USB interface

The SMD3 appears as a virtual COM port when plugged into a PC. Configuration and control may be performed via the provided SMD3 software, your own application or any terminal program.

This section is only relevant if you are using terminal software or writing your own application. The provided SMD3 software interacts with the SMD3 in the same way and requires no specialist knowledge to use. A C# API is also provided, allowing you to easily integrate communication with the SMD3 into your own C# .NET application. This is available from our website, at https://arunmicro.com/products/smd3-stepper-motor-drive/.

Configuration

The SMD3 uses a USB Type-C connector, which is both robust and reversible (it may be inserted either way up). The device appears as a virtual COM port on your PC. Configure the serial port as follows:

- 115200 baud
- 1 Stop bit
- No parity
- No flow control

Protocol

A simple text-based protocol is used. Commands are sent to the SMD3, checked and executed, and a response returned. Wait for a response to the previous command before sending the next to avoid buffer overflow in the SMD3. Commands are in the form (Note that angle brackets are shown for clarity only, they are not part of the protocol):

```
<mnemonic>,<argument 1>,<argument 2>,<argument n>…<CR><LF>
```

And responses are in the form:

```
<SFLAGS>,<EFLAGS>,<data 1>,<data 2>,<data n>…<CR><LF>
```

If the command executed successfully, or:

```
<SFLAGS>,<EFLAGS>,<error code><CR><LF>
```

If the command failed to execute correctly.

Where:

Item	Description
<mnemonic>	

	Short sequence of characters representing a command, case insensitive
<argument n>	Zero or more command arguments
<data n>	Zero or more response data items
<error code>	An error code, see section Error Codes. This includes both a number and text description of the error to aid when using the SMD3 via a terminal program.
<SFLAGS>	Set of flags representing the status of the SMD3, such as the state of the limit inputs or whether the joystick is connected. See section Status Flags
<EFLAGS>	Set of flags representing the error state of the SMD3, such as invalid mnemonic, or motor over-temperature fault. See 0
<CR><LF>	Message terminator; carriage return followed by line-feed (0x07,0x0D)

Comma separation

All elements are comma-separated, except for the message terminator which immediately follows the last item. A response is always sent on receipt of a message terminator. If an argument was supplied with a command, for example, to set a value, the value set will be returned in the response and serves as an additional confirmation of the command having executed as expected.

Many commands accept a real number argument when the underlying quantity is an integer, or finite set of real numbers. In this case, the supplied value being otherwise acceptable is rounded to the closest integer or real number from the allowed set, and it is this value that is returned in the response.

White space

Additional white space (tabs and spaces) are ignored except where they are surrounded by characters comprising the data item, in which case they will be considered as part of the data item itself.

No data items to return

If there are no data items as part of a response, only the SFLAGS and EFLAGS are returned. If an error occurred, then this will be reflected in the EFLAGS.

Argument types

Arguments may be one or a mix of the following types, depending on the command. Data returned by the SMD3 uses the same types, which are always presented as indicated in the “SMD3 response” column.

Type	Name	Description	Example argument values	SMD3 response

INT	Integer	Integer value, with or without sign	100, -10, +7	Sign included for negative values only. E.g. 100, -10
UINT	Unsigned integer	Unsigned integer value, no sign. Hexadecimal representation may also be used, case insensitive	99, 1000, 0x-A74F, 0xd7	Numeric format E.g. 100, 200 Except for status and error flags which are returned in upper case 2-byte hexadecimal format, E.g. 0x1234, 0x-A4DE
FLOAT	Real number	Real number, with or without sign. Scientific format may also be used, case insensitive	10.23, 100e-3, 100E4, 10	Scientific format, with 5 places after the decimal point and a 2-digit exponent E.g. 1.23000E+04, 5.76159E-10
STRING	ASCII string	ASCII string, consisting of characters 0x20 to 0x7E inclusive	Abc123 78-%^A	ASCII string, E.g. “1234 abc”, “10%”
BOOL	Boolean	Binary, true/false value	0, 1	E.g. 0, 1

Flags

Error flags are reported by the SMD3 in hexadecimal format as explained above. E.g. a value of 0x0002 means bit 1 is set (TOPEN), indicating that the SMD3 has been disabled due to an open circuit temperature sensor.

Error flags (EFLAGS)

These indicate error conditions and are latching (i.e. remain set even after the error condition that caused them no longer persists). Clear using the CLR command or by using the reset fault input. The motor is disabled if one or more error flags are set.

Bit	Name	Description

0	TSHORT	Selected temperature sensor is short-circuited (Not applicable to Thermocouple)
1	TOPEN	Selected temperature sensor is open circuit
2	TOVR	Selected temperature sensor is reporting temperature > 190 °C and power has been removed from the motor to protect the windings
3	MOTOR SHORT	Motor phase to phase or phase to ground short has been detected
4	EXTERNAL DISABLE	Motor disabled via external input
5	EMERGENCY STOP	Motor disabled via software
6	CONFIGURATION ERROR	Motor configuration is corrupted
7-15	Reserved	Reserved, read as ‘0’

Status flags (SFLAGS)

Bit	Name	Description
0	JSCON	Joystick is connected
1	LIMIT NEGATIVE	Limit input is active (Note that the polarity is configurable, so active can mean high or low signal level)
2	LIMIT POSITIVE	
3	EXTEN	External enable input state
4	IDENT	Ident mode is active, green status indicator is flashing to aid in identifying SMD3
5	Reserved	Reserved, read as ‘0’
6	STANDBY	Motor stationary. Check this bit before performing a function that requires the motor to be stopped first, such as changing mode
7	BAKE	Bake mode running
8	ATSPEED	

