL MODE
6	REF VEL LEFT DIST	MAX USER	
7	REF VEL RIGHT DIST		
8	REF CURRENT LEFT		
9	REF CURRENT RIGHT		
10	REF CURRENT LEFT DIST		
11	REF CURRENT RIGHT DIST		
12	REF NONE		
13	not used		
14	MAX REFTYPE		

Value	PosSystemType	PosSystemMount	PosSystemResolver Freq
0	POS SYSTEM ENCODER	MOUNT INPUT SIDE	RESOLVER FREQ 1kHz
1	POS SYSTEM ENCODER INDEX	MOUNT OUTPUT SIDE	RESOLVER FREQ 2kHz
2	POS SYSTEM RESOLVER	MOUNT MIDDLE SIDE	RESOLVER FREQ 4kHz

Value	PosSystemType	PosSystemMount	PosSystemResolver Freq
3	reserved	reserved	RESOLVER FREQ 8kHz
4	reserved	MAX POS MOUNT	MAX RESOLVER FREQ
5	reserved		
6	POS SYSTEM ENCODER DIFF		
7	POS SYSTEM ENCODER DEF INDEX		
8	reserved		
9	reserved		
10	reserved		
11	reserved		
12	MAX POS SYSTEM TYPE		

Value	RampModeType	DigitalInType	DigitalOutType
0	RAMP TRAPEZOID	DIGITAL_IN_NORMAL	DIGITAL_OUT_NORMAL
1	RAMP JERK	reserved	reserved
2	RAMP TRAPEZOID SRU	reserved	reserved
3	RAMP NO RAMP	reserved	reserved
4	RAMP USER	reserved	reserved
5	RAMP LINEAR	reserved	reserved
6	MAX RAMP MODE	reserved	MAX DIGITAL OUT TYPE
7		MAX DIGITAL IN TYPE	
8			
9			
10			
11			
12			
13			
14			

Value	Data type		
0	DT_UNKOWN		
1	DT_UINT8		
2	DT_INT8		
3	DT_UINT16		
4	DT_INT16		
5	DT_UINT32		

Value	Data type		
6	DT_INT32		
7	DT_UINT64		
8	DT_INT64		
9	DT_FLOAT		
10	DT_DOUBLE		
11	DT_CHAR_ARRAY		
12	DT_BOOL		
13	DT_BINARY		
14	DT_ENUM		
15	MAX_DATA_TYPE		

7.2 Information codes and error codes

Hex	Dec	Name	Page
0x0001	1	INFO BOOT	▸ 4.2.1 [🗋 51]
0x03	3	INFO NO RIGHTS	▸ 4.2.2 [🗋 51]
0x04	4	INFO UNKNOWN COMMAND	▸ 4.2.3 [🗋 51]
0x05	5	INFO FAILED	▸ 4.2.4 [🗋 51]
0x06	6	NOT REFERENCED	▸ 4.2.5 [🗋 51]
0x0007	7	INFO SEARCH SINE VECTOR	▸ 4.2.6 [🗋 51]
0x0008	8	INFO NO ERROR	▸ 4.2.7 [🗋 52]
0x09	9	INFO COMMUNICATION ERROR	▸ 4.2.8 [🗋 52]
0x10	16	INFO TIMEOUT	▸ 4.2.9 [🗋 52]
0x12	18	INFO WRONG DATA TYPE	▸ 4.2.10 [🗋 52]
0x19	25	INFO CHECKSUM	▸ 4.2.12 [🗋 52]
0x1B	27	INFO VALUE LIMIT MAX	▸ 4.2.13 [🗋 52]
0x1C	28	INFO VALUE LIMIT MIN	▸ 4.2.14 [🗋 53]
0x1D	29	INFO MESSAGE LENGTH	▸ 4.2.15 [🗋 53]
0x1E	30	INFO WRONG PARAMETER	▸ 4.2.16 [🗋 53]
0x23	35	INFO UNKNOWN PARAMETER	▸ 4.2.17 [🗋 53]
0x60	96	ERROR FILE NOT FOUND	▸ 4.2.18 [🗋 53]
0x61	97	ERROR FILE IS CORRUPT	▸ 4.2.19 [🗋 53]
0x62	98	ERROR FILE TYPE WRONG	▸ 4.2.20 [🗋 53]
0x64	99	ERROR FILE SYSTEM WRONG	▸ 4.2.21 [🗋 53]
0x65	100	ERROR FILE READ	▸ 4.2.22 [🗋 53]
0x66	101	ERROR FILE IS NOT CREATED	▸ 4.2.23 [🗋 54]
0x67	102	ERROR FILE WRITE	▸ 4.2.24 [🗋 54]
0x6A	106	ERROR CONNECTION TEMP LOW	▸ 4.2.47 [🗋 57]
0x6B	107	ERROR CONNECTION TEMP HIGH	▸ 4.2.48 [🗋 57]
0x6C	108	ERROR MOTOR TEMP LOW	▸ 4.2.49 [🗋 58]
0x6D	109	ERROR MOTOR TEMP HIGH	▸ 4.2.50 [🗋 58]
0x6E	110	ERROR TEMP LOW OPTION	
0x6F	111	ERROR TEMP HIGH OPTION	
0x70	112	ERROR TEMP LOW	▸ 4.2.51 [🗋 58]
0x71	113	ERROR TEMP HIGH	▸ 4.2.52 [🗋 58]
0x72	114	ERROR LOGIC LOW	▸ 4.2.53 [🗋 58]
0x73	115	ERROR LOGIC HIGH	▸ 4.2.54 [🗋 58]
0x74	116	ERROR MOTOR VOLTAGE LOW	▸ 4.2.55 [🗋 59]
0x75	117	ERROR MOTOR VOLTAGE HIGH	▸ 4.2.56 [🗋 59]
0x76	118	ERROR CABLE BREAK	▸ 4.2.57 [🗋 59]
0x7A	122	ERROR LIFE SIGN	▸ 4.2.58 [🗋 60]

