SmarAct GmbH

# *SCU - Simple Control Unit*
# *RS232 Interface Documentation*

Flötenstraße 70
D 26125 Oldenburg
http://www.smaract.de
Tel.: +49 (0) 441 - 800879 - 0
Fax: +49 (0) 441 - 800879 - 21
E-Mail: info@smaract.de

# Table of Contents



Document Version: 10-02-19-194

# 1 Introduction

The HCU-3D and CU-3D use an RS232 interface to receive commands and provide status information. This document describes the RS232 software interface of the HCU-3D / CU-3D.

On power-up the device will initialize itself and wait for a valid RS232 signal (logic high) on its RX line. The device will answer with a logic high on its TX line and is then ready to receive commands.

The interface parameters are set to “8N1” (8 data bits, no parity, one stop bit). The factory default baud rate is set to 9,600. Depending on the product version the baud rate may be configured by the user (see CB-command).

## *1.1 Command Format*

Each command consists of an initial character (':', 0x3a), an ASCII string coding the actual command (hereafter referred to as *command string*) and a termination character (line feed, 0x0a). Empty strings (i.e. a line feed character that follows a colon character) are ignored. All characters between a line feed and a colon are also ignored.

The general command string format is:

<command><channelIndex>[<parameter><value>] ... [<parameter><value>]

Some parameters are required, some are optional. Optional parameters may be given in any order. If omitted they are treated as given with a default value of 0.

## *1.2 Answer Format*

Some commands cause the device to send an answer, such as status information. Answers are again ASCII strings (hereafter referred to as *answer string*) that are preceded by a colon and terminated by a line feed character.

As with the commands the general answer string format is:

<answer><channelIndex>[<parameter><value>] ... [<parameter><value>]

In this case the channel index is the answer source. If the answer does not originate from a specific channel, then the channel index will be 255, indicating the device in general.

## *1.3 Command Summary*

The following table summarizes the instruction set of the device. Upper case letters mark a single ASCII character. An 'x' marks a parameter value.

The command and answer strings are described in more detail below (see section Command Description). Note that not all commands might be available, depending on your product version.

| Command | Meaning | Answer | Meaning |
|---|---|---|---|
| **Initialization Commands** | | | |
| I | get device Identification | Ix | device Identification |
| GID | Get device ID number | IDx | device ID number |
| V | get firmware Version | Vx.x.x | firmware Version |
| R | Device reset | | |
| **Configuration Commands** | | | |
| AxAx | set step Amplitude | | |
| SSPxSxAxFx | Set Step Parameters | | |
| SZx | Set Zero position | | |
| SCLFxFx | Set Closed-Loop max Frequency | | |
| GCLFx | Get Closed-Loop max Frequency | CLFxFx | current frequency |
| GSPx | Get Sensor Present | SPxP / SPxN | sensor Present / Not present |
| SSTxTx | Set Sensor Type | | |
| GSTx | Get Sensor Type | STxTx | sensor type |
| SPAxAxFxBx | Set Positioner Alignment | | |
| GPAx | Get Positioner Alignment | PAxAxFxBx | alignment, forward amplitude, backward amplitude |
| SSDxDx | Set Safe Direction | | |
| GSDx | Get Safe Direction | SDxDx | direction |
| **Movement Control Commands** | | | |
| UxFxAxSx | move Up | | |
| DxFxAxSx | move Down | | |
| MPAxPxHx | Move to Position Absolute | | |
| MPRxPxHx | Move to Position Relative | | |
| MAAxAxRxHx | Move to Angle Absolute | | |
| MARxAxRxHx | Move to Angle Relative | | |
| MTRxHxZx | Move To Reference | | |
| MESxDx | Move to End-Stop | | |
| CSx | Calibrate Sensor | | |
| Sx | Stop | | |
| **Positioner Feedback Commands** | | | |
| GPx | Get Position | PxPx | Position |
| GAx | Get Angle | AxAxRx | Angle and Revolution |
| Mx | get movement status | Mxx | Movement status |
| **Miscellaneous Commands** | | | |
| Kx | Keep alive | | |
| Ex | get Error status | Ex | Error status |
| CBx | Configure rs232 Baud rate. | CBx | Configured Baud rate |

## 1.4 Errors

Whenever a command string can not be processed an error is generated. There are two modes of error handling. The *auto error report mode* and the *non auto error report mode*, the latter being the default.

- In non auto error report mode commands that don't naturally invoke an answer return no answer string, but instead write the resulting error code to an internal register that may be read out via the E-command. If no error occurred a value of 0 is written to the register.
- In auto error report mode all commands return an answer string in any case. Query commands may return the normal answer string or an error string. Other commands always return an error string (with code 0 in case of no error). Note that in this mode the E-command always returns a value of 0, since errors are already implicitly reported and the internal error code register has been set to 0.

