MCS - Modular Control System
ASCII Programming Interface

SmarAct GmbH
Schuette-Lanz-Strasse 9
D-26135 Oldenburg

Tel.: +49 (0) 441 8008 79-0
Fax: +49 (0) 441 8008 79-21

eMail: info@smaract.de
www.smaract.de

Document Version 14-10-09-357
Interface Version 1.2.44

Table of Contents

1 Introduction

This document describes the ASCII command set that is used for programming of Modular Control Systems (MCS) that understand the MCS ASCII protocol.

Each MCS has a maximum number of channels (see `GNC`). Commands that are directed to a specific channel require a channel index to address the selected channel. The channel indexes are zero based. Note that the number of channels is constant for a given system and describes the number of positioners and/or end effectors that may be connected to the system and **not** the number that currently are connected to the system.

Channels are divided into two types: positioner channels and end effector channels. Each channel can control a single positioner or end effector, depending on its type. Use the `GCT` command to check which type a channel is of. Some commands are only applicable to positioner channels while others may only be sent to end effector channels. Please refer to the Command Description for details of specific commands.

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 colon character followed by a line feed character) are ignored. All characters between a line feed and a colon are also ignored.

Generally, command strings have the following format:

`<command name>[param][,param]...`

The command name is a combination of uppercase letters. Parameters are given as decimal values and may be positive or negative. The number of parameters that are required and also their valid ranges depend on the command given. White spaces are not allowed. See section 3 “Command Description” for more information.

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.

Answer strings have the same format - a combination of uppercase letters and optional parameters.

If a command could not be executed for some reason, an error answer string is returned in the format `E<sourceChannel>,<errorCode>`. The <sourceChannel> indicates which channel of the system generated the error. The value is zero based. If the value of <sourceChannel> is -1, this indicates that the error does not originate from a specific channel, but rather from the overall system. An <errorCode> of 0 indicates that the command was successful and therefore corresponds to an acknowledge. See appendix 4.2 “Error Codes” for a list of error codes.

The command and answer strings are described in more detail in section 3 “Command Description”. Note that in these descriptions the preceding colon and trailing line feed characters of command and answer strings are omitted. Also, the answer strings that are stated assume that no error occurred.

1.2 RS232 Interface

Some notes regarding MCS with RS232 interface:

After a power-up the system will initialize itself. As soon as the TX line of the system is logic 1 it is 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 9600. The baud rate may be configured using the `CB` command.

2 Functional Documentation

Besides basic features like moving and stopping positioners the MCS offers many features that are activated or configured via additional commands. You may refer to the Command Description on how to use these commands, parameter ranges etc. However, some features require more information for a better understanding. This chapter explains these features in more detail and describes some basic concepts of the MCS controller.

2.1 Communication Modes

The MCS supports two different communication modes: synchronous and asynchronous communication.

- In sync mode it is assured that every command sent to the system generates one answer. If no real information is to be returned, the system generates a simple acknowledge to indicate that the command was successful.
- In async mode acknowledges are omitted to reduce communication overhead. For example a movement command - assuming that no error occurred - will be "silently" executed.

The mode that is used may be configured with the `SCM` command.

Note that some commands are only available in asynchronous mode (e.g. `SRC`). Executing these commands in synchronous mode will generate a "wrong mode" error (code 8).

Configuring the communication mode is one of the first things you should do after connecting to the system. The default mode after a power-up is the synchronous mode.

2.2 Channel Properties

Each channel of an MCS controller has various properties that affect the behavior of the channel. These can be global parameters, operation modes etc. To manipulate these properties you may use the commands `GCP` and `SCP`. When issuing these commands you must supply a property key to indicate which property you wish to read or write. A property key is a 32-bit code that refers to a property. Please see the appendix (4.4 "Channel Properties") for a list of property key codes.

An MCS channel contains several functional components that may be further divided into several sub components. Each of these (sub) components may have several properties. The general properties are described in the following sections.

2.2.1 Emergency Stop

Note: This feature is not available on all MCS controllers. Please contact SmarAct for more information.

The MCS controller may be equipped with a dedicated hardware TTL input signal that is used as an emergency stop. A negative pulse on this line (*ES line*) may cause channels of the system to stop immediately. The operation mode property of the Emergency Stop component controls the behavior of a channel in case of such an emergency.

There are four modes available:

- 0 (Normal): This is the default mode. In this mode a falling edge on the ES line has the same effect as issuing a single stop command (`S`). After such an event the system continues to behave normally.
- 1 (Restricted): In this mode a falling edge on the ES line will issue a stop and make the channel enter a locked state. In this state you may communicate with the channel normally, but all movement commands will return an error 146 (movement locked). The locked state may be reset by setting the operation mode to any valid value, thereby unlocking the movement again.
- 2 (Disabled): In this mode falling edges on the ES line are simply ignored.
- 3 (Auto Release): In this mode a falling edge on the ES line will issue a stop and make the channel

enter a locked state. In this state you may communicate with the channel normally, but all movement commands will return an error 146 (movement locked). This state remains until the channel detects a rising edge on the ES line or the operation mode is set to any valid value.

Note that the ES line will be also triggered internally when the RS232 cable is unplugged.

The default mode of the Emergency Stop feature can be changed with the Channel Property Key `16842797` (see p. 88).

2.2.2 Low Vibration Mode

SmarAct's positioners allow macroscopic movement while still offering ultra-high precision on the nanometer scale. However, the stick-slip driving principle is accompanied by high-frequent vibrations that may cause trouble in some applications.

For this the MCS offers a special operation mode in which movement commands are executed to produce as little vibrations as possible. To activate this mode simply set its operation mode property to 1 (Enabled). E.g. for channel 0 you would send

```
SCP0,16908289,1
```

Note that the Low Vibration mode requires the acceleration control feature to be active (see `SCLA`). Enabling the Low Vibration mode while acceleration control is inactive will cause the acceleration control to be implicitly activated with a default value.

Also note that a channel must be completely stopped (status code 0) in order to be able to change the Low Vibration operation mode. See also section 2.5.1 “Dependency Chains“.

Note: This feature is not available on all MCS controllers. Please contact SmarAct for more information.

2.2.3 Broadcast Stop

This feature can trigger an Emergency Stop on the MCS controller when an end stop is detected. It is typically useful when multiple channels are moving simultaneously and an end stop on one channel should cause a halt on all other channels. Please note that a channel whose Emergency Stop Mode is set to 2 (disabled) is not affected by this trigger.

2.2.4 Position Control

This sub component is used to change parameters which are associated with a channels movement controller. Currently there is only one property available.

Forced Slip: When reaching a target position, e.g. after a move absolute command was issued, the channel will try to stop at approx. 50% of its step size, thus improving the holding feature. If this behavior is unwanted it can be disabled with this channel property. This feature is enabled by default.

2.2.5 Sensor

This sub component controls the channel parameters which specify the handling of the sensor. Please note that this features are not available on channels without sensor.

Power Supply

This property selects if the sensor should be enabled, disabled or in power save mode, for more details on the operation modes refer to section 2.3.2 Sensor Modes. Whereas the command `SSE` (Set Sensor Enabled) sets the property for all channels of an MCS Controller (see p. 50), this feature can be used to modify the operation mode of a single channel.

Scale

This sub component allows an alternative access to the scale settings of the channel. For more details on this functionality refer to paragraph “Shifting the Measuring Scale“ on p.9.

2.3 Working With Sensor Feedback

This section covers some features of the MCS when using positioners with integrated sensors and explains them in more detail.

2.3.1 Rotary Sensors

In contrast to linear sensors, where the position is simply given by a single signed value, rotary sensors are handled a little differently.

Suppose a rotary positioner is currently aligned to a 45° angle and shall be instructed to move to 90°. This can be accomplished in two ways: clockwise or counter-clockwise. To eliminate such ambiguities, a rotary position is defined by a combination of an angle and a revolution. The angle value is given in micro degrees and has a valid range of 0..359,999,999. The revolution value indicates complete 360° rotations of the positioner and has a valid range of -32,768..32,767.

If a rotary positioner starting at zero (angle = 0; revolution = 0) moves in positive direction, the angle value will increase, reflecting the current orientation of the positioner. If the positioner moves further over the 360° boundary, the angle value will wrap around to zero and the revolution value will be incremented by 1 to indicate one full rotation of the positioner. The reverse direction is done accordingly.

Consequently, when issuing absolute movement commands (`MAA`), the movement direction is implicitly defined by the parameters given. In the example above (moving from 45° to 90°) the direction would be distinguished by specifying 0 or -1 as the revolution parameter (assuming the current revolution is 0).

Note that the valid range of the `MAR` command is extended in the negative range to -359,999,999..359,999,999. This is simply for convenience. For example, the following commands have the same effect:

```
MAR0,-90000000,0,0
MAR0,270000000,-1,0
```

Both commands will move the positioner 90° in negative direction.

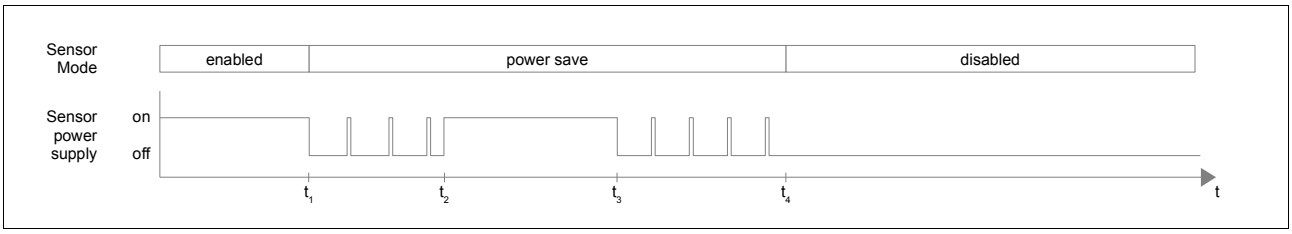
2.3.2 Sensor Modes

In order for a positioner to track its position, its sensor needs to be supplied with power. However, since this generates heat (causing drift effects), it might be desirable to disable the sensors in some situations (especially in temperature critical environments). For this, there are three different modes of operation for the sensor, which may be configured with the `SSE` command.

- *0 (Disabled)* - In this mode the power supply of the sensor is turned off. This avoids the generation of heat. Closed-loop commands such as `MPA` will not be executed, but rather an error returned (code 140) informing about the sensor state. This mode may also be useful if the light that is emitted by the sensors interferes with other components of your setup (e.g. detectors inside an SEM chamber). **Note:** Open-loop commands such as `MST` are still executed. This implies that the position information will become invalid, since the positioner cannot track its position during the movement. It is the users responsibility to enable the sensors again before moving the positioner, should the position tracking be needed. Be aware that position calculation is done incrementally. So even after turning on the sensors again the position will be invalid if the positioner was moved while the sensor was offline.
- *1 (Enabled)* - In this mode the sensor is supplied with power continuously. All movement commands are executed normally.
- *2 (Power Save)* - If set to this mode the power supply of the sensor will be handled by the system automatically. If the positioner is idle the sensor will be offline most of the time, avoiding unnecessary heat generation. A movement command (open-loop or closed-loop) will cause the

system to activate the sensor before the movement is started. Since it takes a few milliseconds to power up the sensor, the movement will be delayed during this time and the positioners will have a movement status code of 5.

The figure below illustrates the different sensor modes and shows when the sensors are supplied with power.



In this example the sensor mode is initially set to *enabled*. The sensors are continuously supplied with power. At time t_1 the sensor mode is switched to *power save*. In this mode the system starts to pulse the power supply of the sensors to keep the heat generation low. At time t_2 a movement command is issued, which requires the sensors to be online in order to keep track of the current position. Note that the sensor mode stays unchanged during this time. As soon as the movement has finished (t_3) the system will start to pulse the power supply again. At time t_4 the sensor mode is switched to *disabled*, in which the power supply is turned off continuously. If movement commands are issued during this time then the system will not be able to track the position. The position data will become invalid.