		Set when the motor is at target velocity
9-15	Reserved	Reserved, read as ‘0’

Error codes

Error	Description
-1 (Stop motor first)	Several actions, such as changing resolution or operating mode require that the motor is stopped first. Trying to run such a command before the motor has come to a stop and the standby flag in the status register is set will result in this error.
-2 (Argument validation)	An argument supplied to the command is invalid, for example, it is outside the allowable range.
-3 (Unable to get)	The command is write-only, read is not valid. This applies to commands such as RUNV where a read would have no meaning.
-5 (Action failed)	The command failed to execute due to an internal error, for example, the internal flash in which settings are stored has reached the end of life and data cannot be reliably written to it.
-6 (Not possible in mode)	The command is not applicable to this mode, for example, trying to start bake using RUNB when not in bake mode.
-7 (Not possible when motor disabled)	The motor is disabled (due to a fault, or external enable) and the command is one that starts motion, for example RUNV.
-101 (Argument type)	The argument is of the wrong type, for example a non-integer value was given where an integer value was required.
-102 (Argument count)	The argument count is incorrect, either too few or too many arguments have been supplied.

Quick reference

General

Mnemonic	Description	R	W	Arguments
SER	Read the serial number	●		

FW	Read the firmware version number	●		
CLR	Clear error flags		●	
LOAD	Load saved configuration		●	
STORE	Store configuration		●	
LOADFD	Load factory defaults		●	
IDENT	Identify SMD3 by blinking the status indicator		●	BOOL
MODE	Mode of operation	●	●	UINT
JSMODE	Joystick mode	●	●	UINT
AUTOJS	Auto switch to Joystick mode on JS connect	●	●	BOOL
EXTEN	External enable used	●	●	BOOL
FLAGS	Returns ascii table of status and error flag states	●		

Command movement

Mnemonic	Description	R	W	Arguments
RUNV	Move motor velocity mode		●	STRING
RUNA	Move motor absolute positioning mode		●	INT
RUNR	Move motor relative positioning mode		●	INT
RUNB	Activate bake mode		●	
RUNH			●	STRING

	Start home mode procedure			
STOP	Bring motor to a stop according to the current profile		●	
SSTOP	Stop motor in 1 second on full step position independently of the current motion profile		●	
ESTOP	Emergency stop. Stops the motor immediately		●	

Motor

Mnemonic	Description	R	W	Arguments
TSEL	Temperature sensor selection, T/C or RTD	●	●	UINT
TMOT	Temperature in °C	●		
IR	Run current in amps	●	●	FLOAT
IA	Acceleration current in amps	●	●	FLOAT
IH	Hold current in amps	●	●	FLOAT
PDDEL	Power down delay in milliseconds	●	●	FLOAT
IHD	Power down ramp delay in milliseconds	●	●	FLOAT
F	Freewheel mode	●	●	UINT
RES	Resolution	●	●	UINT

Limit inputs

Mnemonic	Description	R	W	Arguments
L	Global enable	●	●	BOOL
L+	Limit positive (Limit 1) enable	●	●	BOOL
L-	Limit negative (Limit 2) enable	●	●	BOOL
LP+	Limit n polarity (0 for active high, 1 for active low)	●	●	BOOL
LP-		●	●	BOOL
LP	Limit polarity for both Limit positive (Limit 1) and negative (Limit 2), (0 for active high, 1 for active low)		●	BOOL
LSM	How to stop on limit being triggered	●	●	BOOL

Profile

Mnemonic	Description	R	W	Arguments
AMAX	Acceleration in Hz/s	●	●	FLOAT
DMAX	Deceleration in Hz/s	●	●	FLOAT
VSTART	Start frequency in Hz	●	●	FLOAT
VSTOP	Stop frequency in Hz	●	●	FLOAT
VMAX	Target step frequency in Hz	●	●	FLOAT
VACT	Actual frequency in Hz	●		
PACT	Actual position in steps	●	●	FLOAT

PREL	Relative position in steps	●	●	FLOAT
TZW	Time to stop before moving again in seconds	●	●	FLOAT
THIGH	Full step – micro stepping transition	●	●	FLOAT

Step/Direction

Mnemonic	Description	R	W	Arguments
EDGE	Which edges of step input to generate a step on	●	●	UINT
INTERP	Interpolate step input to 256 micro steps	●	●	UINT

Bake

Mnemonic	Description	R	W	Arguments
BAKET	Bake temperature setpoint	●	●	UINT

Command reference

General

IDENT - Blinks status indicator

Set or query enable blinking of that status indicator to aid in identifying the SMD3 among others.

Command: **IDENT**, **Enable**<CR><LF>
Query: **IDENT**<CR><LF>

Arguments

Enable BOOL

The enable state.

[0: Disable]

1:	Enable

Returns

The enable state, as above.

Examples

```
Tx: IDENT,1<CR><LF>                    // Set ident function on
Rx: 0x0000,0x0000,1<CR><LF>
Tx: IDENT<CR><LF>                      // Query state of ident function
Rx: 0x0000,0x0000,1<CR><LF>
```

MODE - Choose mode of operation

Set or query the operating mode. See section Operating Modes for an explanation of each mode.

Command: **MODE**, **Value**<CR><LF>
Query: **MODE**<CR><LF>

Arguments

Value	UINT

The operating mode.

0:	Step/direction
1:	Step/direction triggered velocity
[2:	Remote]
3:	Joystick
4:	Bake
5:	Home

Returns

The mode, as above, followed by a space and the name of the mode in brackets.

Remarks

If the motor is moving when attempting to change the mode, a stop motor first error is returned and the mode is unchanged.