Returned error strings have the format E<code>. The error code <code> indicates the cause of the failure. A code of 0 indicates no error and corresponds to an acknowledge in auto error report mode. Please refer to the appendix for a list of error codes.

# 2 Command Description

## *2.1 Initialization Commands*

### I - get device Identification

**Description:**

This command may be used to verify that you are communicating with the correct RS232 device. It returns a string that identifies the device.

**Command String Format:**

`I`

**Parameters:**

none

**Answer string:**

`I<identification>`

- <identification>: Device identification.

**Example:**

Command: `I`

Answer: `ISmarAct HCU-3D`

Requests the device identification, which is "SmarAct HCU-3D" or "SmarAct CU-3D" depending on the model.

# GID - Get device ID number

**Description:**

This command may be used to physically identify a device. Each device has a unique ID which makes it possible to distinguish one from another. Useful when communicating with several devices.

**Command String Format:**

`GID`

**Parameters:**

none

**Answer String:**

`ID<IDnumber>`

- <IDnumber>: Decimal device ID number.

**Example:**

Command: `GID`

Answer: `ID1234567890`

Requests the device ID number, which is "1234567890".

## V - get firmware Version

**Description:**

This command may be used to retrieve the firmware version that is installed on the device.

**Command String Format:**

`V`

**Parameters:**

none

**Answer String:**

`V<versionHigh>.<versionLow>.<versionBuild>`

- <versionHigh>: High number of the firmware version.
- <versionLow>: Low number of the firmware version
- <versionBuild>: Build number of the firmware version

**Example:**

Command: `V`

Answer: `V1.2.3`

Requests the version of the firmware on the device, which is "1.2.3".

# R - device Reset

**Description:**

When this command is sent to the device it will perform a reset. It has the same effect as a power-down/power-up cycle. No answer is sent before the reset. The device is ready to receive commands again after the TX line has gone logic high.

**Command String Format:**

`R`

**Parameters:**

none

**Answer String:**

none

**Example:**

Command: `R`

Initiates a reset of the device. It will be ready to receive commands again after the TX line has gone logic high.

## *2.2 Configuration Commands*

# A - set step Amplitude

**Description:**

This command sets the amplitude for the selected channel. An ongoing movement of this channel is implicitly stopped. If the channel index is set to 99 the command will be executed for all channels at once.

Please note that this command is deprecated. The SSP command should be used instead.

**Command String Format:**

`A<channel>A<amplitude>`

**Parameters:**

- <channel>: Zero-based channel index. If the index is set to 99 the command will be executed for all channels. Valid indices are 0, 1, 2 and 99.
- <amplitude>: Amplitude that the steps are performed with. Lower amplitude values result in a smaller step width. The parameter must be given in tenths of Volts and has a valid range of 150 to 1,000. 0 corresponds to 0V, 1,000 to 100V.

**Answer String:**

none

**Example:**

Command: `A2A1000`

Sets the amplitude of channel 2 to 100 Volts. If the positioner is currently moving, it is stopped.

# SSP - Set Step Parameters

**Description:**

This command sets the step parameters for a channel without performing an actual movement. It may be useful to preset the parameters and then use short command strings such as “D0”. This command is also used in conjunction with the hardware step/direction interface (CU only).

An ongoing movement of this channel is implicitly stopped. If the channel index is set to 99 the command will be executed for all channels at once.

**Command String Format:**

`SSP<channel>S<steps>A<amplitude>F<frequency>`

**Parameters:**

- <channel>: Zero-based channel index. If the index is set to 99 the command will be executed for all channels. Valid indices are 0, 1, 2 and 99.
- <steps>: Number of steps that are to be performed per burst. The valid range is 1 .. 30,000. The special value of 30,000 indicates an unbounded move.
- <amplitude>: Amplitude that the steps are performed with. Lower amplitude values result in a smaller step width. The parameter must be given in tenths of Volts and has a valid range of 150 to 1,000. 0 corresponds to 0V, 1,000 to 100V.
- <frequency>: Frequency in Hz that the steps are performed with. The valid range is 1 to 18,500.

**Answer String:**

none

**Example:**

Commands:
```
SSP2S10F500
D2
D2
```

Configures a burst of 10 steps at 500Hz with an unchanged amplitude and then performs two bursts in down direction.

## SZ - Set Zero position

**Description:**

For positioners that have a sensor installed, this command may be used to define the current position of the positioner as the zero position.