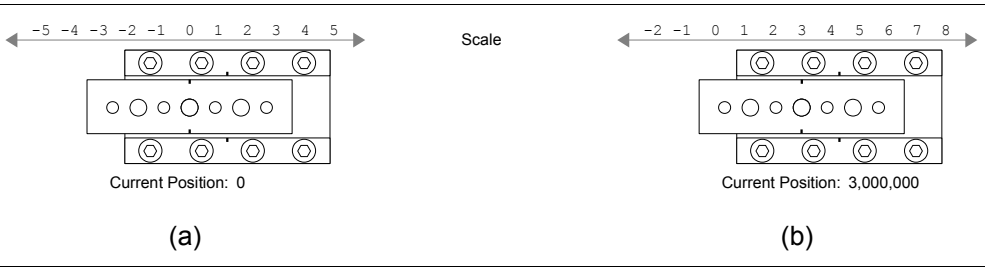
Note on the power save mode: If closed-loop commands are issued with a hold time, then the system will start to pulse the power supply of the sensor as soon as the target position has been reached. At this point the hold time starts. The positioner will still hold the target position and compensate for drift effects while pulsing, although it might not be as accurate as in the *enabled* mode.

2.3.3 Defining Positions

Since the position calculation is done on an incremental basis, a positioner has no way of knowing its physical position after a system power-up. It simply assumes its starting position as the zero position.

However, in many applications it is convenient to define a certain physical position as the zero position. The `SP` command may be used for this purpose. It defines the current position to have an arbitrary value. This can be the zero position or any other position, which makes it possible to have the zero position outside the complete travel range of the positioner.

The figure below shows an example of a linear positioner. (a) shows the situation after a system power-up. The positioner assumes its current position as zero. (b) shows the situation after sending `SP0,3000000`. The current position has been defined to +3mm and the measuring scale is shifted accordingly.



Reference Marks

In the example above the physical position of a positioner has to be determined by some external method and then configured to the system. Moreover, this procedure must be done on every system power-up.

To overcome this inconvenience the `FRM` command may be used to move a positioner to a known physical position in an automated fashion. Depending on the product option of your positioner it may be equipped with a single reference mark or with multiple reference marks. Some positioners do not have a physical reference mark, but are rather referenced via a mechanical end stop. The different search algorithms are outlined in the following:

- Single Reference Mark: In this case the reference mark (which is usually located near the middle of the travel range) is used to determine the physical position. The positioner starts to move in the given initial direction. As soon as the reference mark has been detected the positioner stops and the search is successful. Should the positioner detect an end stop then the search direction is reversed and the search is continued. If a second end stop is detected before the reference mark is found the search will abort unsuccessfully. (When using the asynchronous communication mode an error will be generated.)
- Distance Coded Reference Marks: In this case the distance between any two neighboring reference marks is measured in order to determine the physical position. The positioner starts to move in the given initial direction. When the first reference mark has been detected then the current position value is stored and the search continues in the same direction for a second mark. When the second reference mark has been detected then the positioner stops and the search is successful. The distance between the two reference marks is calculated to determine the physical position. Should the positioner detect an end stop then the search direction is reversed and the process is repeated. If a second end stop is detected before two reference marks have been found the search will abort unsuccessfully. (When using the asynchronous communication mode an error will be generated.)
- End Stops: In this case a mechanical end stop is used as a known physical position. The positioner will move in the safe direction (see `SSD`) until it detects an end stop. The sensor signals are then used to align the position to the reference position with high repeat accuracy. Note that the configured end stop must be calibrated with `CS` before it can be properly used as a reference point.

When the command has completed successfully the system knows the physical position of the positioner (see also `GPPK`).

Shifting the Measuring Scale

The *physical* measuring scale is fix for each positioner (see previous section) and cannot be changed. However, the MCS controller uses a *logical* measuring scale when calculating positions to result in position values returned by `GP` (or `GA`). The logical measuring scale may be shifted or inverted by the user so that the controller returns a desired value at a certain physical position.

The relation between the physical and the logical scale is defined by two parameters. The offset value, which represents the shift, and the inversion value, which inverts the count direction, of the logical scale relative to the physical scale. The default value of the offset and the inversion is zero which makes the physical and the logical scale identical.

There are two methods to modify the offset value:

- Sending a `SP` command sets the offset implicitly by shifting the logical scale so that the current position equals the desired value (see example below).
- Sending a `SSC` command sets the offset explicitly and the current position will have a value that reflects the new scale shift and direction.

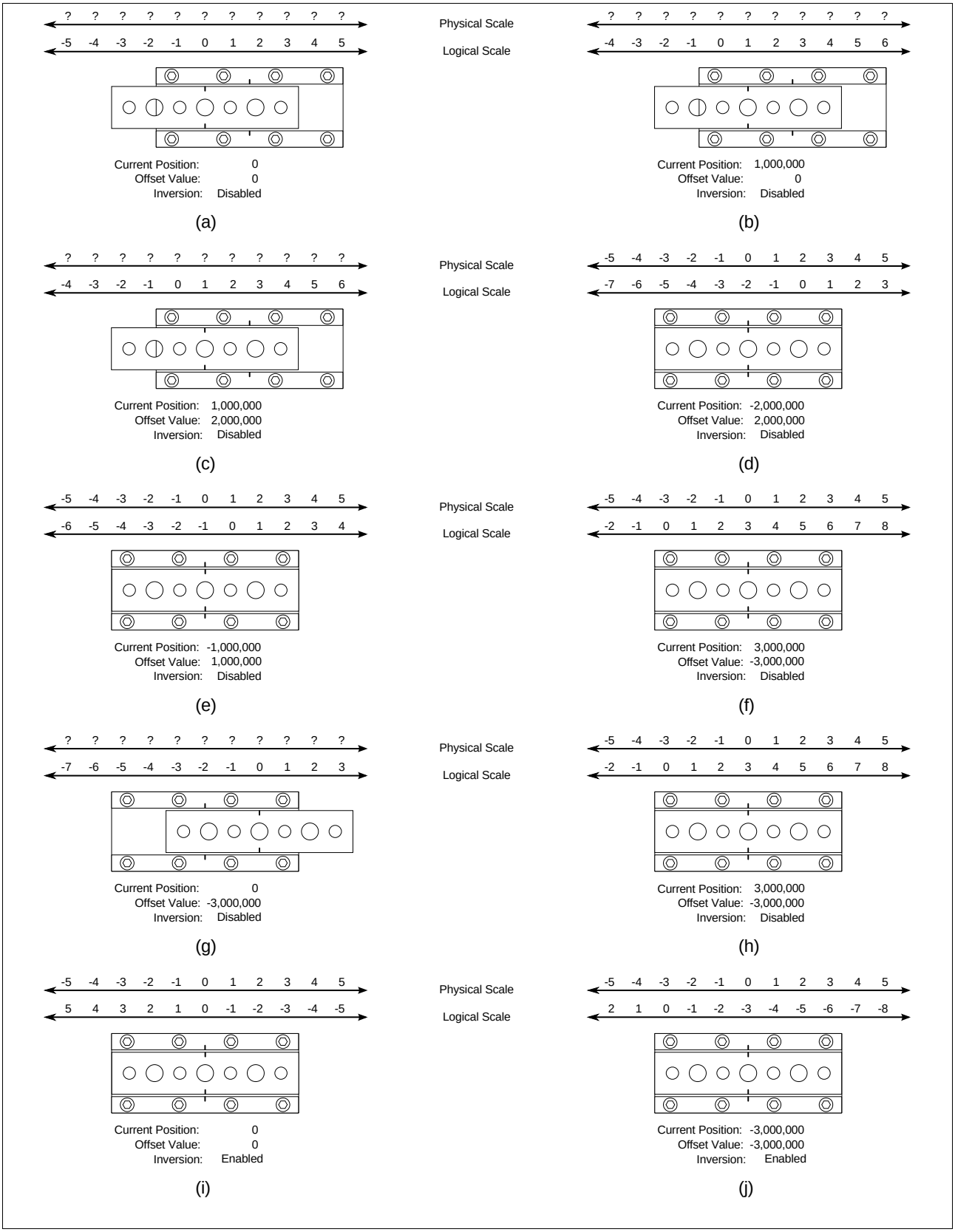
The offset and inversion value is stored in non-volatile memory. Once it is configured you only need to send a `FRM` to restore your settings on future power-ups.

Note: The behavior of the system when sending `SP` or `SSC` differs slightly depending on whether the physical position is known or not (see `GPPK`). When the physical position is unknown sending a `SP` will not update the offset value in the non-volatile memory. Likewise, sending a `SSC` will have no immediate effect on the values returned by `GP`. The following table summarizes the behavior.

	physical position is known		physical position is unknown	
	SSC	SP	SSC	SP
offset value is written to non-volatile memory	yes	yes	yes	no
command has immediate effect on position values	yes	yes	no	yes

Example

To further demonstrate the behavior of the system in different situations the figure below shows an example of an SLC positioner with a single reference mark. The small markings indicate the location of the reference mark. When the markings overlap the positioner is on the mark. The code section below shows the corresponding command sequence.



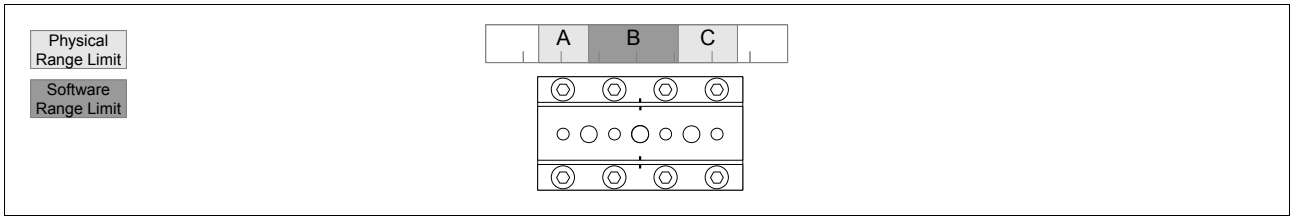
```
// (a) - The system is powered up. The physical position is unknown and the
//       current position is assumed to be 0. The offset value is in the default
//       setting.
SP0,1000000
// (b) - The current position has been set to +1mm. Since the physical position is
//       unknown, the offset value in the non-volatile memory could not be updated
//       implicitly.
SSC0,2000000,0
// (c) - The offset value in the non-volatile memory has been set to +2mm. Since
//       the physical position is unknown, the current position could not
//       be updated implicitly.
FRM0,0,60000,0
// (d) - The positioner has moved to the reference mark. The physical position
//       is now known and the position value has been updated to reflect the
//       configured offset between the physical and the logical scale.
SP0,-1000000
// (e) - The current position has been set to -1mm. Since the physical position is
//       known, the offset value in the non-volatile memory was updated implicitly.
SSC0,-3000000,0
// (f) - The offset value in the non-volatile memory has been set to -3mm. Since
//       the physical position is known, the current position was updated implicitly.
// (g) - The system was shut down, the positioner was moved externally to some
//       random location and the system is then powered up again. The physical
//       position is unknown and the positioner assumes its current position as 0
//       again.
FRM0,1,60000,0
// (h) - The positioner has moved to the reference mark. The physical position
//       is now known and the position value has been updated to reflect the
//       configured offset between the physical and the logical scale.
SSC0,0,1
// (i) - The offset and inversion value in the non-volatile memory have been set to 0mm
//       and SA_TRUE. Thus the counting direction of the scale got inverted. Since
//       the physical position is known, the current position was updated implicitly.
SSC0,-3000000,1
// (j) - The offset value was changed and both values were stored inside the non-volatile
//       memory. Since the physical position is known, the current position is updated
//       implicitly. Notice the difference between (i) and (h).
```

2.3.4 Software Range Limit

While linear positioners have a limited physical travel range it might be useful to further limit this range if the positioner must not be allowed to move beyond a certain point. Rotary positioners usually have no physical end stops, but e.g. wiring may require to limit the rotation here as well.

For these situations the MCS offers to limit the travel range of a positioner by software. The commands `SPL`, `GPL`, `SAL` and `GAL` offer control over this feature.

By default no range limit is set. Once it is defined the positioner will not move beyond the boundaries of the range limit. This affects all movements except scan movements (e.g. `MSCA`). If a movement command is issued that would move the positioner beyond the defined limit then the positioner is stopped. (When using the asynchronous communication mode an error will be generated.) Further movements are only allowed if they move the positioner in the direction pointing back inside the range limit. This also applies if the positioner has been moved outside the defined range limit by external means. The figure below shows an example of a linear positioner. The positioner is limited by software to move only within zone B. Should the positioner somehow be pushed into zone A it will only be allowed to move to the right towards zone B. The same applies to zone C accordingly.