Examples

```
Tx: MODE,2<CR><LF>                     // Set mode to remote
Rx: 0x0000,0x0000,2 (Remote)<CR><LF>
```

```
Tx: MODE<CR><LF>                          // Query state of mode
Rx: 0x0000,0x0000,1 (Remote)<CR><LF>
```

JSMODE – Joystick mode

Set or query the joystick mode. Choose between single step, which allows precise single steps or continuous rotation, or continuous which requires only a single button press to make the motor move.

Command: **JSMODE,** **Mode**<CR><LF>
Query: **JSMODE**<CR><LF>

Arguments

Mode	UINT

The joystick mode.

[0:	Single step]
1:	Continuous

Returns

The mode, as above.

Remarks

Set requires the motor to be in standby, otherwise, a stop motor first error will be returned.

In single step mode, a brief button press (< 0.5 s) will execute one step in that direction, while pressing the button for > 0.5 s will cause the motor to accelerate up to slewing speed and continue to rotate in that direction until the button is released, at which point the motor will decelerate to a stop.

In continuous mode, a brief button press will trigger the motor to accelerate up to slewing speed. A subsequent press of the same button causes it to decelerate to a stop. If, for example, the clockwise button is pressed while the motor is rotating anti-clockwise, the motor will first decelerate to a stop before changing direction.

Examples

```
Tx: JSMODE,1<CR><LF>                      // Set JSMODE to continuous
Rx: 0x0000,0x0000,1<CR><LF>
Tx: JSMODE<CR><LF>                        // Query state of JSMDOE
Rx: 0x0000,0x0000,1<CR><LF>
```

AUTOJS – Auto switch to joystick mode

Set or query auto switching to joystick mode. When enabled the SMD3 switches automatically to joystick mode and reverts to the previous mode on disconnection of the joystick.

Command: **AUTOJS, Enable**<CR><LF>

Query: **AUTOJS**<CR><LF>

Arguments

Enable	BOOL

The enable state.

0:	Disable
[1:	Enable]

Returns

The Enable, as above.

Examples

```
Tx: AUTOJS,1<CR><LF>                    // Set AUTOJS on
Rx: 0x0000,0x0000,1<CR><LF>
Tx: AUTOJS<CR><LF>                      // Query state of AUTOJS function
Rx: 0x0000,0x0000,1<CR><LF>
```

EXTEN – External enable used?

Set or query whether the external enable signal should be used.

Command: **EXTEN, Used**<CR><LF>

Query: **EXTEN**<CR><LF>

Arguments

Used	BOOL

External enable signal.

[0:	False]
1:	True

Returns

True if the external enable signal is used.

Remarks

The external enable input requires a voltage to be applied between SDE COM and EN on the I/O connector which may be inconvenient if you do not wish to use the enable input. In that case, disable the enable input by sending this command with the argument set to false.

Examples

```
Tx: EXTEN,1<CR><LF>                    // Set EXTEN on
Rx: 0x0000,0x0000,1<CR><LF>
Tx: EXTEN<CR><LF>                      // Query state of EXTEN function
Rx: 0x0000,0x0000,1<CR><LF>
```

Command movement

RUNV – Run, velocity

Command to move the motor using velocity mode.

Command: **RUNV, Direction** <CR><LF>

Arguments

Direction	String

Direction velocity motion.

'+':	Positive
'-':	Negative

Remarks

Ensure the profile is set.

Examples

```
Tx: RUNV,+<CR><LF>                     // Start velocity mode to the positive direction
Rx: 0x0000,0x0000<CR><LF>
Tx: RUNV,-<CR><LF>                     // Start velocity mode to the negative direction
Rx: 0x0000,0x0000<CR><LF>
```

RUNA – Run, absolute position

Command to move the motor to an absolute position using the positioning mode.

Command: **RUNA, Absolute** <CR><LF>

Arguments

Absolute	INT
Minimum:	-2^{23}
Maximum:	$2^{23}-1$

Remarks

Ensure the profile is set.

Examples

Tx: RUNA,1000<CR><LF>	// Start absolute positioning mode, move to the step position 1000
Rx: 0x0000,0x0000<CR><LF>	
Tx: RUNA,-1000<CR><LF>	// Start absolute positioning mode to the step position -1000
Rx: 0x0000,0x0000<CR><LF>	

RUNR - Run, relative position

Command to move the motor to a relative position using the positioning mode.

Command: **RUNR, Relative** <CR><LF>

Arguments

Relative	INT
Minimum:	$-2^{23}-1$
Maximum:	$2^{23}-1$

Remarks

Command requires the motor to be in standby, otherwise, a stop motor first error will be returned. Ensure the profile is set.

Examples

Tx: RUNR,2000<CR><LF>	// Start relative positioning mode, move 2000
Rx: 0x0000,0x0000,1<CR><LF>	steps to the positive direction
Tx: RUNR,-2000<CR><LF>	// Start relative positioning mode, move 2000
Rx: 0x0000,0x0000,1<CR><LF>	steps to the negative direction

RUNB – Run bake

Command to start the bake mode.

Command: **RUNB**<CR><LF>

Remarks

Set mode to bake first. To stop the bake mode send the STOP command.

Examples

Tx: RUNB<CR><LF>	// Start bake mode
Rx: 0x0000,0x0000<CR><LF>	
Tx: STOP<CR><LF>	// Stop bake mode
Rx: 0x0000,0x0000<CR><LF>	

RUNH – Home to a limit switch

Command to start the home procedure, in which the motor will move in the specified direction until the limit switch for that direction is triggered, at which point a homing procedure is initiated, see section Limits.