Hex	Dec	Name	Page
0x7B	123	ERROR CUSTOM DEFINED	▶ 4.2.59 [🗋 60]
0x7C	123	ERROR REBOOT	▶ 4.2.25 [🗋 54]
0x7D	123	ERROR MOTOR PHASE	▶ 4.2.26 [🗋 54]
0x82	130	ERROR OVERSHOOT	▶ 4.2.60 [🗋 60]
0x83	131	ERROR HARDWARE VERSION	▶ 4.2.61 [🗋 60]
0x84	132	ERROR SOFTWARE VERSION	▶ 4.2.62 [🗋 60]
0xC8	200	ERROR WRONG RAMP TYPE	▶ 4.2.27 [🗋 54]
0xD1	210	ERROR WRONG DIRECTION	▶ 4.2.28 [🗋 54]
0xD2	210	ERROR CONFIG MEMORY	▶ 4.2.29 [🗋 54]
0xD5	213	ERROR SOFT LOW	▶ 4.2.30 [🗋 54]
0xD6	214	ERROR SOFT HIGH	▶ 4.2.31 [🗋 55]
0xD8	216	ERROR SERVICE	▶ 4.2.32 [🗋 55]
0xD9	217	ERROR FAST STOP	▶ 4.2.33 [🗋 55]
0xDA	218	ERROR TOW	▶ 4.2.34 [🗋 55]
0xDB	219	ERROR VPC3	▶ 4.2.35 [🗋 55]
0xDC	220	ERROR FRAGMENTATION	▶ 4.2.36 [🗋 55]
0xDD	221	ERROR COMMUTATION	▶ 4.2.37 [🗋 56]
0xDE	222	ERROR CURRENT	▶ 4.2.39 [🗋 56]
0xDF	223	ERROR I2T	▶ 4.2.38 [🗋 56]
0xE0	224	ERROR INITIALIZE	▶ 4.2.45 [🗋 57]
0xE1	225	ERROR INTERNAL	▶ 4.2.46 [🗋 57]
0xE4	228	ERROR TOO FAST	▶ 4.2.40 [🗋 56]
0xE5	228	ERROR POS SYSTEM	▶ 4.2.41 [🗋 56]
0xEB	229	ERROR RESOLVER CHECK FAILED	▶ 4.2.42 [🗋 56]
0xEC	236	ERROR MATH	▶ 4.2.43 [🗋 56]
0xEE	236	ERROR CALIB CURRENT	▶ 4.2.44 [🗋 57]

7.3 Firmware update

A firmware update can only be done via USB.

- USB Device: the software tool *MTS* (see "Schunk Motion Tool" software manual) can be used to perform a firmware update.
- USB host:
  - Copy the desired firmware file onto a USB flash drive
  - Rename it "firmware.bin"
  - Connect USB flash drive to module
  - Confirm USB enable (hardware manual)
  - Enable update command via corresponding switch (hardware manual)
  - Wait for update process
  - Restart module

SCHUNK GmbH & Co. KG
Clamping and gripping technology

Bahnhofstr. 106 - 134
D-74348 Lauffen/Neckar
Tel. +49-7133-103-0
Fax +49-7133-103-2399
info@de.schunk.com
schunk.com

Folgen Sie uns I *Follow us*

01-2022 © 2022 SCHUNK GmbH & Co. KG 02.00 | SCHUNK motion protocol | Software manual | en | 0389052