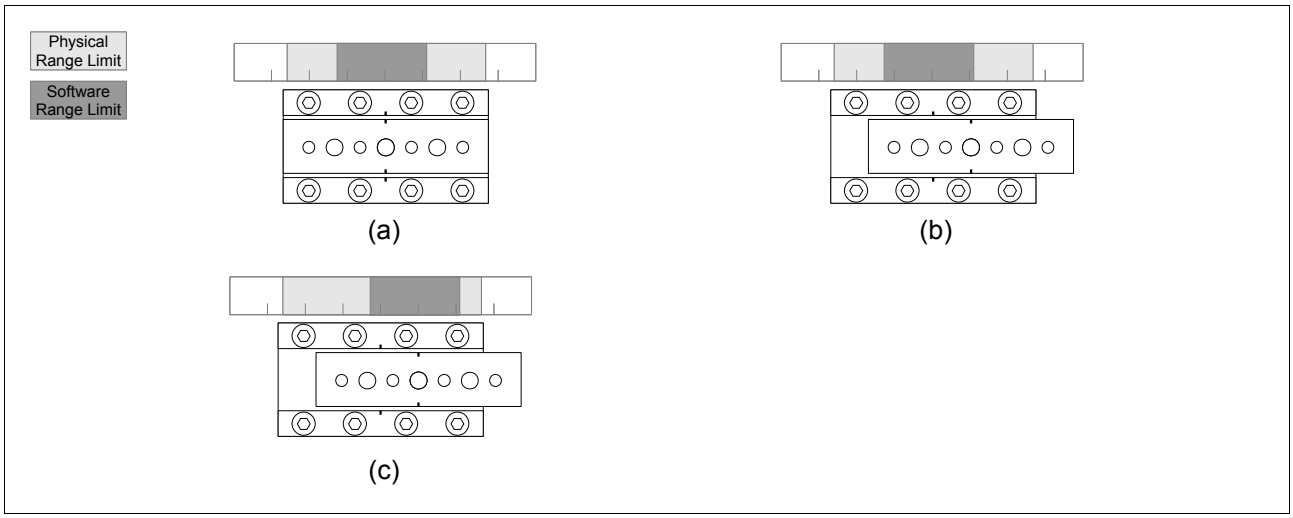


Please note the following restrictions:

- The range limit may only be set if the positioner “knows” its physical position, i.e. after the reference mark has been found (see `FRM`). The command `GPPK` may be used to check this special state.
- The range limit is not saved to non-volatile memory and must be configured in each session. Typically, after a system power-up you would send the commands `FRM` and then `SPL`.
- The range limit has a limited accuracy. The positioner may pass over the boundary by a few micro meters resp. milli degrees. Therefore, the range should be defined with sufficient tolerance.

The software range limit has some consequences that you might want to consider.

- Both the minimum and maximum position of the range limit behave similarly to a physical end stop. For example, the `FRM` command will reverse its movement direction while looking for the reference mark if a range limit boundary is reached. If the reference mark is located outside the range limit then it will not be found.
  You should also avoid sending a `CS` command while near a range limit boundary. Otherwise the calibration will be aborted.
- Sending an `SP` command does not automatically adjust the software range limit accordingly. This means that shifting the measurement scale of the positioner with this command will also shift the physical position of the software range limit. This is illustrated in the figure below. Suppose the positioner is currently at position 0. A software range limit is defined as indicated by the dark gray area in (a). In (b) the positioner has moved one unit to the right. After this the current position is set to zero again (with `SP`) as shown in (c). As a result, the physical position of the software range limit has moved to another location, which enables the positioner to move beyond the boundary that was defined in (a). Therefore, care should be taken when working with these commands.



2.4 Controller Event System

Each positioner channel of an MCS includes an event system that may be configured flexibly to trigger various internal features. Generally you can think of components that can generate events (event sources) and components that may be configured to receive events (event receivers). The latter ones may trigger actions when an event occurs.

The table below lists the currently available event sources and event receivers.

Component	Event Source	Event Receiver
Digital In	x	
Software Trigger	x	
Counter		x
Capture Buffer		x
Command Queue		x

Event sources and receivers must be configured with the `SCP` command (see also section 2.2 “Channel Properties”). Event sources must be enabled (as they are disabled by default) and usually given a parameter to tell them under which conditions to generate an event. Event receivers must be “connected” to an event source (by default no source is configured) to trigger its functionality.

Note that it is insufficient to only configure an event source. If no receiver component is configured to receive the events, then the functionality of the receiver component will not trigger.

To “connect” an event receiver component with an event source you must set its trigger source property. The value of a trigger source property is a selector value which is (similar to property keys) a 32-bit code that refers to an event source. Please see the appendix 4.4.1 “Trigger Source Codes“ for a list of trigger source codes.

To “disconnect” an event receiver component from an event source simply set its trigger source property to 0 (Disabled).

Note: The Command Queue cannot be configured with a global trigger source (it has no trigger source property). Rather the commands in the queue are each given an individual trigger source when sending the `ATC` command. The *triggerSource* parameter of this command is used in the same way as a trigger source component property, except that a value of 0 (Disabled) is not allowed.

The components that generate and receive events are described in more detail in the following sections.

2.4.1 Digital Inputs

A digital input is an external TTL input line that is connected to a channel of the MCS controller. It may be used to generate internal events which may trigger other components.

Note: This feature is not available on all MCS controllers. Please contact SmarAct for more information.

A digital input's sub component is an index value which selects a specific input. Currently there are two inputs available, so the index value must be either 0 or 1.

Digital inputs have two component properties:

- 1 (Operation Mode) (Read/Write): Can be 0 (Disabled, default) or 1 (Enabled). Enables or disables event generation. If disabled, event receivers connected to this event source will not receive any events and therefore not trigger their functionality.
- 2 (Active Edge) (Read/Write): Can be 0 (Falling Edge, default) or 1 (Rising Edge). Determines on which edge events are generated.

Note that the active edge property value has no effect if the operation mode is disabled. It is recommended to configure the active edge before enabling the input.

The table below lists the electrical specification of the digital inputs.

	Input 0		Input 1		
Parameter	Minimum	Maximum	Minimum	Maximum	Unit
Pulse Frequency		1000		1000	Hz
Pulse Width	0.1		100		µs
Input Low Voltage	-0.5	1.5	-0.5	1.5	V
Input High Voltage	3	5.5	3	5.5	V

2.4.2 Software Triggers

A software trigger is an event source that is triggered when the `TC` command is sent and is typically used in combination with command queuing to synchronize movement commands of several channels (see 2.4.5 “Command Queues“ and the example there).

A software trigger's sub component is an index which may be used to distinguish between different triggers (although one trigger is sufficient in most cases, so the index is typically zero).

The software trigger has no component properties and therefore cannot be configured. It is always implicitly enabled.

2.4.3 Counters

Counters are used to count events. They may be configured with an event source to increment the counter value every time an event is received.

A counter's sub component is an index value which selects a specific counter. Currently there is only one counter available, so the index value must be 0.

Counters have two component properties:

- 3 (Trigger Source) (Read/Write): Selects an event source to trigger the counter. The value of this property is a selector value. See section 2.4 “Controller Event System“.
- 5 (Value) (Read/Write): Reading this property returns the current counter value. You may also write this property, e.g. to reset the counter to zero or set an arbitrary starting value.
  The valid range is 0 .. 2,147,486,647.

2.4.4 Capture Buffers

When reading feedback data from a channel you are normally forced to read multiple data values consecutively which generates a time gap between the queries. However, there might be situations where you wish to synchronize the data.

Capture buffers take snapshots of groups of internal values when events occur. Reading out the capture buffer contents therefore enables you to perform an atomic read operation on multiple data values (see `GB`).

The type of data that a capture buffer holds cannot be configured. You may only choose a buffer via its index to capture predefined value combinations.

The following table lists the currently defined capture buffers and the data that they capture.

Buffer Index	Parameter 1	Parameter 2	Parameter 3
0	position	revolution	counter 0

Once an event source is configured for a capture buffer the corresponding internal values are copied to the capture buffer if an event is received. The values in the capture buffer remain unchanged until the next event is received.

Since the content of a capture buffer is defined by its index, capture buffers have only one property:

- 3 (Trigger Source) (Read/Write): Selects an event source to trigger the capture buffer. The value of this property is a selector value. See section 2.4 “Controller Event System“.

2.4.5 Command Queues

Normally when a channel receives a movement command it will be executed immediately. However, it is possible to let the execution be delayed until a certain event occurs. This allows for example to synchronize movements with external processes.

Note: This feature is only available in the asynchronous communication mode.

The command queue is organized as a FIFO queue, i.e. the first command that was appended to the queue is the first one to be executed as soon as the configured event occurs.

To append a command to the command queue you must issue two commands consecutively:

1. Send a `ATC` command and pass an event source that should trigger the command. The event source is coded in form of a selector value (see section 2.4 “Controller Event System“).
2. Send the desired movement command, e.g. `MPA`. Note that calling a non-movement command here will generate an error code 152 (command not triggerable).

The movement command will not be executed immediately, but rather stored in the command queue and executed as soon as it is triggered by the configured event source.

Please note that there is no way to read out the command queue. Therefore, you should keep track of the queued commands and how they are triggered in your software application if necessary.

If there is at least one command in the command queue you will not be able to execute further movement commands until the queue is empty again. You may append more commands to the queue (if not full), but issuing movement commands without previously sending `ATC` will generate an error code 151 (waiting for trigger). The command queue will become empty if all commands in the queue have been triggered. Alternatively, you may clear the queue manually by sending a `CTCQ` command.

A channel may be configured to send a notification when a command has been triggered with the `SRT` command (see there).

Command queues have two properties:

- 4 (Size) (Read only): Indicates how many commands are currently in the queue. The value of this property will be incremented when a command is appended to the queue and decremented when the next command in the queue is triggered. (A queued command that is triggered is removed from the queue. It does not reside in the queue while it is in execution.)
- 6 (Capacity) (Read only): Indicates how many commands the queue is able to store. This value is a constant and currently has a value of 1.

Note that the command queue has no global trigger source property. The commands in the queue each have their own trigger source.

Example

The following command sequence demonstrates how to synchronize the movement of two positioners to minimize the time between the start of their movements.

Command	Note
`SCM1`	Switch to asynchronous communication (needed for `ATC`).
`ATC0,1792`	The next movement command for channel 0 will be queued and triggered later by software (index 0).
`MPR0,1000000,0`	Move channel 0 by +1mm (not executed yet).
`ATC1,1792`	The next movement command for channel 1 will be queued and triggered later by software (index 0).
`MPR1,1000000,0`	Move channel 1 by +1mm (not executed yet).
`TC0`	Trigger both movement commands simultaneously.

2.5 Miscellaneous Topics

2.5.1 Dependency Chains

Some configuration options of a channel depend on other options to be set. Specifically, there is the dependency chain

Speed Control ← Acceleration Control ← Low Vibration Mode.

The arrows represent a “requires” relation. Hence, only the following configuration combinations are valid:

State	Speed Control	Acceleration Control	Low Vibration Mode
1	inactive	inactive	inactive
2	active	inactive	inactive
3	active	active	inactive
4	active	active	active

When activating or deactivating one of these features the system implicitly takes actions to maintain a consistent state. E.g. when activating the low vibration mode while in state 1, the speed control and acceleration control features will be implicitly enabled (with default settings). Likewise, deactivating the speed control while in state 4, will implicitly deactivate the acceleration control and the low vibration mode.

Note that a channel must be completely stopped in order to change the low vibration mode. Otherwise an error code 150 (command not processable) will be generated. This implies that deactivating the speed control may also produce this error, if the low vibration mode cannot be implicitly deactivated.

Note: The Speed Control, Acceleration Control and Low Vibration Modes are not available on all controllers. Please contact SmarAct for more information.

3 Command Description

3.1 Initialization Commands

GCM - Get Communication Mode

Description:

This command may be used to determine the communication mode that is currently configured for the system. Inverse command to `SCM`.