Command: **RUN, Direction** <CR><LF>

Arguments

Direction	String

Direction velocity motion.

'+':	Positive
'-':	Negative

Remarks

Ensure the profile is set and the mode is Home mode.

Examples

```
Tx: RUNH,+<CR><LF>                    // Start homing mode to the positive direction
Rx: 0x0000,0x0000<CR><LF>
Tx: RUNH,-<CR><LF>                    // Start homing mode to the negative direction
Rx: 0x0000,0x0000<CR><LF>
```

STOP – Stop motor

Command motor to stop moving according to the current profile.

Command: **STOP**<CR><LF>

Remarks

During the deceleration phase that stops the motor, any modifications to the acceleration or deceleration interrupt the stopping phase. Re-send the command to restart the motor stopping phase.

Examples

```
Tx: STOP<CR><LF>                      // Stop the motor
Rx: 0x0000,0x0000<CR><LF>
```

SSTOP – Stop motor in 1 s

Command motor to stop the motion in 1 second.

Command: **SSTOP**<CR><LF>

Remarks

This command does not consider the deceleration set in the profile. Instead, it calculates the deceleration required to stop in 1 second, according to the actual velocity. The motor will stop in a full step position. Steps may be lost if the load requires greater than this duration to stop.

Examples

```
Tx: SSTOP<CR><LF>                     // Stop the motor in 1 seconds
Rx: 0x0000,0x0000<CR><LF>
```

ESTOP – Emergency stop

Command stops immediately and disables the motor. Note that this should not be relied on as a safety interlock.

Command: **ESTOP**<CR><LF>

Remarks

The motor may stop on a fractional step position, but this is irrelevant as motor power is removed and the motor will snap to a full step position. Steps may be lost.

Examples

```
Tx: ESTOP<CR><LF>                    // Stop the motor immediately
Rx: 0x0000,0x0000<CR><LF>
```

Motor

TSEL – Temperature sensor selection

AML motors can be ordered with a K-Type thermocouple or a PT100 RTD. Select the correct option for your motor.

Command: **TSEL, SensorType**<CR><LF>

Query: **TSEL**<CR><LF>

Arguments

SensorType	UINT

Motor temperature sensor type.

[0:	Thermocouple]
1:	RTD

Returns

Selected temperature sensor type, as above.

Remarks

The drive will not allow the motor to run unless a functioning temperature sensor is connected to the selected sensor connection; be sure to select the correct type.

Examples

```
Tx: TSEL,0<CR><LF>                          // Select thermocouple sensor
Rx: 0x0000,0x0000,0<CR><LF>
Tx: TSEL<CR><LF>                            // Queue the state of temperature sensor
Rx: 0x0000,0x0000,0<CR><LF>
```

TMOT – Motor temperature

Query the motor temperature.

Query: **TMOT**<CR><LF>

Returns

Motor temperature in °C, rounded to the nearest 1 °C.

Remarks

The reported temperature is intended only for the purposes of monitoring motor temperature and should not be relied upon for any other purpose within the vacuum system.

Examples

```
Tx: TMOT<CR><LF>                            // Query motor temperature
Rx: 0x0000,0x0000,25<CR><LF>                // Response is 25 degree Celsius
```

IR – Run current

Set or query the motor run current.

Command: **IR, RunCurrent**<CR><LF>
Query: **IR**<CR><LF>

Arguments

RunCurrent	FLOAT

The motor run current in amps rms.

[Default:	1.044]
Minimum:	0
Maximum:	1.044

Returns

The motor run current in amps rms, rounded to the closest multiple of 1.044 A / 31 (approx. 33 mA).

Remarks

IR must be set equal to or smaller than IA. This is enforced by the SMD3; IA is automatically adjusted to be equal to IR, if a change to IR makes it greater than IA.

Examples

```
Tx: IR,1<CR><LF>                          // Set run current to 1A
Rx: 0x0000,0x0000,1.0000E+00<CR><LF>
Tx: IR<CR><LF>                            // Query run current
Rx: 0x0000,0x0000,1.0000E+00<CR><LF>
```

IA – Acceleration current

Set or query the motor acceleration/deceleration current.

Command: **IA, AccCurrent**<CR><LF>
Query: **IA**<CR><LF>

Arguments

AccCurrent	FLOAT

The motor acceleration current in amps rms.

[Default:	1.044]
Minimum:	0
Maximum:	1.044

Returns

The motor acceleration current in amps rms, rounded to the closest multiple of 1.044 A / 31 (approx. 33 mA).

Remarks

IA must be set equal to or greater than IR. The SMD3 will not force IA to match IR if IA is smaller than IR.

Examples

```
Tx: IA,1.044<CR><LF>                          // Set acceleration current to 1.044 A
Rx: 0x0000,0x0000,1.0440E+00<CR><LF>
Tx: IA<CR><LF>                                // Query acceleration current
Rx: 0x0000,0x0000,1.0440E+00<CR><LF>
```

IH – Hold current

Set or query the motor hold current. If your application allows it, set PDDEL, IHD and IH to zero in order to reduce run current to zero as quickly as possible after stopping which minimises motor temperature rise.

Command: **IH,** **HoldCurrent**<CR><LF>
Query: **IH**<CR><LF>

Arguments

HoldCurrent	FLOAT

The motor hold current in amps rms.

[Default:	0.1]
Minimum:	0
Maximum:	1.044

Returns

The motor hold current in amps rms, rounded to the closest multiple of 1.044 A / 31 (approx. 33 mA).