**Command String Format:**

`SZ<channel>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.

**Answer String:**

none

**Example:**

Command: `SZ0`

Defines the current position of positioner 0 as its zero position.

# SCLF - Set Closed-Loop max Frequency

**Description:**

For positioners that have a sensor installed, this command may be used to define the maximum frequency that the positioners are driven with when issuing closed-loop movement commands (e.g. MPA). This parameter may be set for each channel independently. Once set, all subsequent closed-loop commands will execute with the new setting.

**Command String Format:**

`SCLF<channel>F<frequency>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.
- <frequency>: Defines the maximum driving frequency in Hz. The valid range is 1 .. 18,500. The default value is 5,000.

**Answer String:**

none

**Example:**

Command: `SCLF0F4000`

Sets the maximum closed-loop frequency of channel 0 to 4 kHz.

# GCLF - Get Closed-Loop max Frequency

**Description:**

This is the inverse command to SCLF. It may be used to read out the current setting for the maximum driving frequency for closed-loop commands.

**Command String Format:**

`GCLF<channel>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.

**Answer String:**

`CLF<channel>F<frequency>`

- <channel>: Zero-based channel index.
- <frequency>: Maximum closed-loop driving frequency in Hz.

**Example:**

Command: `GCLF0`

Answer: `CLF0F5000`

Returns the maximum driving frequency that is currently set for channel 0, which is the default 5 kHz.

## GSP - Get Sensor Present

**Description:**

This command may be used to check whether a sensor is present on a positioner or not.

**Command String Format:**

`GSP<channel>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.

**Answer String:**

`SP<channel>P`

in case a sensor is present or

`SP<channel>N`

in case no sensor is present.

- <channel>: Zero-based channel index.

**Example:**

Command: `GSP1`

Answer: `SP1P`

Checks whether a sensor was detected on channel 1. In the example the answer is positive.

# SST - Set Sensor Type

**Description:**

When using positioners with integrated sensors, this command may be used to tell a channel what type of positioner is connected (e.g. linear or rotary). The type affects position calculation and functions that may be called for a channel (see for example GP and GA).
Note that each channel stores this setting to non-volatile memory. Consequently, there is no need to call this function on every initialization.

Note: If this command is issued, the positioner is implicitly stopped.

**Command String Format:**

`SST<channel>T<type>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.
- <type>: The type code of the sensor. See the table below for a list of sensor types.

**Answer String:**

none

**Example:**

Command: `SST2T1`

Sets the sensor type of channel 2 to 1 (linear).

| Sensor Type | Positioner Series | Positioner Type |
|---|---|---|
| M | SL-xxxx-M,<br>SLC-xxxx-M | Linear positioner |
| GA | SGO-45-M | Goniometer with 45mm radius |
| GB | SGO-50-M | Goniometer with 50mm radius |
| GC | SR-2110-M | Rotary positioner with end-stops |
| GD | SGO-60.5-M | Goniometer with 60.5mm radius |
| GE | SGO-77.5-M | Goniometer with 77.5mm radius |
| RA | SFW-M | Filter wheel with absolute position sensor |
| GF | SR1209-M | Rotatory positioner |
| RB | SR1910-M | Rotatory positioner |

## GST - Get Sensor Type

**Description:**

Inverse command to SST. It may be used to check which sensor type is currently configured for a channel.

**Command String Format:**

`GST<channel>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.

**Answer String:**

`ST<channel>T<type>`

- <channel>: Zero-based channel index.
- <type>: Currently configured sensor type code. See the appendix for a list of sensor types.

**Example:**

Command: `GST0`

Answer: `ST0T1`

Returns the sensor type code that is currently configured for channel 0, which is 1 in this case.

# SPA - Set Positioner Alignment

**Description:**

When using positioners with integrated sensors, this function is useful for positioners that are mounted vertically and carry high loads. It may be used to tell a channel how the positioner is physically aligned (horizontally or vertically). This setting affects only closed-loop control of a positioner. Open-loop commands are unaffected. Please refer to section 4.1 “Vertical Axis Configuration” for more information.

Note that each channel stores this setting to non-volatile memory. Consequently, there is no need to call this command on every initialization.

**Command String Format:**

`SPA<channel>A<alignment>F<forwardAmplitude>B<backwardAmplitude>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.
- <alignment>: Alignment of the positioner. Must be either 0 (horizontal, default) or 1 (vertical).
- <forwardAmplitude>: Only valid for vertical alignment. Omit for horizontal alignment. Specifies the step amplitude that is used when the positioner is moved in forward direction while in closed-loop control. Lower amplitude values result in a smaller step width. The parameter must be given as an integer value in tenths of Volts. The valid range is 150 (15V) to 1000 (100V).
- <backwardAmplitude>: Only valid for vertical alignment. Omit for horizontal alignment. Specifies the step amplitude that is used when the positioner is moved in backward direction while in closed-loop control. Lower amplitude values result in a smaller step width. The parameter must be given as an integer value in tenths of Volts. The valid range is 150 (15V) to 1000 (100V).