Parameters:

none

Answer String:

`CM<mode>`

The <mode> will be either 0 for synchronous communication or 1 for asynchronous communication.

Example:

`GCM`

Requests the currently configured communication mode of the system.

GCT - Get Channel Type

Description:

This command may be used to determine the type of a channel. Each channel of an MCS has a specific type. Currently there are two types of channels: positioner channels and end effector channels. Most commands are only executable by certain channel types. The command descriptions list for which types they may be called.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
CT<channelIndex>,<type>
```

Example:

```
GCT1
```

Requests the channel type of channel 1 of the system.

GIV - Get Interface Version

Description:

This command may be used to retrieve the interface version of the system. It is useful to check if changes have been made to the software interface. An application may check the version in order to ensure that the system behaves as the application expects it to do.

Parameters:

none

Answer String:

MCS device with RS232 interface:

`IV<versionHigh>,<versionLow>,<versionUpdate>`

MCS device with network interface:

`IV(<firmwareType>)<versionHigh>,<versionLow>,<versionUpdate>`

Example:

`GIV`

Returns the interface version.

GNC - Get Number of Channels

Description:

This command may be used to determine how many control channels are available on a system. This includes positioner channels and end effector channels. Each channel is of a specific type. Use the `GCT` command to determine the types of the channels.

Note that the number of channels does not represent the number positioners and/or end effectors that are currently *connected* to the system.

The channel indexes throughout the interface are zero based. If your system has N channels then the valid range for a channel index is 0.. N-1.

Parameters:

none

Answer String:

`N<channels>`

Example:

`GNC`

Returns the number of channels.

GSI - Get System Id

Description:

This command may be used to physically identify a system connected to the PC. Each system has a unique ID which makes it possible to distinguish one from another.

The ID returned is a generic decimal number that uniquely identifies the system.

Parameters:

none

Answer String:

`ID<id>`

Example:

`GSI`

Returns the system ID.

R - Reset

Description:

When this command is sent the system will perform a reset. It has the same effect as a power down/power up cycle. The system replies with an acknowledge string before resetting itself.

Parameters:

none

Answer String:

```
E-1,0
```

Example:

```
R
```

Performs a system reset.

SCM - Set Communication Mode

Description:

The `SCM` command lets you select between the communication modes. The mode is system global and should be called at initialization. The default mode after power-up is synchronous.

Parameters:

- *mode* - must be either 0 (synchronous mode) or 1 (asynchronous mode)

Answer String:

The answer behavior is that of the new mode, i.e. in sync mode an acknowledge will be returned, in async mode it will be omitted.

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SCM1
```

Configures the system for the asynchronous communication mode. No answer will be returned.

SHE - Set HCM Enabled

Description:

If a Hand Control Module (HCM) is connected to the system, this command may be used to enable or disable it in order to avoid interference while the software is in control of the system. There are three possible modes to set:

- `0`: In this mode the Hand Control Module is disabled. It may not be used to control positioners.
- `1`: This is the default setting where the Hand Control Module may be used to control the positioners.
- `2`: In this mode the Hand Control Module cannot be used to control the positioners. However, if there are positioners with sensors attached, their position data will still be displayed.

Parameters:

- *mode* - Must be 0, 1 or 2.

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SHE0
```

Disables the Hand Control Module.

3.2 Configuration Commands

GAL – Get Angle Limit

Channel Type: Positioner

Description:

Inverse command to `SAL` (see there). May be used to read out the travel range limit that is currently configured for a rotary channel.

See section 2.3.1 “Rotary Sensors” for information on rotary positions.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
AL<channelIndex>,<minAngle>,<minRevolution>,<maxAngle>,<maxRevolution>
```

See the `SAL` command for a description of the parameters.

Example:

```
GAL0
```

Requests the travel range limit that is currently configured for channel 0.

See also: `SAL`, `GPL`, `SPL`

GCLA - Get Closed Loop Acceleration

Channel Type: Positioner

Description:

Inverse command to `SCLA` (see there). Returns the currently configured movement acceleration that is used for closed-loop commands for a channel.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
CLA<channelIndex>,<acceleration>
```

The <acceleration> is given in µm/s² for linear positioners and in m°/s² for rotary positioners. A value of 0 means that the acceleration control feature is deactivated.

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
GCLA0
```

Requests the closed-loop movement acceleration that is currently configured for channel 0.

See also: `SCLA`

GCLS - Get Closed Loop move Speed

Channel Type: Positioner

Description:

Inverse command to `SCLS` (see there). Returns the currently configured movement speed that is used for closed-loop commands for a channel.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
CLS<channelIndex>,<speed>
```

The <speed> is given in nm/s for linear positioners and in µ°/s for rotary positioners. A value of 0 means that the speed control feature is deactivated.

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
GCLS0
```

Requests the closed-loop movement speed that is currently configured for channel 0.

See also: `SCLS`

GCP - Get Channel Property

Channel Type: Positioner

Description:

Inverse command to `SCP` (see there). Retrieves a configuration value from a channel. This is a universal command to read out various channel properties. The property which is to be read is selected via the key parameter. This 32-bit parameter codes a combination of values and has the following structure:

31 24	23 16	15 8	7 0
component	sub component	property	

See the appendix 4.4 “Channel Properties“ for a list of component selectors and properties. The sub component selector is usually an index, but there can also be special sub component selectors. Note that not all properties are valid for all components. Please refer to section 2.2 “Channel Properties” for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *key* - Selects the property of which the value should be read.

Answer String:

```
CPR<channelIndex>,<key>,<value>
```

Example:

```
GCP0,100663300
```

Requests the current command queue size of channel 0.

See also: `SCP`

GEET - Get End Effector Type

Channel Type: End effector

Description:

Inverse command to `SEET` (see there). Returns the current end effector configuration of an end effector channel.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
EET<channelIndex>,<type>,<param1>,<param2>
```

Please refer to the table on page 42 for a description of the types and their parameters.

Example:

```
GEET0
```

Requests the end effector configuration for channel 0.

See also: `SEET`

GESM - Get Emergency Stop Mode

Please note that this command is deprecated. Get the according channel property with `GCP` instead.

Description:

Inverse command to `SESM` (see there). Returns the emergency stop mode that is currently configured for the system.

Parameters:

none

Answer String:

`ESM<mode>`

See the `SESM` command for a description of the different modes.

Example:

`GESM`

Requests the currently configured emergency stop mode.

See also: `SESM`

GPL - Get Position Limit

Channel Type: Positioner

Description:

Inverse command to `SPL` (see there). May be used to read out the travel range limit that is currently configured for a linear channel.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
PL<channelIndex>,<minPosition>,<maxPosition>
```

<minPosition> and <maxPosition> are given in nm.

Example:

```
GPL0
```

Requests the travel range limit that is currently configured for channel 0.

See also: `SPL`, `GAL`, `SAL`

GSC - Get Scale

Channel Type: Positioner

Description:

Inverse command to `SSC` (see there). May be used to read out the currently configured scale of a channel.

Parameters:

- channelIndex - Selects the channel of the system. The index is zero based.

Answer String:

```
SC<channelIndex>,<scale>,<inverted>
```

<scale> is the currently configured scale.

<inverted> will be either 0 (disabled) or 1 (enabled).

Caution: Please note that only the logical scale of the positioner will be inverted when the `inverted` value has changed. Parameters like the *SafeDirection* will not be altered. Thus the positioner will move in the opposite direction when e.g. calling `FRM` with the same parameters prior to the inversion change.

Example:

```
GSC0
```

Requests the scale that is currently configured for channel 0.

See also: `SSC`

GSD - Get Safe Direction

Channel Type: Positioner

Description:

Inverse command to `SSD` (see there). May be used to read out the currently configured safe direction for a channel.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
SD<channelIndex>,<direction>
```

<direction> will be either 0 (forward) or 1 (backward).

Example:

```
GSD0
```

Requests the safe direction that is currently configured for channel 0.

See also: `SSD`, `FRM`, `CS`

GSE - Get Sensor Enabled

Description:

Inverse command to `SSE`. It may be used to read the sensor operation mode that is currently configured for the sensors that are attached to the positioners of the system. The mode is system global and applies to all channels of a system equally.

Please refer to section 2.3.2 “Sensor Modes” for more information on the sensor modes.

Parameters:

none

Answer String:

`SE<mode>`

The <mode> is either 0 (disabled), 1 (enabled) or 2 (power save).

Example:

`GSE`

Requests the current sensor mode.

See also: `SSE`

GST - Get Sensor Type

Channel Type: Positioner

Description:

Returns the type of sensor that is configured for the given channel (see `SST`). The returned type will be one of the types listed in the appendix (4.3 “Sensor Types”).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
ST<channelIndex>,<typeCode>
```

The <typeCode> is one of the codes listed in the table on page 87.

Example:

```
GST1
```

Requests the sensor type that is currently configured for channel 1 of the system.

See also: `SST`

SARP - Set Accumulate Relative Positions

Channel Type: Positioner

Description:

This command is of interest in conjunction with the closed-loop commands `MPR` and `MAR` (see there). It sets a flag that affects the behavior of a positioner if a relative position command is issued before a previous one has finished. If relative position commands are to be accumulated all new relative position commands are added to the previous target position. Otherwise the movement is executed relative to the position of the positioner at the time of command arrival.

Example: Say the positioner is currently at its zero position. Two relative movement commands are issued in fast succession both with +1mm as relative target. With accumulation active the final position will be 2mm. With accumulation inactive the final position will vary (e.g. 1.12mm) depending on when the second command arrives.

By default relative position targets are accumulated.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *accumulate* - Must be either 0 (no accumulation) or 1 (accumulation).

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SARP2,1
```

Configures channel 2 of the system to accumulate relative target positions.

See also: `MPR`, `MAR`

SAL - Set Angle Limit

Channel Type: Positioner

Description:

For positioners with integrated sensors this command may be used to limit the travel range of a rotary positioner by software. (For linear positioners see `SPL`.) By default there is no limit set. If defined the positioner will not move beyond the limit. This affects open-loop as well as closed-loop movements.
Note that the limit may only be set if the physical position is known at the time of the call (see `FRM`, `GPPK`).

See section 2.3.1 “Rotary Sensors“ and 2.3.4 “Software Range Limit“ for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *minAngle* - Absolute minimum angle given in micro degrees. The valid range is 0 .. 359,999,999.
- *minRevolution* - Absolute minimum revolution. The valid range is -32768 .. 32767.
- *maxAngle* - Absolute maximum angle given in micro degrees. The valid range is 0 .. 359,999,999.
- *maxRevolution* - Absolute maximum revolution. The valid range is -32768 .. 32767.

Note: The *maxAngle* / *maxRevolution* pair must be greater than the *minAngle* / *minRevolution* pair, otherwise the positioner will not move at all. If both pairs have the same value then the software range limit is disabled.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SAL2,315000000,-1,45000000,0
```

Limits the travel range to +/- 45° around the zero angle.

See also: `GAL`, `SPL`, `GPL`

SCLA - Set Closed Loop Acceleration

Channel Type: Positioner

Description:

This command configures the acceleration control feature of a channel for closed-loop commands such as `MPA`. By default the acceleration control is inactive. If a movement acceleration is configured, all following closed-loop commands will be executed with the new acceleration.

Note that the acceleration control feature requires the speed control feature (see `SCLS`). Enabling acceleration control will implicitly enable the speed control should it be inactive.

Likewise, the acceleration control is required by the Low Vibration mode (see 2.2.2 “Low Vibration Mode”). Disabling the acceleration control will cause the Low Vibration mode to be implicitly disabled as well (see also 2.5.1 “Dependency Chains“).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *acceleration* - Defines the acceleration in µm/s² for linear positioners and in m°/s² for rotary positioners. The valid range is 0..10,000,000. A value of 0 (default) deactivates the acceleration control feature.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
SCLA1,1000
```

Sets the closed-loop acceleration of channel 1 to 1,000 µm/s².

See also: `GCLA`

SCLF - Set Closed Loop max Frequency

Channel Type: Positioner

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.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *frequency* - Defines the maximum frequency in Hz. The valid range is 50 .. 18,500.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SCLF1,3000
```

Sets the closed-loop frequency of channel 1 to 3,000 Hz.