Examples

```
Tx: IH,0.5<CR><LF>                            // Set hold current to 0.5A
Rx: 0x0000,0x0000,5.0000E-01<CR><LF>
Tx: IH<CR><LF>                                // Query hold current
Rx: 0x0000,0x0000,5.0000E-01<CR><LF>
```

PDDEL – Power down delay

Set or query the delay time between standstill occurring and the motor current being reduced from the acceleration current to the hold current. Refer to Figure 1. If your application allows it, set PDDEL, IHD and

IH to zero in order to reduce run current to zero as quickly as possible after stopping which minimises motor temperature rise.

Command: **PDDEL, Duration**<CR><LF>
Query: **PDDEL**<CR><LF>

Arguments

Duration	FLOAT

The power-down delay in milliseconds.

[Default:	0]
Minimum:	0
Maximum:	5570

Returns

The power-down delay rounded to the closest settable value.

Examples

```
Tx: PDDEL,100<CR><LF>                          // Set PDDEL to 100 ms
Rx: 0x0000,0x0000,1.0000E+02<CR><LF>
Tx: PDDEL<CR><LF>                              // Query PDDEL
Rx: 0x0000,0x0000, 1.0000E+02<CR><LF>
```

IHD – Current reduction delay

Set or query the delay per current reduction step that occurs when run current is reduced to hold current. See Figure 1. If your application allows it, set PDDEL, IHD and IH to zero in order to reduce run current to zero as quickly as possible after stopping which minimises motor temperature rise.

Command: **IHD, Duration**<CR><LF>

Query: **IHD**<CR><LF>

Arguments

Duration	FLOAT

The delay per current reduction step in milliseconds.

[Default:	0]
Minimum:	0

Maximum:	327

Returns

The delay per current reduction step.

Remarks

See also section Going to standby

Examples

```
Tx: IHD,327<CR><LF>                          // Set IHD to 327 ms
Rx: 0x0000,0x0000,3.2700E+02<CR><LF>
Tx: IHD<CR><LF>                              // Query IHD
Rx: 0x0000,0x0000,3.2700E+02<CR><LF>
```

F – Freewheel mode

Set or query the freewheel mode. Set the option to use passive braking or freewheeling when the motor is in standby. This feature can be enabled when 'IH' is 0. The desired option becomes active after a time period specified by ‘PDDEL’ and ‘IHD’

Command: **FW, Mode**<CR><LF>

Query: **FW**<CR><LF>

Arguments

Mode	UINT

The freewheel mode:

0:	Normal
1:	Freewheel
[2:	Phases shorted to GND]

Returns

The freewheel mode selection, as above.

Remarks

Use the freewheel mode to allow the motor shaft to spin freely when the motor current is zero. The phases shorted to GND option supplies no power to the motor, but by shorting the phases together a holding

torque is produced, and the motor shaft offers considerable resistance to movement. This is enough in many applications to remove the need for any holding current, with the benefit that no heat is generated because the motor phases are not energised.

Examples

```
Tx: F,1<CR><LF>                    // Set F to freewheel
Rx: 0x0000,0x0000,1<CR><LF>
Tx: F<CR><LF>                      // Query state of F
Rx: 0x0000,0x0000,1<CR><LF>
```

RES - Resolution

Set or query the microstep resolution. Although the drive may only stop on full step positions in all modes except step/direction, microstepping is still desirable as it reduces resonances for slow movements. Note that full step resolution is always used above a specified stepping rate, regardless of the resolution set here, see command [THIGH].

Command: **RES, Resolution**<CR><LF>

Query: **RES**<CR><LF>

Arguments

Resolution	UINT

The microstep resolution as an integer.

[Default:	256]
Possible values:	8, 16, 32, 64, 128, 256

Returns

The microstep resolution, as above.

Remarks

Query is applicable any time, Set requires the motor in standby condition.

The resolution applies globally, including for the step/direction interface. Each step on the step/direction interface generates a 1/8, 1/16, 1/32 etc step according to the resolution set here.

Above a configurable step frequency, the drive switches from the microstepping resolution specified here to full step mode in any case. See section THIGH

Examples

Tx: RES,256<CR><LF>	// Set resolution to 256
Rx: 0x0000,0x0000,256<CR><LF>	
Tx: RES<CR><LF>	// Query resolution
Rx: 0x0000,0x0000,256<CR><LF>	

Limit inputs

L – Limits global enable

Set or query global enable of limits inputs. This does not affect other limits configuration settings, allowing limits to be configured as desired, then globally enabled or disabled if required.

Command:	**L, Enabled**<CR><LF>
Query:	**L**<CR><LF>

Arguments

Enabled	BOOL

Enable state of limits.

[0:	Disable]
1:	Enable

Returns

True if limits are globally enabled.

Remarks

This option globally enables or disabled limits; remaining limits settings remain unchanged.

Examples

Tx: L,0<CR><LF>	// Set global enable of limits to disable
Rx: 0x0000,0x0000,0<CR><LF>	
Tx: L<CR><LF>	// Query state of global enable of limits
Rx: 0x0000,0x0000,0<CR><LF>	

L+, L- Individual limit enable

Set or query enable of Lx, where ‘x’ is '+’ or ‘-’.

Command:	**Lx, Enabled**<CR><LF>
Query:	**Lx**<CR><LF>

Arguments

Enabled	BOOL

Enable state of limit n.

0:	Disable
[1:	Enable]

Returns

True if limit n is enabled.

Remarks

L+ refers to LIMIT 1, associated with movement resulting in incrementing of the position and L- to LIMIT 2, associated with movement decrementing the position counter.