**Answer String:**

none

**Example:**

Command: `SPA0A1F1000B600`

Configures channel 0 for vertical positioning with a backward amplitude of 60V.

## GPA - Get Positioner Alignment

**Description:**

Inverse command to SPA. It may be used to check which positioner alignment is currently configured for a channel.

**Command String Format:**

`GPA<channel>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.

**Answer String:**

`PA<channel>A0`

in case of horizontal alignment, or

`PA<channel>A1F<forwardAmplitude>B<backwardAmplitude>`

in case of vertical alignment.

- <channel>: Zero-based channel index.
- <forwardAmplitude>: Currently configured amplitude for forward movement given in 1/10 Volts.
- <backwardAmplitude>: Currently configured amplitude for backward movement given in 1/10 Volts.

**Example:**

Command: `GPA0`

Answer: `PA0A0`

Returns the alignment that is currently configured for channel 0, which is horizontal in this case.

# SSD - Set Safe Direction

**Description:**

When using positioners with integrated sensors this command may be used to configure the safe direction for a channel. The safe direction is the direction in which the positioner may safely move while calibrating (see CS command) or moving to the reference position (see MTR command).

When calling this function the setting is saved to non-volatile memory. Therefore, the safe direction does not have to be configured in each session. Note though that if the safe direction is changed then the sensor must be calibrated again for the MTR command to work properly.

**Command String Format:**

`SSD<channel>D<direction>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.
- <direction>: Safe direction. Must be either 0 (backward) or 1 (forward).

**Answer String:**

none

**Example:**

Command: `SSD0D1`

Configures the safe direction for channel 0 to forward.

## GSD - Get Safe Direction

**Description:**

Inverse command to SSD. It may be used to check which safe direction is currently configured for a channel.

**Command String Format:**

`GSD<channel>`

**Parameters:**

- <channel>: Zero-based channel index of the device. Valid indices are 0, 1 and 2.

**Answer String:**

`SD<channel>D<direction>`

- <channel>: Zero-based channel index.
- <direction>: Currently configured safe direction. Will be either 0 (backward) or 1 (forward).

**Example:**

Command: `GSD0`

Answer: `SD0D1`

Returns the safe direction that is currently configured for channel 0, which is forward in this case.

## 2.3 Movement Control Commands

### U - move Up

**Description:**

This command causes the selected channel to perform a movement in “up” direction with the given frequency, amplitude and number of steps. If the channel index is set to 99 the command will be executed for all channels at once.

The F, A and S parameters may be omitted in which case the last value for the selected channel is used. If omitted on the first call, default values are used. These are 1kHz for the frequency, 100V for the amplitude and “unlimited” for the number of steps.

**Command String Format:**

`U<channel>F<frequency>A<amplitude>S<steps>`

**Parameters:**

- <channel>: Zero-based channel index. If the index is set to 99 the command will be executed for all channels. Valid indices are 0, 1, 2 and 99.
- <frequency>: Frequency in Hz that the steps are performed with. The valid range is 1..18,500.
- <amplitude>: Amplitude that the steps are performed with. Lower amplitude values result in a smaller step width. The parameter must be given in tenths of Volt and has a valid range of 150 to 1,000. 0 corresponds to 0V, 1,000 to 100V.
- <steps>: Number of steps to be performed. The valid range is 1..30,000. The special value of 30,000 indicates an unbounded move.

**Answer String:**

none

**Example:**

Command: `U99F5A500`

Moves all positioners in positive direction with a step frequency of 5Hz, an amplitude of 50V and the corresponding last number of steps. If a positioner is currently moving, the previous movement command is overridden.

# D - move Down

**Description:**

This command causes the selected channel to perform a movement in “down” direction with the given frequency, amplitude and number of steps. If the channel index is set to 99 the command will be executed for all channels at once.

The F, A and S parameters may be omitted in which case the last value for the selected channel is used. If omitted on the first call, default values are used. These are 1kHz for the frequency, 100V for the amplitude and “unlimited” for the number of steps.