SCLS - Set Closed Loop move Speed

Channel Type: Positioner

Description:

This command configures the speed control feature of a channel for closed-loop commands such as `MPA`. By default the speed control is inactive. In this state the behavior of closed-loop commands is influenced by the maximum driving frequency (see `SCLF`). If a movement speed is configured, all following closed-loop commands will be executed with the new speed. Note that the channel will not drive the positioner with frequencies above the maximum allowed frequency. If the maximum frequency is set too low for a certain movement speed, then the movement speed might not be reached or held. In this case increase the maximum frequency.

Be aware that different positioners reach different speeds. If a positioner is not able to move as fast as the configured move speed, then the driver will cap at the maximum driving frequency.

Note that the sensor of a positioner should be calibrated for proper operation of the speed control (see `CS`).

Note that the speed control feature is required by the acceleration control feature (see `SCLA`). Disabling the speed control will cause the acceleration control to be implicitly disabled as well.

See also section 2.5.1 “Dependency Chains“.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *speed* - Movement speed given in nm/s for linear positioners and in µ°/s for rotary positioners. The valid range is 0 .. 100,000,000. A value of 0 (default) deactivates the speed control feature.

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
SCLS0,1000000
```

Configures channel 0 to execute closed-loop commands with a movement speed of 1mm/s.

See also: `GCLS`

SCP - Set Channel Property

Channel Type: Positioner

Description:

Sets a configuration value of a channel. This is a universal command to configure various channel properties. The property which is to be set is selected via the key parameter. This 32-bit parameter codes a combination of values and has the following structure:

31 24	23 16	15 8	7 0
component	sub component	property	

See the appendix 4.4 “Channel Properties“ for a list of component selectors and properties. The sub component selector is usually an index, but there can also be special sub component selectors. Note that not all properties are valid for all components. Please refer to section 2.2 “Channel Properties” for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *key* - Selects the property of which the value should be set.
- *value* - Defines the value that the selected property should have.

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SCP1,67108869,0
```

Resets the event counter 0 of channel 1 to zero.

See also: `GCP`

SEET - Set End Effector Type

Channel Type: End effector

Description:

Each end effector channel must be configured with the type of end effector that is connected to it before it can be used. This command configures the type along with its parameters depending on the type. There are three types of end effectors:

- Gripper (type 1)
- Force sensor (type 2)
- Gripper with integrated force sensor (type 3)

The parameters that the end effectors must be configured with are taken from the data sheets that come along with the end effectors.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *type* – Specifies the type of the end effector. See table below.
- *param1* - The meaning and valid range of this parameter depends on the type given. See table below.
- *param2* - The meaning and valid range of this parameter depends on the type given. See table below.

Type	param1		param2	
	Meaning	Valid Range	Meaning	Valid Range
Gripper (1)	maximum voltage [1/100 V]	100 – 22,500 (1 - 225V)	none	0
Force Sensor (2)	sensor gain [1/10 µN/V]	1 – 10,000 (0.1 – 1000.0 µN/V)	none	0
Force Gripper (3)	sensor gain [1/10 µN/V]	1 – 10,000 (0.1 – 1,000.0 µN/V)	maximum voltage [1/100 V]	100 – 22,500 (1 - 225V)

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SEET3,1,10000,0          // a gripper with 100V maximum voltage
SEET3,2,3000,0           // a force sensor with 300µN/V sensor gain
```

See also: `GEET`

SESM - Set Emergency Stop Mode

Please note that this command is deprecated. Set the according channel property with `SCP` instead.

Description:

The MCS may be equipped with a hardware TTL input signal that is used as an emergency stop. A negative pulse on this line (*ES line*) causes all channels of the system to stop immediately. The `SESM` command may be used to control the behavior of the system in case of such an emergency.

There are three modes available:

- 0 (Normal): This is the default mode. In this mode a falling edge on the ES line will issue a single stop to all channels. Note that the ES line will be also triggered internally when the RS232 cable is unplugged. After such an event the system continues to behave normally.
- 1 (Restricted): In this mode a falling edge on the ES line will issue a stop to all channels and make the system enter a locked state. In this state you may communicate with the system normally, but all movement commands (open-loop, closed-loop, `FRM`, `CS`...) will return an error (code 146). The locked state may be reset by any call of the `SESM` command, thereby unlocking the movement again.
- 2 (Disabled): In this mode falling edges on the ES line are simply ignored.

Note that this command is global to all channels of a system.

Parameters:

- *mode* - Selects the mode. Must be either 0, 1 or 2.

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SESM1
```

Configures the system for the restricted emergency stop mode.

See also: `GESM`

SP - Set Position

Channel Type: Positioner

Description:

For positioners that have a sensor installed, this command may be used to define the current position resp. angle of the positioner to have a specific value. This command replaces `SZP`.

If the positioner “knows” its physical position (via `FRM`) when sending this command, the position will be saved to non-volatile memory. On future power-ups it will recall its physical position automatically after a `FRM` command. See section 2.3.3 “Defining Positions” for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *position* - Defines the value that the current position of the positioner should have. In case of a rotary positioner the range of this parameter is limited to 0 .. 359,999,999. Note that the revolution implicitly will always be set to 0.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SP0,1000000
```

Defines the current position of positioner 0 to be 1mm (in case of a linear positioner).

SPL - Set Position Limit

Channel Type: Positioner

Description:

For positioners with integrated sensors this command may be used to limit the travel range of a linear positioner by software. (For rotary positioners see `SAL`.) By default there is no limit set. If defined the positioner will not move beyond the limit. This affects open-loop as well as closed-loop movements.
Note that the limit may only be set if the physical position is known at the time of the call (see `FRM`, `GPPK`).

See section 2.3.4 “Software Range Limit“ for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *minPosition* - Absolute minimum position given in nanometers.
- *maxPosition* - Absolute maximum position given in nanometers.

Note: maxPosition must be greater than minPosition, otherwise the positioner will not move at all. If both parameters have the same value then the software range limit is disabled.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SPL2,-1000000,1000000
```

Limits the travel range of the positioner on channel 2 to +/- 1mm around the zero position.

See also: `GPL`

SRC - Set Report on Complete

Channel Type: Positioner, End effector

Description:

This command tells a channel whether or not to report the completion of the last movement command (see section 3.3 “Movement Control Commands”). If set to true, the channel will send a “completed” answer string when it has completed the movement. The default behavior is no reporting.

Note that this command is only available in asynchronous communication mode (see `SCM` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *report* - Must be either 0 (no report) or 1 (report).

Answer String:

none

Example:

```
SRC2,1
```

Instructs channel 2 of the system to report completed movement commands.

SRT - Set Report on Triggered

Channel Type: Positioner

Description:

This command tells a channel whether or not to report when a movement command from the command queue has been triggered (see section 2.4.5 “Command Queues“ for more information). If set to true, the channel will send a notification (“Tn”, with n being the channel index) when the next command in the queue has been triggered. The default behavior is no reporting.

Note that this command is only available in asynchronous communication mode (see `SCM` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *report* - Must be either 0 (no report) or 1 (report).

Answer String:

none

Example:

```
SRT0,1
```

Instructs channel 0 of the system to report when a command was triggered.

SSC - Set Scale

Channel Type: Positioner

Description:

This command may be used to the shift and invert the measuring scale of a positioner. Please see section 2.3.3 “Defining Positions” for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *scale* – Sets the desired scale shift relative to the physical scale of the positioner. The value is given in nano meters for linear positioners and in micro degrees for rotary positioners.
- *inverted* – Sets the scale inversion. Must be either 0 (disabled) or 1 (enabled).

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Caution: Please note that only the logical scale of the positioner will be inverted when the `inverted` value has changed. Parameters like the *SafeDirection* will not be altered. Thus the positioner will move in the opposite direction when e.g. calling `FRM` with the same parameters prior to the inversion change.

Example:

```
SSC0,1000000,0
```

Sets the scale shift of channel 0 to +1mm relative to the physical scale.

See also: `GSC`

SSD - Set Safe Direction

Channel Type: Positioner

Description:

Some sensor types are not equipped with a physical reference mark. For these positioners a mechanical end stop is used as a reference point when sending the `FRM` command. Which end stop is used is configured by the safe direction. This should be the direction in which the positioner may safely move without endangering the physical setup of your manipulator system.

Since the end stop must be calibrated before it can be properly used as a reference point, the safe direction setting also affects the behavior of the `CS` command. Positioners that are referenced via an end stop also move to the configured end stop as part of the calibration routine. Please be aware though that the calibration routine must not start near a mechanical end stop. Otherwise the calibration might fail and cause unexpected behavior in closed-loop mode.

The safe direction setting is a global parameter for a channel and affects the `CS` command as well as the `FRM` command. Please note that latter command will ignore its direction parameter for positioners that are referenced via an end stop and will implicitly use the safe direction parameter instead.

This command has no effect on channels that have a sensor type configured that is referenced via a reference mark. See appendix 4.3 “Sensor Types“ for a list of sensor types and their reference marks.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *direction* – Sets the safe direction. Must be either 0 (forward) or 1 (backward).

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SSD0,1
```

Configures backwards as the safe direction of channel 0.

See also: `GSD`

SSE - Set Sensor Enabled

Description:

This command may be used to activate or deactivate the sensors that are attached to the positioners of a system. The command is system global and affects all positioner channels of a system equally. It effectively turns the power supply of the sensors on or off. Please refer to section 2.3.2 “Sensor Modes” for more information on the sensor modes. End effector channels are not affected by this command.

If this command is issued, all positioner channels of the system are implicitly stopped.

This setting is stored to non-volatile memory immediately and need not be configured on every power-up.

Parameters:

- *enabled* - Selects the mode. Must be either 0 (disabled), 1 (enabled) or 2 (powersave).

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
SSE0
```

Disables the sensors of all positioner channels of the systems.

See also: `GSE`

SST - Set Sensor Type

Channel Type: Positioner

Description:

When using positioners with integrated sensors, this command may be used to tell a channel what type of positioner is connected. The type affects position calculation and commands that may be issued for a channel (see `GP`, `MPA`, `MPR`, `GA`, `MAA` and `MAR`). For example, a channel that is configured as rotary will not accept a `GP` command.

Please refer to appendix 4.3 “Sensor Types“ for a list of available sensor types.

Note that each channel stores this setting to non-volatile memory. Consequently, there is no need to issue this command on every initialization. If the sensor type of a channel is changed, you must call `CS` to ensure proper operation of the sensor.

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

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *typeCode* - Specifies the type of the sensor. Must be a type code listed in appendix 4.3 “Sensor Types”.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SST0,2
```

Configures the sensor type of channel 0 as rotary nanosensor (SR3610s).

See also: `GST`

SSWS - Set Step While Scan

Channel Type: Positioner

Description:

This command is of interest in conjunction with closed-loop commands (e.g. `MPA`, see there) and sets a flag that affects the behavior of a positioner. If the positioner is instructed to hold the target position after reaching it, the scanning mode will primarily be used to hold the position. In this mode it may become necessary to do further steps to hold the position if the deflection of the piezo reaches a boundary. However, if this is not desired, this command may be used to forbid the execution of steps even if this means that the position can not be held.

By default, steps are allowed.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *step* - Must be either 0 (forbid steps) or 1 (allow steps).

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
SSWS0,0
```

Forbids channel 0 of the system to execute step movements while holding a position after a closed-loop command.

SZF - Set Zero Force

Channel Type: End effector

Description:

End effectors that have a force sensor do not measure absolute force, but rather a change of force. For proper force measurement this command may be used to set the measured force to zero and should be called when the force sensor is mechanically unstressed.
Setting the zero force takes about one second. During this time the end effector will report a status of 6. Note that in the asynchronous mode the channel will report completion of the command if configured so with the `SRC` command (see there).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SZF0
```

Defines the current force of end effector 0 as its zero force.

SZP - Set Zero Position

Channel Type: Positioner

Description:

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

Please note that this command is deprecated. Call `SP` instead.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
SZP0
```

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

See also: `SP`

3.3 Movement Control Commands

ATC – Append Triggered Command

Channel Type: Positioner

Description:

This command is used in combination with movement commands to fill the command queue with commands. Queued movement commands are not executed right away, but rather triggered by a configurable event source. Please refer to section 2.4.5 “Command Queues” for more information.

After calling `ATC`, the next movement command will be put into the command queue for later execution.