Examples

```
Tx: L+,1<CR><LF>                    // Set positive limit enable
Rx: 0x0000,0x0000,1<CR><LF>
Tx: L+<CR><LF>                      // Query state of positive limit enable
Rx: 0x0000,0x0000,1<CR><LF>
```

LP+, LP- Individual limit polarity

Set or query the polarity of LPx, where ‘x’ is ‘+’ or ‘-’. Limits are active low by default; use this option to make the limit active low.

Command:	**LPx, ActiveLow**<CR><LF>
Query:	**LPx**<CR><LF>

Arguments

ActiveLow	BOOL

Polarity of LPx.

[0:	Active high]
1:	Active low

Returns

The polarity of LPx, as above.

Remarks

LP+ refers to Polarity of LIMIT 1 and LP- to Polarity of LIMIT 2.

Examples

```
Tx: LP-,1<CR><LF>                    // Set negative limit polarity to active low
Rx: 0x0000,0x0000,1<CR><LF>
Tx: LP-<CR><LF>                      // Query state of negative limit polarity
Rx: 0x0000,0x0000,1<CR><LF>
```

LP – Global limit polarity

Set the polarity for both L+ and L-. Limits are active high by default; use this option to make the limit active low.

Command: **LP, ActiveLow**<CR><LF>

Arguments

ActiveLow	BOOL

Polarity of LP.

[0:	Active high]
1:	Active low

Returns

The polarity of LP, as above.

Remarks

LP+ refers to Polarity of LIMIT 1 and LP- to Polarity of LIMIT 2.

Examples

```
Tx: LP,1<CR><LF>                        // Set negative and positive limit polarity to active
Rx: 0x0000,0x0000,1<CR><LF>             low
Tx: LP-<CR><LF>
Rx: 0x0000,0x0000,1<CR><LF>             // Query state of negative limit polarity
Tx: LP+<CR><LF>
Rx: 0x0000,0x0000,1<CR><LF>             // Query state of positive limit polarity
```

LSM – Limit stop mode

Set or query the stop mode; determines behaviour when a limit is triggered.

Command:	**LSM,** Mode<CR><LF>
Query:	**LSM**<CR><LF>

Arguments

Mode	BOOL

The stop mode.

[0:	Hard stop; the motor will stop immediately on a being triggered]
1:	Soft stop; the motor decelerates according to th file

Returns

The stop mode, as above.

Remarks

When using hard stop, keep in mind that steps may be lost depending on the slewing speed and load on the motor. Treat position counters with caution until the true position has been established. Conversely, when using soft stop, ensure that the motor can decelerate to a stop before the physical end of travel is reached and steps are lost.

Examples

```
Tx: LSM,1<CR><LF>                       // Set limits stop mode to soft stop
Rx: 0x0000,0x0000,1<CR><LF>
```

```
Tx: LSM<CR><LF>                    // Query state of limits stop mode
Rx: 0x0000,0x0000,1<CR><LF>
```

Profile

AMAX - Acceleration

Set or query the acceleration, in Hz/s (steps per second per second)

Command: **AMAX, Acceleration**<CR><LF>

Query: **AMAX**<CR><LF>

Arguments

Acceleration FLOAT

The acceleration in Hz/s.

[Default: 5000]

Minimum: $\frac{65.48362}{Resolutio}$

Where resolution is the microstep resolution 8, 32, 64, 128 or 256

Maximum: $(2^{16} - 1) \times \frac{65.4836}{Resolutic}$

Where resolution is the microstep resolution 8, 32, 64, 128 or 256

Returns

The user-defined AMAX (data 1) and the real value after the conversion (data 2).

Remarks

Notice that the maximum acceleration depends on the motor resolution. Therefore, when changing resolution, the SMD3 validates the acceleration value and may change it if necessary, to constrain it according to the above equation.

Examples

Tx: AMAX,150<CR><LF>	// Set acceleration to 150Hz/s
Rx: 0x0000,0x0000,1.5000E+02,1.4988E+02<CR><LF>	// Note that the target value of 150 has been adjusted to the closest real value, which deviates from the requested value by 0.12 Hz/s
Tx: AMAX<CR><LF>	
Rx: 0x0000,0x0000,1.5000E+02,1.4988E+02<CR><LF>	

DMAX - Deceleration

Set or query the deceleration, in Hz/s (steps per second per second)

Command: **DMAX, Deceleration**<CR><LF>

Query: **DMAX**<CR><LF>

Arguments

Deceleration FLOAT

The deceleration in Hz/s.

[Default: 5000]

Minimum:

$$\frac{65.48362}{Resolutio}$$

Where resolution is the microstep resolution 8, 32, 64, 128 or 256

Maximum:

$$(2^{16} - 1) \times \frac{65.4836}{Resolutio}$$

Where resolution is the microstep resolution 8, 32, 64, 128 or 256

Returns

The user-defined DMAX (data 1) and the real value after the conversion (data 2).

Remarks

Notice that the maximum deceleration depends on the motor resolution. Therefore, when changing resolution, the SMD3 validates the deceleration value and may change it if necessary, to constrain it according to the above equation.

Examples

```
Tx: DMAX,150<CR><LF>                                  // Set deceleration to 150Hz/s
Rx:
0x0000,0x0000,1.5000E+02,1.4988E+02<CR><L             // Query deceleration
F>
Tx: DMAX<CR><LF>
Rx:
0x0000,0x0000,1.5000E+02,1.4988E+02<CR><L
F>
```

VSTART – Start frequency

Set or query the start frequency in Hz.

The start frequency is the initial step rate, and helps to allow the motor to overcome inertia and start moving smoothly; if start frequency were zero, the duration of the initial few steps might be long enough that the motor would overcome inertia on the first step, then effectively stop for a time, then have to overcome inertia once more for the second step, and so on, until the steps were frequent enough that the motor remains moving.