**Command String Format:**

`D<channel>F<frequency>A<amplitude>S<steps>`

**Parameters:**

- <channel>: Zero-based channel index. If the index is set to 99 the command will be executed for all channels. Valid indices are from 0, 1, 2 and 99.
- <frequency>: Frequency in Hz that the steps are performed with. The valid range is 1 to 18,500.
- <amplitude>: Amplitude that the steps are performed with. Lower amplitude values result in a smaller step width. The parameter must be given in tenths of Volt and has a valid range of 150 to 1,000. 0 corresponds to 0V, 1,000 to 100V.
- <steps>: Number of steps to be performed. The valid range is 1..30,000. The special value of 30,000 indicates an unbounded move.

**Answer String:**

none

**Example:**

Command: `D1F5`

Moves the positioner connected to channel 1 in negative direction with a step frequency of 5Hz and the amplitude and the number of steps that channel 1 was last driven with. If the positioner is currently moving, the previous movement command is overridden.

# MPA - Move to Position Absolute

**Description:**

Instructs a positioner to move to a specific position via closed-loop control. This command is only executable by a positioner that has a sensor integrated.

While executing the command the positioner has the movement status code T (targeting). After having reached the desired position, the positioner holds this position for a specified time (hold time). The movement status code then changes to H (holding). After the hold time has elapsed the positioner is stopped and has a status code of S (stopped).

**Command String Format:**

`MPA<channel>P<position>H<holdTime>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.
- <position>: Absolute position to move to in micrometers.
- <holdTime>: Time in milli seconds that the target position is to be held. The valid range is 0 .. 60,000. A value of 60,000 represents infinte.

**Answer String:**

none

**Example:**

Command: `MPA0P1000H1000`

Moves the positioner connected to channel 0 to the position 1mm which is then held for one second.

# MPR - Move to Position Relative

**Description:**

Instructs a positioner to move to a position via closed-loop control, which is given in relation to the current position. This command is only executable by a positioner that has a sensor integrated.

While executing the command the positioner has the movement status code T (targeting). After having reached the desired position, the positioner holds this position for a specified time (hold time). The movement status code then changes to H (holding). After the hold time has elapsed the positioner is stopped and has a status code of S (stopped).

**Command String Format:**

`MPR<channel>P<position>H<holdTime>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.
- <position>: Relative position to move to in micrometers.
- <holdTime>: Time in milli seconds that the target position is to be held. The valid range is 0 .. 60,000. A value of 60,000 represents infinte.

**Answer String:**

none

**Example:**

Command: `MPR2P500`

Moves the positioner connected to channel 2 500µm in positive direction. This position is not held.

# MAA - Move to Angle Absolute

**Description:**

Instructs a positioner to turn to a specific angle via closed-loop control. This command is only executable by a positioner that has a sensor integrated. Additionally, the command is only executable if the addressed channel is configured to be of type rotary (see SST command). A linear channel will return an error. See also MPA command.

While executing the command the positioner has the movement status code T (targeting). After having reached the desired position, the positioner holds this position for a specified time (hold time). The movement status code then changes to H (holding). After the hold time has elapsed the positioner is stopped and has a status code of S (stopped).

**Command String Format:**

`MAA<channel>A<angle>R<revolution>H<holdTime>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.
- <angle>: Absolute angle to turn to in millidegrees
- <revolution>: Reserved for future use. Omit or set to 0.
- <holdTime>: Time in milli seconds that the target position is to be held. The valid range is 0 .. 60,000. A value of 60,000 represents infinte.

**Answer String:**

none

**Example:**

Command: `MAA0A500`

Turns the positioner connected to channel 0 to the angle 0,5°. The angle is not actively held.

# MAR - Move to Position Relative

**Description:**

Instructs a positioner to move to an angle via closed-loop control, which is given in relation to the current angle. This command is only executable by a positioner that has a sensor integrated. Additionally, the command is only executable if the addressed channel is configured to be of type rotary (see SST command). A linear channel will return an error. See also MPA command.

While executing the command the positioner has the movement status code T (targeting). After having reached the desired position, the positioner holds this position for a specified time (hold time). The movement status code then changes to H (holding). After the hold time has elapsed the positioner is stopped and has a status code of S (stopped).

**Command String Format:**

`MAR<channel>A<angle>R<revoltution>H<holdTime>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.
- <angle>: Relative angle to move to in millidegrees.
- <revolution>: Reserved for future use. Omit or set to 0.
- <holdTime>: Time in milli seconds that the target position is to be held. The valid range is 0 .. 60,000. A value of 60,000 represents infinte.

**Answer String:**

none

**Example:**

Command: `MAR1A-1500`

Moves the positioner connected to channel 1 1,5° in negative direction. This position is not held.