The *triggerSource* parameter must be given in form of a selector value which is a 32-bit code that refers to an event source and has the following structure:

31 24	23 16	15 8	7 0
unused		component	index

Please refer to the appendix 4.4.1 “Trigger Source Codes“ for a list of trigger source codes.

Note that this command is only available in asynchronous communication mode (see `SCM` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *triggerSource* - Defines the event source that should be used to trigger the next command. This parameter is coded in form of a selector value that refers to an event source.

Answer String:

none

Example:

```
ATC1,1792
```

Instructs channel 1 to queue the next command issued into the command queue. The command will be triggered by software (see `TC`) with the trigger index 0.

See also the example in section 2.4.5 “Command Queues“.

CS - Calibrate Sensor

Channel Type: Positioner

Description:

This command may be used to increase the accuracy of the position calculation. It is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see `SSE` command). If this is not the case the channel will return an error.

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 nonvolatile memory. If the mechanical setup is unchanged, it is not necessary to send this command on each initialization, but newly connected positioners have to be calibrated in order to ensure proper operation.

During the calibration the positioner will perform a movement of up to several mm. You must ensure, that the command is not executed while the positioner is near a mechanical end stop. Otherwise the calibration might fail and lead to unexpected behavior when executing closed-loop commands. As a safety precaution, also make sure that the positioner has enough freedom to move without damaging other equipment.

The calibration takes a few seconds to complete. During this time the positioner will report a status code of 6 (see `GS` command and refer to appendix 4.1 “Channel Status Codes” for a list of status codes).

Positioners that are referenced via a mechanical end stop (see 4.3 “Sensor Types“) are moved to the end stop as part of the calibration routine. Which end stop is used for referencing is configured with the `SSD` command. Note that when changing the safe direction the end stop must be calibrated again for proper operation.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
CS1
```

Calibrates the sensor of the positioner connected to channel 1.

CTCQ – Clear Triggered Command Queue

Channel Type: Positioner

Description:

This command is used in conjunction with command queuing (see section 2.4.5 “Command Queues”). If there are movement commands in the command queue then it is not possible to issue (non-queued) movement commands until the queue is empty again (all commands in the queue must have been triggered).

This command may be used to cancel all commands that are in the command queue. The commands will not be executed, but simply removed from the queue. After this the queue size will be zero.

Note that this command is only available in asynchronous communication mode (see `SCM` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

none

Example:

```
CTCQ0
```

Cancel all commands in the command queue.

FRM - Find Reference Mark

Channel Type: Positioner

Description:

For positioners that are equipped with sensor feedback, this command may be used to move the positioner to a known physical position of the positioner. Some sensor types are equipped with a physical reference mark (which is typically located near the middle of the travel range), others are referenced via a mechanical end stop (see appendix 4.3 “Sensor Types“). For latter types you must configure the safe direction with `SSD` and issue `CS` before the positioner can be properly referenced. The safe direction is then used instead of the direction parameter described below.

If the auto zero flag is set, the current position resp. angle is set to zero after the reference position has been reached. Otherwise the position is set according to the information stored in non-volatile memory of the last `SP` command. See section 2.3.3 “Defining Positions“ for more information.

As a safety precaution, make sure that the positioner has enough freedom to move without damaging other equipment.

The positioner may be instructed to hold the position of the reference mark after it has been reached. This behavior is similar to that of the other closed-loop commands, e.g. `MPA`. See there for more information.

While executing the command the positioner will have a movement status of 7. While holding the position the positioner will have a movement status of 3 (see `GS`).

If this command was successful, then the physical position of the positioner becomes known. See `GPPK`.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *direction* - Must be either 0 (forward) or 1 (backward).
- *holdTime* - Specifies how long (in milliseconds) the position is actively held after reaching the reference mark. The valid range is 0..60,000. A 0 deactivates this feature, a value of 60,000 is infinite (until manually stopped, see `S` command).
- *autoZero* - Must be either 0 (no auto zero) or 1 (auto zero).

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
FRM2,0,2000,1
```

Searches for the reference mark of the positioner on channel 2 in forward direction. When reached the position will be set to zero and held for 2 seconds.

MAA - Move to Angle Absolute

Channel Type: Positioner

Description:

Instructs a positioner to turn to a specific angle. This command is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see `SSE` command). If this is not the case an error will be returned. 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 (use `MPA` instead).

The positioner may be instructed to hold the target angle after it has been reached. This may be useful to compensate for drift effects and the like. Note that the positioner will use the scan mode to hold the angle if needed. When the piezo element of the positioner reaches a scanning boundary a single step is performed. However, if this behavior is not desired the correction steps can be disabled with the `SSWS` command (see there). Note though that disabling the steps might mean that the position cannot be held.

While executing the command the positioner will have a movement status code of 4. While holding the target angle the positioner will have a movement status code of 3 (see `GS` command).

If a mechanical end stop is detected while the command is in execution, the movement will be aborted. In asynchronous communication mode an error will be reported in this case.

Please refer to section 2.3.1 “Rotary Sensors” for more information on the angle and revolution parameters.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *angle* - Absolute angle to move to in micro degrees. The valid range is 0..359,999,999.
- *revolution* - Absolute revolution to move to. The valid range is -32,768..32,767.
- *holdTime* - Specifies how long (in milliseconds) the angle is actively held after reaching the target. The valid range is 0..60,000. A 0 deactivates this feature, a value of 60,000 is infinite (until manually stopped, see `S` command).

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
MAA0,90000000,0,1000
```

Moves the positioner on channel 0 to 90° angle (revolution 0) and holds it for one second.

See also: `MAR`, `MPA`, `MPR`

MAR - Move to Angle Relative

Channel Type: Positioner

Description:

Instructs a positioner to move to an angle relative to its current angle. This command is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see `SSE` command). If this is not the case the channel will return an error. 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 (use `MPR` instead).

If a relative positioning command is issued before a previous one has finished, normally the relative targets are accumulated. If this is not desired it can be disabled with the `SARP` command (see there for more information).

The positioner may be instructed to hold the target angle after it has been reached. See `MAA` command for more information.

While executing the command the positioner will have a movement status code of 4. While holding the target angle the positioner will have a movement status code of 3 (see `GS` command).

If a mechanical end stop is detected while the command is in execution, the movement will be aborted. Note that in asynchronous communication mode an error will be reported.

Please refer to section 2.3.1 “Rotary Sensors” for more information on the angle and revolution parameters.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *angleDiff* - Relative angle to move to in micro degrees. The valid range is -359,999,999..359,999,999.
- *revolutionDiff* - Relative revolution to move to. The valid range is -32,768..32,767.
- *holdTime* - Specifies how long (in milliseconds) the angle is actively held after reaching the target. The valid range is 0..60,000. A 0 deactivates this feature, a value of 60,000 is infinite (until manually stopped, see `S` command).

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
MAR0,-45000000,-1,60000
```

Performs one full turn plus another 45° in negative direction on channel 0. The target angle is held until given another movement command or explicitly stopped.

See also: `MAA`, `MPA`, `MPR`

MGFA - Move to Gripper Force Absolute

Channel Type: End effector

Description:

This closed-loop command is only executable if the end effector that is connected to the channel is a gripper with an integrated force sensor (see `SEET` command). The command may be used to grab an object with a defined and constant force. The channel will adjust the output voltage of the gripper so that the force sensor measures the given force.

Note that the force sensor must be calibrated in order for this command to function properly. See the `SZF` command.

While executing the command the end effector will have a status of 4 (see `GS` command). Once the target force is reached it will try to hold the given force and have a status of 3 until the channel is explicitly stopped (see `S` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *force* - Specifies the force that is to be applied. It is given in tenths of µN, e.g. a value of 100 would be 10µN. The valid range for this parameter is -100,000 ..100,000.
- *speed* - This parameter may be used to limit the speed with which the gripper is opened or closed. It is given in Volts per second. The valid range for this parameter is 1 .. 225,000.
- *holdTime* – This parameter has not been implemented yet. Set to zero.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
MGFA0,1000,10,0
```

Grabs an object with a force of 100µN carefully with 10 V/s.

MGOA - Move to Gripper Opening Absolute

Channel Type: End effector

Description:

For end effectors that have a gripper this command may be used to open or close the gripper. For this a voltage is applied to it. Applying 0V will open the gripper all the way. The higher the applied voltage the further the gripper will close. The maximum allowed voltage depends on the gripper type (see `SEET`). If the given voltage is higher than the allowed voltage for the currently configured end effector the channel will stop at the maximum voltage.

While executing the command the end effector will have a status of 8 (see `GS`).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *opening* - Specifies the target voltage which is given in 1/100 Volts, e.g. a value of 10,000 would be 100V. The valid range for this parameter is 0 .. 22,500.
- *speed* - Specifies the speed of the voltage adjustment. It is given in Volts per second. The valid range for this parameter is 1.. 225,000.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
MGOA0,5000,50
```

Opens the gripper to 50V with 50V/s.

See also: `MGOR`

MGOR - Move to Gripper Opening Relative

Channel Type: End effector

Description:

This command is similar to `MGOA` (see there) with the difference that a relative movement is performed. If the resulting target voltage exceeds the allowed range (below zero or above the maximum voltage for the currently configured gripper type) the channel will stop at the boundary.

While executing the command the end effector will have a status of 8 (see `GS`).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *diff* - Specifies the relative target voltage which is given in 1/100 Volts, e.g. a value of -1,000 would "open the gripper by 10V". The valid range for this parameter is -22,500 .. 22,500.
- *speed* - Specifies the speed of the voltage adjustment. It is given in Volts per second. The valid range for this parameter is 1.. 225,000.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
MGOR0,2000,40
```

Closes the gripper by 20V in half a second.

See also: `MGOA`

MPA - Move to Position Absolute

Channel Type: Positioner

Description:

Instructs a positioner to move to a specific position. This command is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see SSE command). If this is not the case an error will be returned. Additionally, the command is only executable if the addressed channel is configured to be of type linear (see SST command). A rotary channel will return an error (use MAA instead).

The positioner may be instructed to hold the target position after it has been reached. This may be useful to compensate for drift effects and the like. Note that the positioner will use the scan mode to hold the position if needed. When the piezo element of the positioner reaches a scanning boundary a single step is performed. However, if this behavior is not desired the correction steps can be disabled with the SSWS command (see there). Note though that disabling the steps might mean that the position cannot be held.

While executing the command the positioner will have a movement status code of 4. While holding the target position the positioner will have a movement status code of 3 (see GS command).

If a mechanical end stop is detected while the command is in execution, the movement will be aborted. In asynchronous communication mode an error will be reported in this case.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *position* - Absolute position to move to in nano meters.
- *holdTime* - Specifies how long (in milliseconds) the position is actively held after reaching the target. The valid range is 0..60,000. A 0 deactivates this feature, a value of 60,000 is infinite (until manually stopped, see S command).

Answer String:

- Sync mode: E<channelIndex>,0
- Async mode: none

Example:

```
MPA0,-1000000,0
```

Moves to the absolute position of minus one millimeter on channel 0. The position is not held after the movement has completed.

See also: MPR, MAA, MAR

MPR - Move to Position Relative

Channel Type: Positioner

Description:

Instructs a positioner to move to a position relative to its current position. This command is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see SSE command). If this is not the case the channel will return an error. Additionally, the command is only executable if the addressed channel is configured to be of type linear (see SST command). A rotary channel will return an error (use MAR instead).

If a relative positioning command is issued before a previous one has finished, normally the relative targets are accumulated. If this is not desired it can be disabled with the SARP command (see there for more information).

The positioner may be instructed to hold the target position after it has been reached. See MPA command for more information.

While executing the command the positioner will have a movement status code of 4. While holding the target position the positioner will have a movement status code of 3 (see GS command).

If a mechanical end stop is detected while the command is in execution, the movement will be aborted. In asynchronous communication mode an error will be reported in this case.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *position* - Relative position to move to in nano meters.
- *holdTime* - Specifies how long (in milliseconds) the position is actively held after reaching the target. The valid range is 0..60,000. A 0 deactivates this feature, a value of 60,000 is infinite (until manually stopped, see S command).

Answer String:

- Sync mode: E<channelIndex>,0
- Async mode: none

Example:

```
MPR0,500000,2000
```

Moves the positioner on channel 0 500 microns in positive direction. After this the position is held for two seconds.

See also: MPA, MAA, MAR

MSCA - Move SCan Absolute

Channel Type: Positioner

Description:

Performs a scanning movement of a positioner to a specific target scan position. This command may be used to directly control the deflection of the piezo of the positioner.

While executing the command the positioner will have a movement status code of 2 (see `GS` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *target* - Target scan position to which to scan to. The value is given as a 12bit value (range 0..4,095). 0 corresponds to 0V, 4,095 to 100V.
- *scanSpeed* - The valid range is 1 .. 4,095,000,000 and represents single 12bit increments per second. With a value of 1 a scan over the full range from 0 to 4,095 takes 4,095 seconds while at full speed the scan is performed in one micro second.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
MSCA3,2048,100000
MSCA3,0,1024
```

Scans to 50V on channel 3, then to 0V within two seconds.