Command:	**VSTART,** StartFrequency<CR><LF>
Query:	**VSTART**<CR><LF>

Arguments

StartFrequency	FLOAT

The start frequency in Hz.

[Default:	10]
Minimum:	0
Maximum:	15 kHz, when resolution is 8.

$$(2^{18}-1)\times\frac{0.7152}{Resoluti}$$

When resolution is 16, 32, 64, 128 or 256.

Returns

The user-defined VSTART (data 1) and the real value after the conversion (data 2).

Remarks

VSTART must be set equal to or less than VSTOP. This is enforced by the SMD3; if a change to VSTART makes it bigger than VSTOP, VSTOP is automatically adjusted to be equal to VSTART.

VSTART must be set equal to or less than VMAX. The SMD3 will not force VSTART to match VMAX if VSTART is greater than VMAX.

Examples

```
Tx: VSTART,0<CR><LF>                          // Set VSTART to 0 Hz
Rx:
0x0000,0x0000,0.0000+00,0.0000+00<CR><LF>     // Query VSTART
Tx: VSTART<CR><LF>
Rx:
0x0000,0x0000,0.0000+00,0.0000+00<CR><LF>
```

VSTOP – Stop frequency

Set or query the stop frequency in Hz.

The stop frequency is the frequency at which the deceleration ramp ends; i.e. the deceleration ramp does not go from the target frequency linearly down to 0, but from the target frequency linearly down to the stop frequency.

Command:	**VSTOP, StopFrequency**<CR><LF>
Query:	**VSTOP**<CR><LF>

Arguments

StopFrequency	FLOAT

The stop frequency in Hz.

[Default:	10]
Minimum:	1
Maximum:	15 kHz, when resolution is 8.

$$(2^{18} - 1) \times \frac{0.7152}{Resoluti}$$

When resolution is 16, 32, 64, 128 or 256.

Returns

The user-defined VSTOP (data 1) and the real value after the conversion (data 2).

Remarks

VSTOP must be set equal to or greater than VSTART. This is enforced by the SMD3; if a change to VSTOP makes it smaller than VSTART, VSTART is automatically adjusted to be equal to VSTOP.

VSTOP must be set equal to or less than VMAX. The SMD3 will not force VSTOP to match VMAX if VSTOP is greater than VMAX.

Examples

```
Tx: VSTOP,10<CR><LF>                          // Set VSTOP to 10 Hz
Rx:
0x0000,0x0000,1.0000+01,9.9996+00<CR><LF>     // Query VSTOP
Tx: VSTOP<CR><LF>
Rx:
0x0000,0x0000,1.0000+01,9.9996+00<CR><LF>
```

VMAX – Step frequency

Set or query the target frequency, in Hz. This is the maximum speed the motor will be run at. The target frequency will only be reached if there is enough time or distance to do so; if moving for a short time, for example, the motor may only accelerate to some fraction of the target frequency before it is time to decelerate to a stop.

Command:	**VMAX, TargetFrequency**<CR><LF>
Query:	**VMAX**<CR><LF>

Arguments

TargetFrequency	FLOAT

The target frequency in Hz.

[Default:	1 kHz]
Minimum:	1 Hz

Maximum: 15 kHz

Returns

The user-defined VMAX (data 1) and the real value after the conversion (data 2).

Remarks

Motor torque decreases with speed, and each motor will have a different maximum frequency that it can achieve while reliably maintaining synchronicity (when synchronicity is lost, the motor fails to complete the steps that it is commanded to, leading to a difference between the true and actual positions), depending on the load it is driving.

VMAX must be set equal to or greater than VSTART and VSTOP. The SMD3 will not force VMAX to match VSTART and VSTOP if VMAX is smaller than VSTART and VSTOP.

Examples

```
Tx: VMAX,1000<CR><LF>                          // Set VMAX to 1 kHz
Rx:
0x0000,0x0000,1.0000E+03,1.0000E+03<CR><L      // Query VMAX
F>
Tx: VMAX<CR><LF>
Rx:
0x0000,0x0000,1.0000E+03,1.0000E+03<CR><L
F>
```

VACT – Actual frequency

Query the actual frequency (the frequency at which the motor is currently spinning) in Hz (steps per second).

Query: **VACT**<CR><LF>

Returns

The frequency at which the motor is spinning in Hz.

Remarks

This value is derived from the stepper motor control logic, there is no feedback from the motor itself. Hence, the motor could be stalled while VACT continues to indicate the expected.

Examples

```
Tx: VACT<CR><LF>                          // Query state of blink
Rx: 0x0000,0x0000,1.0000E+03<CR><LF>
```

PACT – Actual position

Set or query the actual position in steps.

The usual way to position the motor is to initialise the actual position to some reference value, usually 0, then adjust the target position to move the motor. In this way, by setting RUNA to 0 the motor can be homed to the initial 0 position. If you wish to perform relative movements, while still retaining an absolute reference, see PREL command.

Command:	**PACT, ActualPosition**<CR><LF>
Query:	**PACT** <CR><LF>

Arguments

TargetPosition	INT

The target position in steps.

[Default:	0]
Minimum:	-2^{23}
Maximum:	$2^{23}-1$

Returns

The absolute position, as above.

Remarks

Query is applicable any time, Set requires the motor in standby condition.

Examples

```
Tx: PACT<CR><LF>                          // Query actual position
Rx: 0x0000,0x0000,1000.00<CR><LF>
Tx: PACT,0<CR><LF>                        // Set actual position 0
Rx: 0x0000,0x0000,0.00<CR><LF>
```

PREL – Relative position

Set or query the relative position in steps.