# MTR - Move To Reference

**Description:**

For positioners that are equipped with a sensor, this command may be used to move the positioner to one of two known physical positions with high repeat accuracy. These positions are located near the mechanical end stops. Which end stop is moved to is implicitly controlled by the “safe direction” of the positioner (see SSD command). The end stop must be calibrated (see CS command) to ensure proper operation of this command.

Note that this function replaces the MES command, which is less accurate compared to this command.

If the auto zero flag is set then the current position is set to zero after the reference position has been reached.

While executing the command the positioner will have a movement status of R (referencing). If a hold time was specified then the movement status will change to H (holding) after having reached the reference position. Otherwise the movement status will be S (stopped).

**Command String Format:**

`MTR<channel>H<holdTime>Z<autoZero>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.
- <holdTime>: Time in milli seconds that the target position is to be held. The valid range is 0 .. 60,000. A value of 60,000 represents infinte.
- <autoZero>: Must be 0 (no auto-zero) or 1 (auto-zero). The latter will reset the current position to zero upon reaching the reference position.

**Answer String:**

none

**Example:**

**Command:** MTR1H0Z1

Moves the positioner connected to channel 1 to the reference position, thereafter setting the position to zero.

# MES - Move to End-Stop

**Description:**

Instructs a positioner to move to an end stop in a given direction. This command is only executable by a positioner that has a sensor integrated.

While executing the command the positioner has a movement status code T (targeting). After the positioner has reached the end stop, the movement status code then changes to S (stopped).

**Command String Format:**

`MES<channel>D<direction>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.
- <direction>: Direction in which the positioner shall move. The symbol can be one of the following:
  'U': for the up / positive direction
  'D': for the down / negative direction

**Answer String:**

none

**Example:**

Command: `MES1DU`

Moves the positioner connected to channel 1 to the end stop in positive direction.

# CS - Calibrate Sensor

**Description:**

This command may be used to increase the accuracy of the position calculation. It is only executable by a positioner that has an integrated sensor.

This command should be called once for each channel if the mechanical setup changes (different positioners connected to different channels). The calibration data will be saved to non-volatile memory. If the mechanical setup is unchanged, it is not necessary to call this function on each initialization, but newly connected positioners have to be calibrated in order to ensure proper operation.

The calibration consists of two phases. During the first phase the positioner will perform a movement of up to several millimeters in one direction and then in the other direction. The user must ensure, that the positioner is not near a mechanical end stop during this first phase. Otherwise the calibration might fail and lead to unexpected behavior when executing closed-loop commands.

During the second phase the positioner will move to a mechanical end stop. The direction is implicitly given by the “safe direction” which is configured via the SSD command. Once the end stop has been calibrated during this second phase you may use the MTR command to accurately move to the reference position.

The calibration takes a few seconds to complete. During this time the positioner the movement status code C (calibrating). After the calibration has finished, it will change to S (stopped).

**Command String Format:**

`CS<channel>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.

**Answer String:**

none

**Example:**

Command: `CS0`

Calibrates the sensor on channel 0.

# S - Stop

**Description:**

This command stops any ongoing activity of the positioner on the selected channel. If the channel index is set to 99 the command will be executed for all channels at once.

**Command String Format:**

`S<channel>`

**Parameters:**

- <channel>: Zero-based channel index. If the index is set to 99 the command will be executed for all channels. Valid indices are 0, 1, 2 and 99.

**Answer String:**

none

**Example:**

Command: `S99`

Stops all positioners.

## *2.4 Positioner Feedback Commands*

### GP - Get Position

**Description:**

Returns the current position of a linear positioner. This command is only executable by a positioner that has an integrated sensor.

**Command String Format:**

`GP<channel>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.

**Answer String:**

`P<channel>P<position>`

- <channel>: Zero-based channel index.
- <position>: The current position of the positioner in micrometers.

**Example:**

Command: `GP0`
Answer: `P0P-13.5`

Requests the current position of channel 0. The answer indicates that the positioner is currently at position -13.5 µm.

# GA - Get Angle

**Description:**

Returns the current angle of a rotary positioner. This command is only executable by a positioner that has an integrated sensor.

**Command String Format:**

`GA<channel>`

**Parameters:**

- <channel>: Zero-based channel index. Valid indices are 0, 1 and 2.

**Answer String:**

`A<channel>A<angle>R<revolution>`

- <channel>: Zero-based channel index.
- <angle>: The current angle of the positioner in millidegrees.
- <revolution>: Reserved for future use.

Example:

Command: `GA0`
Answer: `A0A-187.2R0`

Requests the current angle of channel 0. The answer indicates that the positioner is currently at angle -187.2m°.

# M - Movement status query

**Description:**

This command may be sent to retrieve the current movement status of a channel.