See also: `MSCR`

MSCR - Move SCan Relative

Channel Type: Positioner

Description:

Performs a relative scanning movement of a positioner.

While executing the command the positioner will have a movement status code of 2 (see `GS` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *diff* - Relative scan target to which to scan to. The valid range is -4,095 .. 4,095. If the resulting absolute scan target exceeds the valid range of 0..4,095 the scan movement will stop at the boundary.
- *scanSpeed* - The valid range is 1 .. 4,095,000,000 and represents single 12bit increments per second. With a value of 1 a scan over the full range from 0 to 4,095 takes 4,095 seconds while at full speed the scan is performed in one micro second.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Note: This Command is not available on all controllers. Please contact SmarAct for more information.

Example:

```
MSCA3,2048,100000
MSCR3,-1024,1024
```

Scans to 50V on channel 3, then to 25V within one second.

See also: `MSCA`

MST - Move STep

Channel Type: Positioner

Description:

This is an open-loop command. It performs a burst of steps with the given parameters. Note that a single step is atomic. When interrupting a burst with the `S` command the positioner will finish the current step before stopping. This implies that the piezo element of the positioner is always at its resting potential after a step command.

While executing the command the positioner will have a movement status code of 1 (see `GS` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *steps* - Number and direction of steps to perform. The valid range is -30,000..30,000. A value of 0 stops the positioner, but see `S` command. A value of 30,000 or -30,000 performs an unbounded move. This should be used with caution since the positioner will only stop on an `S` command.
- *amplitude* - Amplitude that the steps are performed with. Lower amplitude values result in a smaller step width. The parameter must be given as a 12bit value (range 0..4,095). 0 corresponds to 0V, 4,095 to 100V.
- *frequency* - Frequency in Hz that the steps are performed with. The valid range is 1..18,500.

WARNING: It is strongly discouraged to use unbounded moves, especially at high frequencies! Positioners develop heat when they are driven and may take permanent damage if constantly driven over a long period of time (> 1 minute), especially in environments with weak thermal coupling. Be aware that if driven in the ultra sonic frequency range there is the risk of an unintended unbounded move to go unnoticed!

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
MST0,100,4095,100
```

Performs 100 steps at full amplitude and 100 Hz on channel 0.

S - Stop

Channel Type: Positioner, End effector

Description:

Stops any ongoing movement of a positioner. Note that if a stepping movement is performed (see `MST` command) the current step is completed before the positioner is stopped. This command also stops the hold position feature of closed-loop commands, such as `MPA` or even `FRM`.

A positioner that is stopped will have a movement status code of 0 (see `GS` command).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Note: The channel index may be omitted. In this case all channels of the system are stopped simultaneously.

Answer String:

- Sync mode: `E<channelIndex>,0`
- Async mode: none

Example:

```
S0
```

Stops channel 0.

TC - Trigger Command

Channel Type: Positioner

Description:

This command triggers commands that were loaded into the command queue with `ATC` (see chapter 2.4.5 “Command Queues”) while specifying the software trigger as event source. Note that this command is global to a system and is sent to all channels of a system simultaneously. However, the command will only be triggered if the index that was given while specifying the event source is the same as the *triggerIndex* passed to `TC`. This mechanism enables you to e.g. preload all channels of a system with movement commands, but group different sets of channels that should start their movement at different times.

Note that this command is only available in asynchronous communication mode (see `SCM` command).

Parameters:

- *triggerIndex* - Index of the trigger command. Only commands that were loaded with this index are actually triggered. The valid range is 0 .. 255.

Answer String:

none

Example:

```
TC5
```

Triggers all commands that were queued with software trigger index 5.

3.4 Positioner Feedback Commands

GA - Get Angle

Channel Type: Positioner

Description:

Requests the current angle of a positioner. This command is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see `SSE` command). If this is not the case the channel will return an error. 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 (use `GP` instead).

A rotary position is defined by a combination of an angle and a revolution. One revolution equals a full 360° turn. The angle value returned will always be in the range 0..359,999,999. If the positioner moves over a zero boundary, the angle value will wrap around accordingly and the revolution value will be incremented resp. decremented.

Please refer to section 2.3.1 “Rotary Sensors” for more information on the angle and revolution values.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
A<channelIndex>,<angle>,<revolution>
```

<angle> represents the current angle in micro degrees.

Example:

```
GA2
```

Requests the current angle of channel 2.

See also: `GP`

GB - Get capture Buffer

Channel Type: Positioner

Description:

Retrieves the contents of a capture buffer. Capture buffers capture (groups of) internal values on certain events and may be used to synchronize the reading of values.

See section 2.4.4 “Capture Buffers” for a list of currently defined capture buffers.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *bufferIndex* - Selects the buffer of the selected channel.

Answer String:

CB<channelIndex>,<bufferIndex>,<data1>,...,<datan>

Depending on the capture buffer index there will be a number of data fields returned. Please refer to section 2.4.4 “Capture Buffers” for their meanings.

Example:

```
GB1,0
```

Requests the capture buffer with index 0 from channel 1.

GF - Get Force

Channel Type: End effector

Description:

This command is only executable if the end effector that is connected to the select channel has a force sensor (see `SEET`). The command requests the force that is currently measured by the sensor. Note that the sensor must be calibrated in order to provide proper measurement values (see `SZF`).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
F<channelIndex>,<force>
```

<force> represents the currently measured force in 1/10 µN.

Example:

```
GF1
```

Requests the current force of channel 1.

See also: `MGFA`

GGO - Get Gripper Opening

Channel Type: End effector

Description:

This command is only executable if the end effector that is connected to the selected channel is a gripper (see `SEET`). The command requests the voltage that is currently applied to the gripper.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
GO<channelIndex>,<opening>
```

<opening> represents the current gripper opening in 1/100 Volts.

Example:

```
GGO1
```

Requests the current gripper opening of channel 1.

See also: `MGOA`, `MGOR`

GP - Get Position

Channel Type: Positioner

Description:

Returns the current position of a positioner. This command is only executable by a positioner that has a sensor attached to it. The sensor must also be enabled or in power save mode (see `SSE` command). If this is not the case the channel will return an error. Additionally, the command is only executable if the addressed channel is configured to be of type linear (see `SST` command). A rotary channel will return an error (use `GA` instead).

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
P<channelIndex>,<position>
```

<position> represents the current position in nano meters.

Example:

```
GP1
```

Requests the current position of channel 1.

See also: `GA`

GPPK - Get Physical Position Known

Channel Type: Positioner

Description:

Returns whether the positioner “knows” its physical position. After a power-up the physical position is unknown and the current position is implicitly assumed to be the zero position. After the reference mark has been found by sending a `FRM` command the physical position becomes known.

This command can be useful if the software application restarts and connects to a system that has stayed online. If the physical position is already known, traveling to the reference mark again may be omitted.

See also 2.3.3 “Defining Positions” for more information.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
PPK<channelIndex>,<known>
```

<known> will be either 0 (unknown) or 1 (known).

Example:

```
GPPK0
```

Queries whether positioner 0 knows its physical position or not.

See also: `FRM`

GS - Get Status

Channel Type: Positioner, End effector

Description:

Returns the current movement status code of a positioner or end effector (see appendix 4.1 “Channel Status Codes” for a list of movement status codes). This command can be used to check whether a previously issued movement command has been completed.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
S<channelIndex>,<status>
```

<status> will be one of the status codes listed in appendix 4.1 “Channel Status Codes”.

Example:

```
GS0
```

Requests the current status of channel 0.

GVL - Get Voltage Level

Channel Type: Positioner

Description:

Returns the voltage level that is currently applied to the piezo element of a positioner. This command is mainly of interest in conjunction with `MSCA` and `MSCR`, since these are used to control the voltage level.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
VL<channelIndex>,<level>
```

<level> represents the currently applied voltage level. It ranges from 0..4,095, where 0 corresponds to 0V and 4,095 to 100V.

Example:

```
GVL1
```

Requests the current voltage level of channel 1.

See also: `MSCA`, `MSCR`

3.5 Miscellaneous Commands

CB - Configure Baudrate

Description:

This command sets the baud rate of the RS232 interface to the specified value. The setting is stored to non-volatile memory and loaded on future power ups. Note that the 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 (`R`).

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.

Parameters:

- *baudrate* - The valid range is 9600 .. 115,200.

Answer String:

```
BR<baudrate>
```

Example:

```
CB57600
```

Configures the RS232 baud rate to 57600 baud. In this case the string `BR57142` will be returned (0.8% error).

Note: This command is not available on MCS controllers with network interface.

GSN – Get Serial Number

Description:

This command may be used to retrieve the serial number of a channel. Every channel has an unique serial number which can be helpful when maintaining the system. The return value is hexadecimal.

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
SN<channelIndex>,0x<serialnumber>
```

Example:

```
GSN0
```

Requests the serial number of the first channel.

GFV – Get Firmware Version

Description:

This command may be used to retrieve the current firmware version of a channel. The answer string will hold the device type of the current channel (Slip-Stick, Pico-Slide, etc.) as well as the version of the Signal Controller and the Signal Generator. See following table for the corresponding keys:

	Signal Controller				Signal Generator			
Channel Nr.	Device Type	Version High	Version Update	Version Revision	Device Type	Version High	Version Update	Version Revision

The Device Types are shown in the following table:

Type	Product
1	Interface Module USB
2	Signal Controller (Slip-Stick)
3	Signal Generator (Slip-Stick)
5	Hand Contol Module
6	Signal Controller (Pico-Slide)
7	Signal Generator (Pico-Slide)
8	Interface Module (ASCII)
9	End Effector Module

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.

Answer String:

```
FV<channelIndex>,<deviceType>,<versionHigh>,<versionUpdate>,
  <versionRevision>,<deviceType>,<versionHigh>,<versionUpdate>,
  <versionRevision>
```

Example:

```
GFV0
```

Requests the firmware version information of the first channel.

GFP – Get Feature Permissions

Description:

This command may be used to retrieve the current feature permissions of a channel. Since the feature permissions are bit encoded in the controller the user must first determine how many feature permission bytes are available and then retrieve the information byte by byte.

Consequently this command has two parameters, where the first specifies the targeted channel and the second fetches one feature permission byte. If the second parameter is `255` the answer string will hold the number of valid feature permission bytes.

A list of feature permissions is found in the following table, please note that features can be combined resulting in different answer values (see example below). A zero indicates that a feature is permitted:

Byte	Bit	Feature
0	0	Low Vibration Mode
0	1	Periodic Sensor Error Correction

Parameters:

- *channelIndex* - Selects the channel of the system. The index is zero based.
- *byte index* - Specifies which feature byte is targeted (Index is zero based). Note that the value `255` requests the number of the feature bytes.

Answer String:

```
FP<channelIndex>,<size>|<value>
```

Example:

```
GFP0,255          // Requests the feature permission size of channel one.
FP0,1             // Answer: The size of the feature permissions is one byte.
GFP0,0            // Request the enabled features in byte one.
FP0,254           // Answer: 0b11111110 Low Vibration Mode Available.
```

K - Keep alive

Description:

The MCS offers a timeout mechanism which can be used to stop all positioners immediately if the system does not receive a command within a certain interval.

If one parameter is specified with this command, then the timeout is set to this value given in milliseconds. A value of 0 (default) disables the timeout feature.

If no parameter is given, it causes the internal timeout counter to be reset. If a timeout occurs, all positioners are immediately stopped.

If the timeout is enabled any command string that could be parsed successfully implicitly resets the timeout counter. If no command needs to be sent, use the `K` command without a parameter to force the reset of the timeout counter.

Parameters:

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

Answer String:

- Sync mode: `E-1,0`
- Async mode: none

Example:

```
K3000
```

Sets the timeout delay to 3 seconds. If no `K` command or any other valid command is received for 3 seconds all positioners will be stopped.

4 Appendix

4.1 Channel Status Codes

The table below lists the status codes of positioners or end effectors that are returned by the GS command.

Code	Description
0	Stopped - The positioner is currently not performing active movement (see S command).
1	Stepping - The positioner is performing an open-loop movement (see MST command).
2	Scanning - The positioner is performing a scanning movement (see commands MSCA and MSCR).
3	Holding - The positioner is holding its current target position resp. angle (see closed-loop commands, e.g. MPA or MAA) or is holding the reference mark (see FRM command).
4	Targeting - The positioner is performing a closed-loop movement (see closed-loop commands, e.g. MPA or MAA).
5	Move Delay - The positioner is currently waiting for the sensors to power up before executing the movement command. This status may be returned if the the sensors are operated in power save mode. See section 2.3.2 “Sensor Modes” for more information.
6	Calibrating - The positioner is busy calibrating its sensor (see commands CS and SZF).
7	Finding Reference Mark - The positioner is moving to find the reference mark (see FRM command).
9	Locked - An emergency stop has occurred and further movements are not allowed (see SESM command).

4.2 Error Codes

An error answer string has the general format `E<sourceChannel>,<errorCode>`. If the value of <sourceChannel> is -1, this indicates that the error does not originate from a specific channel, but rather from the overall system. The table below lists the error codes and their meanings.

Code	Meaning Description
0	No Error This indicates that no error occurred and therefore corresponds to an acknowledge.
1	Syntax Error The command could not be processed due to a syntactical error.
2	Invalid Command Error The command given is not known to the system.
3	Overflow Error This error occurs if a parameter given is too large and therefore cannot be processed.
4	Parse Error The command could not be processed due to a parse error.
5	Too Few Parameters Error The specified command requires more parameters in order to be executed.
6	Too Many Parameters Error There were too many parameters given for the specified command.
7	Invalid Parameter Error A parameter given exceeds the valid range. Please see the command description for valid ranges of the parameters.
8	Wrong Mode Error This error is generated if the specified command is not available in the current communication mode. For example, the `SRC` command is not executable in synchronous mode.
129	No Sensor Present Error This error occurs if a command was given that requires sensor feedback, but the addressed positioner has none attached.
140	Sensor Disabled Error This error occurs if a command was given that requires sensor feedback, but the sensor of the addressed positioner is disabled (see `SSE` command).
141	Command Overridden Error This error is only generated in the asynchronous communication mode. When the software commands a movement which is then interrupted by the Hand Control Module, an error of this type is generated.
142	End Stop Reached Error This error is generated in asynchronous mode if the target position of a closed-loop command could not be reached, because a mechanical end stop was detected. After this error the positioner will have a movement status code of 0 (stopped).
143	Wrong Sensor Type Error This error occurs if a closed-loop command does not match the sensor type that is currently configured for the addressed channel. For example, issuing a `GP` command while the targeted channel is configured as rotary will lead to this error.
144	Could Not Find Reference Mark Error This error is generated in asynchronous mode (see `SCM`) if the search for a reference mark was aborted. See section ?? “Reference Marks” for more information.

145 Wrong End Effector Type Error
This error occurs if a command does not match the end effector type that is currently configured for the addressed channel. For example, sending GF while the targeted channel is configured for a gripper will lead to this error.

146 Movement Locked Error
This error occurs if a movement command is issued while the system is in the locked state. See section 2.2.1 “Emergency Stop“ for more information.

147 Range Limit Reached Error
If a range limit is defined (SPL or SAL) and the positioner is about to move beyond this limit, then the positioner will stop and report this error (only in asynchronous mode, see SCM). After this error the positioner will have status code of 0 (stopped).

148 Physical Position Unknown Error
A range limit is only allowed to be defined if the positioner “knows” its physical position. If this is not the case, the commands SPL and SAL will return this error code.

150 Command Not Processable Error
This error is generated if a command is sent to a channel when it is in a state where the command cannot be processed. For example, to change the sensor type of a channel the addressed channel must be completely stopped. In this case send a stop command before changing the type.

151 Waiting For Trigger Error
If there is at least one command queued in the command queue then you may only append more commands (if the queue is not full), but you may not issue movement commands for immediate execution. Doing so will generate this error. See section 2.4.5 “Command Queues“.

152 Command Not Triggerable Error
After sending a ATC command you are required to issue a movement command that is to be triggered by the given event source. Commands that cannot be triggered will generate this error.

153 Command Queue Full Error
This error is generated if you attempt to append more commands to the command queue, but the queue cannot hold anymore commands. The queue capacity may be read out with a get channel property command (GCP on p.28).

154 Invalid Component Error
Indicates that a component (e.g. SCP) was selected that does not exist.

155 Invalid Sub Component Error
Indicates that a sub component (e.g. SCP) was selected that does not exist.

156 Invalid Property Error
Indicates that a property (e.g. SCP) was selected that does not exist.

157 Permission Denied Error
This error is generated when you call a functionality which is not unlocked for the system (e.g. Low Vibration Mode).

4.3 Sensor Types

The following table lists the currently available sensor types that may be configured with the `SST` command.

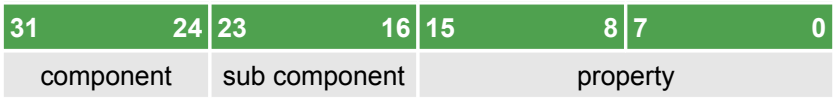
The reference type indicates the way the positioner is referenced when sending the `FRM` command. Positioners with 'mark' are referenced via a physical reference mark that is typically located near the middle of the complete travel range. Positioners with 'end stop' are referenced via a mechanical end stop (see `SSD` command for more information). Positioners with 'none' cannot be referenced.

Symbol	*Type Code*	*Positioner Series*	*Comment*	*Reference Type*
S	1	SLCxxxxs	linear positioner with nano sensor	mark
SR	2	SR36xxs, SR3511s, SR5714s, SR7021s, SR2812s	rotary positioner with nano sensor	mark
SP	5	SLCxxxxrs	linear positioner with nano sensor, large actuator	mark
SC	6	SLCxxxxsc	linear positioner with nano sensor, distance coded reference marks	mark*
SR20	8	SR2013s, SR1612s	rotary positioner with nano sensor	mark
M	9	SLCxxxxm	linear positioner with micro sensor	end stop
GD	11	SGO60.5m	goniometer with micro sensor (60.5mm radius)	end stop
GE	12	SGO77.5m	goniometer with micro sensor (77.5mm radius)	end stop
GF	14	SR1209m	rotary positioner with micro sensor	end stop
G605S	16	SGO60.5s	goniometer with nano sensor (60.5mm radius)	mark
G775S	17	SGO77.5s	goniometer with nano sensor (77.5mm radius)	mark
SC500	18	SLLxxsc	linear positioner with nano sensor, distance coded reference marks	mark*
G955S	19	SGO95.5s	goniometer with nano sensor (95.5mm radius)	mark
SR77	20	SR77xxs	rotary positioner with nano sensor	mark
SD	21	SLCxxxxds, SLLxxs	like S, but with extended scanning Range	mark
R20ME	22	SR2013sx, SR1410sx	rotary positioner with MicroE sensor	mark
SR2	23	SR36xxs, SR3511s, SR5714s, SR7021s, SR2812s	like SR, for high applied masses	mark
SCD	24	SLCxxxxdsc	like SP, but with distance coded reference marks	mark*
SRC	25	SR7021sc	like SR, but with distance coded reference marks	mark*
SR36M	26	SR3610m	rotary positioner, no end stops	none
SR36ME	27	SR3610m	rotary positioner with end stops	end stop
SR50M	28	SR5018m	rotary positioner, no end stops	none
SR50ME	29	SR5018m	rotary positioner with end stops	end stop
G1045S	30	SGO104.5s	goniometer with nano sensor (104.5mm radius)	mark
G1395S	31	SGO139.5s	goniometer with nano sensor (139.5mm radius)	mark
MD	32	SLCxxxxdme	like M, but with large actuator	end stop
G935M	33	SGO93.5me	goniometer with micro sensor (93.5mm radius)	end stop
SHL20	34	SHL-20	high load vertical positioner	mark
SCT	35	SLCxxxxscu	like SCD, but with even larger actuator	mark*

* These positioners are equipped with multiple reference marks. The positioner will only have to move a few milli meters to know its physical position.

4.4 Channel Properties

When sending `GCP` or `SCP` commands you must supply a property key to indicate which property you wish to read or write. A property key is a 32-bit code that refers to a property and has the following structure:



The table below lists the valid component, sub component and property combinations that may be manipulated with the `GCP` and `SCP` commands. If a component has an index as sub component then the column “Sub Component” lists the currently available index values. Some component properties are read-only which is indicated in the “Access” column. These keys may only be passed to `GCP`, but not to `SCP`.

Component	*Sub Component*	*Property*	*Key Code*	*Access*	*Valid Value Range*	*Page*
1 (General)	1 (Emergency Stop)	1 (Operation Mode)	16842753	R / W	0* (Normal), 1 (Restricted), 2 (Disabled), 3 (Auto Release)	5
		45 (Default Operation Mode)	16842797	R / W	0* (Normal), 1 (Restricted), 2 (Disabled), 3 (Auto Release)	5
	2 (Low Vibration) ([1])	1 (Operation Mode)	16908289	R / W	0* (Disabled), 1 (Enabled)	6
	4 (Broadcast Stop)	1 (Operation Mode)	17039361	R / W	0* (Disabled), 1 (Enabled)	6
	5 (Position Control)	17 (Forced Slip)	17104913	R / W	0* (Disabled), 1 (Enabled)	5
8 (Sensor)	11 (Power Supply)	1 (Operation Mode)	134938625	R / W	0 (Disabled), 1 (Enabled), 2* (Powersave)	6
	22 (Scale)	47 (Offset)	135659567	R / W	±2,000,000,000	9
		19 (Inverted)	135659539	R / W	0* (Normal), 1 (Inverted)	9

Controller Event System related Channel Properties

Component	*Sub Component*	*Property*	*Key Code*	*Access*	*Valid Value Range*	*Page*
2 (Digital In)	0	1 (Operation Mode)	33554433	R / W	0* (Disabled), 1 (Enabled)	13
		2 (Active Edge)	33554434	R / W	0* (Falling edge), 1 (Rising edge)	
4 (Counter)	0	3 (Trigger Source)	67108867	R / W	0* (Disabled), <Selector Value>	14
		5 (Value)	67108869	R / W	0* .. 2,147,486,647	
5 (Capture Buffer)	0	3 (Trigger Source)	83886083	R / W	0* (Disabled), <Selector Value>	14
6 (Command Queue)	0	4 (Size)	100663300	R	0* .. n	15
		6 (Capacity)	100663302	R	n	
7 (Software Trigger) ([2])	0 .. 255	none	117440512, 117506048, …, 134152192	N/A	N/A	14

* indicates default values of the properties

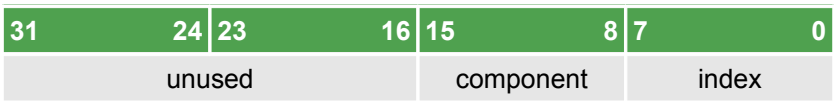
([1]) This feature is not available on all controllers. Please contact SmarAct for more information.

([2]) This component has no properties and may only be used as a trigger source.

For example, to set the value of the counter with index 0 (for channel 2) to 1234, send `SCP2,67108869,1234`.

4.4.1 Trigger Source Codes

The value of a trigger source property is a selector value which is (similar to property keys) a 32-bit code that refers to an event source and has the following structure:



The following table summarizes the codes for the trigger sources when setting trigger source properties (with `SCP`) or appending queued commands (with `ATC`).

Trigger Source	Code
Digital In 0	512
Software Trigger (index 0)	1792
Software Trigger (index 1)	1793
...	...
Software Trigger (index 255)	2047