Use this function to perform relative movement, while still retaining reference to absolute position via PACT. Set the desired value then use the RUNR command to initiate movement.

Command:	**PREL, RelativeDisplacement**<CR><LF>
Query:	**PREL** <CR><LF>

Arguments

RelativeDisplacement	INT

The target position in steps.

[Default:	0]
Minimum:	-2^{23}
Maximum:	$2^{23}-1$

Returns

The relative position, as above.

Remarks

Query is applicable any time. Set requires the motor in standby condition.

Examples

```
Tx: PREL<CR><LF>                    // Query relative position
Rx: 0x0000,0x0000,1000.00<CR><LF>
Tx: PREL,0<CR><LF>                  // Set relative position 0
Rx: 0x0000,0x0000,0.00<CR><LF>
```

TZW – Zero wait time

Set or query the waiting time after ramping down to a stop before the next movement can start.

When using higher values for the start and stop frequency, a subsequent move in the opposite direction would result in a jerk equal to VSTART + VSTOP. The motor may not be able to follow this. TZW can be used to introduce a short delay between the two and eliminate the jerk.

Command:	**TZW, Duration**<CR><LF>
Query:	**TZW** <CR><LF>

Arguments

Duration FLOAT

The waiting time in milliseconds.

[Default:	0]
Minimum:	0
Maximum:	2796

Returns

The zero wait time, as above.

Examples

```
Tx: TZW,100<CR><LF>                          // Set TZW to 100 ms
Rx: 0x0000,0x0000,1.0000E+02<CR><LF>
Tx: TZW<CR><LF>                              // Query TZW
Rx: 0x0000,0x0000,1.0000E+02<CR><LF>
```

THIGH – Microstep transition

Set or query the full step / microstepping transition. When frequency falls below this threshold (approximately), the motor switches from full step to the selected microstep resolution. The SMD3 determines the upper threshold automatically and applies hysteresis to avoid possible jitter between the two stepping modes. The upper threshold cannot be adjusted.

Command:	**THIGH, Threshold**<CR><LF>
Query:	**THIGH** <CR><LF>

Arguments

Threshold FLOAT

Threshold in frequency Hz.

[Default:	10000 Hz]
Minimum:	1 Hz
Maximum:	15000 Hz

Returns

The user-defined THIGH (data 1) and the real value after the conversion (data 2)

Remarks

The SMD3 software calculates and displays the upper threshold value for reference, although as noted above it cannot be adjusted.

Examples

```
Tx: THIGH,500<CR><LF>                          // Set THIGH threshold to 500 Hz
Rx:
0x0000,0x0000,5.0000E+02,5.0400E+02<CR><L      // Query THIGH
F>
Tx: THIGH<CR><LF>
Rx:
0x0000,0x0000,5.0000E+02,5.0400E+02<CR><L
F>
```

Step/Direction

EDGE – Edge to step on

Set or query a value indicating whether a step occurs on both the rising and falling edges of the step input, or just the rising edge.

Command:	**EDGE, Both**<CR><LF>
Query:	**EDGE** <CR><LF>

Arguments

Both	BOOL

Step on both edges.

[0: Rising edge only; a step occurs only on the rising edge]

1: Both; a step occurs on both rising and falling edges

Returns

True if step input is configured to step on both rising and falling edges, as above.

Remarks

Enabling this feature halves the clock rate required to achieve a chosen step rate. The EDGE command is disabled in any other modes than step/direction mode.

Examples

```
Tx: EDGE,1<CR><LF>                    // Set EDGE to rising edge
Rx: 0x0000,0x0000,1<CR><LF>
Tx: EDGE<CR><LF>                      // Query EDGE
Rx: 0x0000,0x0000,1<CR><LF>
```

INTERP – Step interpolation

Set or query a value indicating whether the step input should be interpolated to 256 microsteps.

Command:	**INTERP, Interpolate**<CR><LF>
Query:	**INTERP**<CR><LF>

Arguments

Interpolate	BOOL

Enable interpolation of step input to 256 microsteps.

[0:	Normal; each step input will cause one step at t current resolution]
1:	Interpolate; each step input will be interpolated 256 microsteps.

Returns

True if interpolation mode is active, as above.

Remarks

The INTERP command is useful in step/direction mode. Enabling this feature affords the benefits of high-resolution microstepping, without the drawback of very high step clock rates. Internal logic tracks the rate at which steps are supplied and smooths them out into 256 microsteps.

Examples

```
Tx: INTERP,1<CR><LF>                  // Set interpolation to 256 microstep on
Rx: 0x0000,0x0000,1<CR><LF>
Tx: INTERP<CR><LF>                    // Query INTERP
Rx: 0x0000,0x0000,1<CR><LF>
```

BAKET – Bake temperature setpoint

Set or query the bake temperature setpoint. To run bake, select bake mode using the MODE, then start bake using RUNB. Use STOP to end bake.

Command:	**BAKET, Setpoint**<CR><LF>
Query:	**BAKET** <CR><LF>

Arguments

Setpoint	UINT

Bake temperature setpoint.

[Default:	150 °C]
Minimum:	0 °C
Maximum:	200 °C

Returns

Bake temperature setpoint in °C, as above.

Examples

```
Tx: BAKET,100<CR><LF>                    // Set bake setpoint to 100 °C
Rx: 0x0000,0x0000,100<CR><LF>
Tx: BAKET<CR><LF>                        // Query BAKET
Rx: 0x0000,0x0000,100<CR><LF>
```