**Command String Format:**

`M<channel>`

**Parameters:**

- <channel>: Zero-based channel. If the index is set to 99 the command will be executed for all channels. Valid indices are 0, 1, 2 and 99.

**Answer String:**

`M<channel><status>`

- <channel>: Zero-based channel index.
- <status>: A letter that indicates the status of the channel. See the appendix for a list of status codes.

If no error occurs the answer string contains the channel and a letter indicating the status of the channel. If the channel index was set to '99', then three separate answer strings are generated, one for each channel.

**Example:**

Command: `M0`
Answer: `M0S`

Requests the movement status of channel 0. The answer indicates that positioner 0 is currently stopped.

## *2.5 Miscellaneous Commands*

# K - Keep-alive

**Description:**

A keep-alive timer is provided to stop all positioners in case no command or keep-alive command is sent in predefined intervals.

The K-command has two functions:

1.Keep-alive:
If no parameter is given, it causes the internal timeout counter to be reseted. If a timeout occurs, all positioners are immediately stopped.

2.Set keep-alive timeout:
If a number is given as a parameter, then the timeout is set to this value given in milliseconds. A value of 0 disables the timeout functionality.

If the timeout is enabled all (successful) commands implicitly reset the timeout counter.

It is highly recommended to use this feature when using unbounded moves.

**Command String Format:**

Keep-alive:

```
K
```

Set Keep-alive timeout:

```
K<timeout>
```

**Parameters:**

- <timeout>: Timeout delay in milliseconds. The valid range for the timeout delay is 100 to 60,000. A value of 0 (default) is also valid and disables the timeout functionality.

**Answer String:**

none

**Example:**

Command: `K5000`

Sets the timeout delay to 5 seconds. Subsequent movement commands are halted after 5 seconds if no new command is received.

Command: `K`

Keeps the movement alive having no effect otherwise.

# E - Error status

**Description:**

An internal register is provided which keeps the error code of the last command.

The E-command has two functions.

1.Get error status:
If no parameter is given, then an answer string contains the code of the last error that occurred. After this, the internal error code register is reset to zero (= no error).

2.Set auto error report mode:
A single digit may be given as a parameter to configure the device to auto error report mode or non auto error report mode. The default configuration is non auto error report mode. See section "*Errors*" for more information. See the appendix for a list of error codes and their meanings.

**Command String Format:**

Get error status:

```
E
```

Set auto error report mode:

```
E<mode>
```

**Parameters:**

- <mode>: A single digit will configure the device to
  0: non auto error report mode,
  1: auto error report mode.

**Answer String:**

In case that the E-command has been used without parameters the error code of the previous command is returned:

```
E<code>
```

In case that the E-command has been used with parameters the returned error codes indicates whether setting the auto error report mode was successful or not.

```
E<code>
```

**Parameters:**

- <code>: Error code. See the appendix for a list of error codes and their meanings.

**Examples:**

Command: `E`
Answer: `E17`

Requests the error code of the last command. The answer (error code 17) shows that an invalid parameter was given on the last command.

Command: `E1`
Answer: `E0`

Sets the current mode to be the auto error report mode. The answer shows that this mode setting has been successful.

# CB - Configure RS232 Baud rate

**Description:**

Sets the baud rate of the RS232 interface to the specified value. The setting is stored to EEPROM (non-volatile memory) and loaded on future power-ups. Note that a change to this setting does not take effect until the next system reset. After configuring the interface, either do a power-down/power-up cycle or send a reset command to make the changes take effect.

**Command String Format:**

`CB<baudrate>`

**Parameters:**

- <baudrate>: Baudrate of the RS232 interface. The valid range is 9,600 to 500,000.

**Answer String:**

The answer string contains the baud rate that was effectively configured and reflects the closest value that the internal baud rate generator is able to produce. For standard baud rates the error is small enough for a stable communication.

`CB<baudrate>`

- <baudrate>: Baudrate of the RS232 interface that was effectively configured.

**Example:**

Command: `CB57600`
Answer: `CB57142`

Configures the RS232 baud rate to 57600 baud. The baud rate will be set to 57142 baud (0.8% error) after the next reset.

# 3 General Behavior

Once the system is initialized after power up and the TX line has gone logic high it is able to receive commands at any time. The execution of previous commands does not affect the reception of new commands.

All channels may be driven at the same time. Each channel is independent of the others and has its own parameters. A movement command for a positioner that is already moving overwrites the previous command.

## *3.1 Restrictions*

If a movement command changes the amplitude for a channel, the actual movement is delayed by several milliseconds. If the positioner was previously moving, it is stopped to adjust the amplitude and then the new movement is executed. This may result in a non-smooth speed change when overwriting previous movement commands. Therefore it is recommended to change the amplitude only infrequently. The set amplitude command may be useful to change the amplitude when a positioner is currently stopped. All subsequent movement commands that omit the amplitude parameter (or supply the same value) are driven with the new amplitude.

Driving multiple channels at the same time with slightly different frequencies may result in unstable frequencies due to interference effects.

# 4 Other Issues

## *4.1 Vertical Axis Configuration*

In systems that use a positioner as a vertical axis you may experience inaccurate closed-loop positioning if the positioner has to lift fairly heavy weights. When moving upwards the positioner has to overcome gravity and therefore moves slower. Conversely when moving downwards gravity potentially pushes the positioner beyond the target position. For these situations a special mode of operation can optimize the behavior.

The SPA command may be used to configure a channel for vertical positioning. In this mode closed-loop commands are processed slightly different. Two parameters must be specified to configure the mode: a forward amplitude and a backward amplitude. These values define which step amplitudes are used when moving in either direction. Which direction points upwards and which downwards depends on which way the positioner is mounted. Generally you will want to use a full amplitude (1000 = 100 Volts) when moving upwards and a smaller amplitude when moving downwards. The exact value of the small amplitude may vary depending on the weight to be lifted. A larger weight usually requires smaller amplitudes in down direction. A good rule of thumb is to chose a value where the up and down movements of a fixed travel distance take about the same time to execute.

As an example you would call

```
SPA0A1F1000B500
```

to use an amplitude of 50 Volts for the down direction. If the positioner is mounted the other way around, call

```
SPA0A1F500B1000
```

instead.

Note that the vertical mode is not necessarily needed for all z-axes. If the weight that is to be lifted is small compared to the capability of the positioner, then it is generally sufficient to leave the default horizontal mode unchanged.

Also note that the alignment setting does not affect open-loop control of the positioner, since you already have direct control of the used step amplitudes.

# 5 Appendix

## 5.1 Movement Status Codes

The table below lists the movement status codes of positioners that are returned by the M command.

| Code | Description |
|---|---|
| S | Stopped - The positioner is currently not performing active movement (see S-command). |
| A | Amplitude setting - The amplitude for an open-loop movement command (U and D) is set. |
| M | Moving - The positioner is performing a (open-loop) movement (see movement commands: U, D). |
| T | Targeting - The positioner is executing a closed-loop movement towards a given target position (see MPA, MPR, MES commands). |
| H | Holding - The positioner is holding its current target position (see closed-loop commands: MPA and MPR). |
| C | Calibrating - The positioner is busy calibrating its sensor (see CS command). |
| R | Referencing - The positioner is moving to a reference position (see MTR command). |

## 5.2 Error Codes

An error answer string has the general format E<code>. The table below lists the error codes and their meanings.

| Code | Meaning<br>Description |
|---|---|
| 0 | No error<br>This indicates that no error occurred and therefore corresponds to an acknowledge. |
| 1 | Parse error<br>The command could not be processed due to a parse error. |
| 2 | Unknown command error<br>The command given is not known to the system. |
| 3 | Invalid channel error<br>The channel index given is invalid and the command cannot be processed. |
| 4 | Invalid mode error<br>The parameter that defines the mode for automatic error reporting is not valid, such that the mode change cannot be processed. |
| 13 | Syntax error<br>The command could not be processed due to a syntax error. |
| 15 | Overflow error<br>A number value given was too large to be processed. |
| 17 | Invalid parameter error<br>A parameter that was given with the command was invalid. |
| 18 | Missing parameter error<br>A parameter was omitted where it was required. |
| 19 | No Sensor Present Error<br>This error occurs if a command was given that requires sensor feedback, but the addressed positioner has none attached. |
| 20 | Wrong Sensor Type Error<br>Some commands are only executable for certain sensor types. For example, issuing an MAA command to a linear positioner leads to this error. |

## 5.3 Sensor Types

| Code | Positioner Type<br>Description |
|---|---|
| 1 | M<br>Standard linear positioner. |
| 2 | GA<br>Goniometer with a radius of 43.5mm. |
| 3 | GB<br>Goniometer with a radius of 56mm. |
| 4 | GC<br>Rotary positioner with end stops. 85mm radius. |
| 5 | GD<br>Goniometer with a radius of 60.5mm. |
| 6 | GE<br>Goniometer with a radius of 77.5mm. |
| 7 | RA<br>Rotary positioner with absolute sensory. |
| 8 | GF<br>Rotary positioner with end stops. Type SR1209m. |
| 9 | RB<br>Rotary positioner. Type SR1910m. |