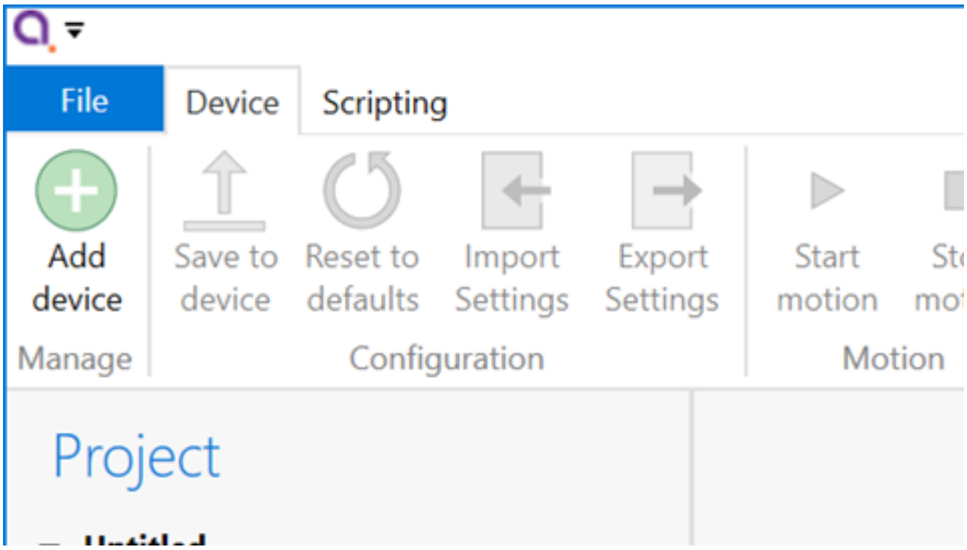
Software

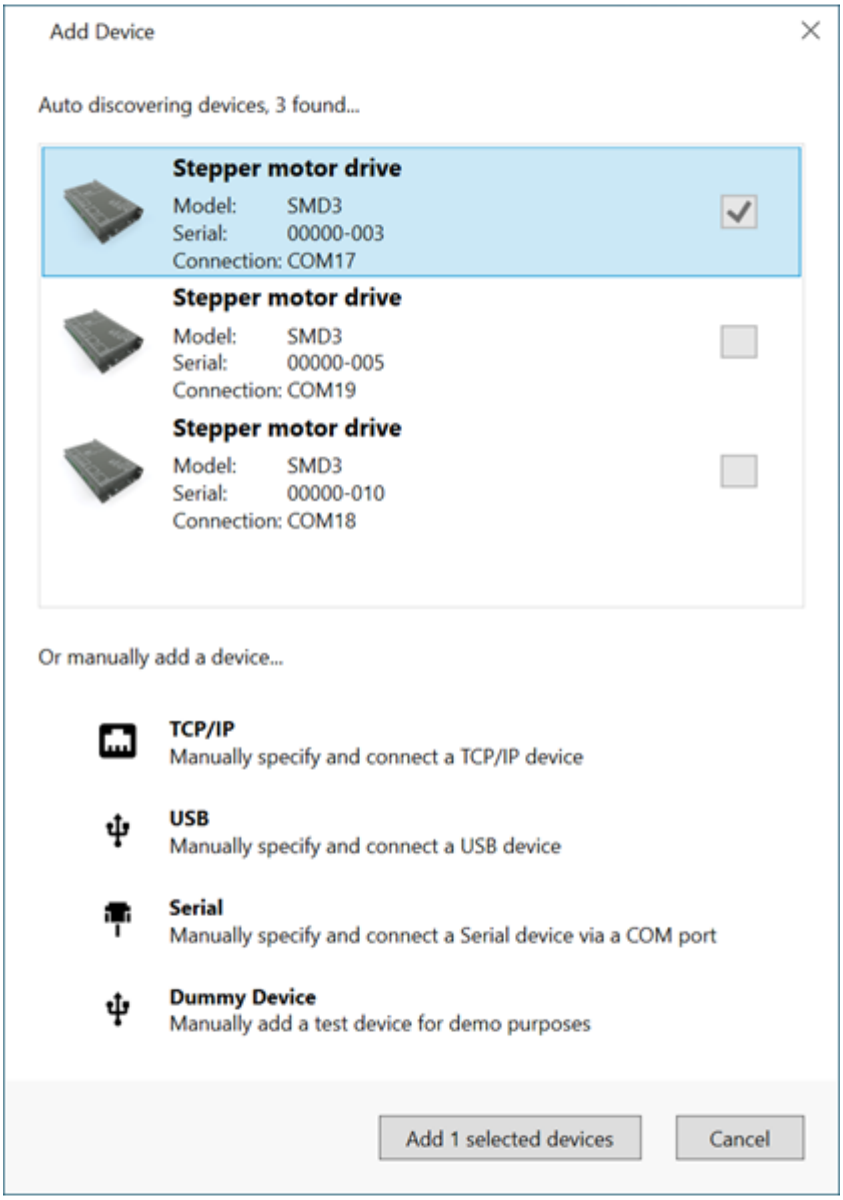
Installation and setup

The SMD3 is compatible with the AML Device Control software, which can be downloaded from the Software page on our website: https://arunmicro.com/documents/software/

- Connect all SMD3 devices to your computer, using a USB lead, and power them on.
- Start the AML Device Control software and click ‘Add device’ in the top left corner



USB connected SMD3 devices should automatically appear in the list. Select all devices that you wish to add and click “Add n selected devices”



Add Device

Auto discovering devices, 3 found...

Stepper motor drive
Model: SMD3
Serial: 00000-003
Connection: COM17
☑

Stepper motor drive
Model: SMD3
Serial: 00000-005
Connection: COM19
□

Stepper motor drive
Model: SMD3
Serial: 00000-010
Connection: COM18
□

Or manually add a device...

TCP/IP
Manually specify and connect a TCP/IP device

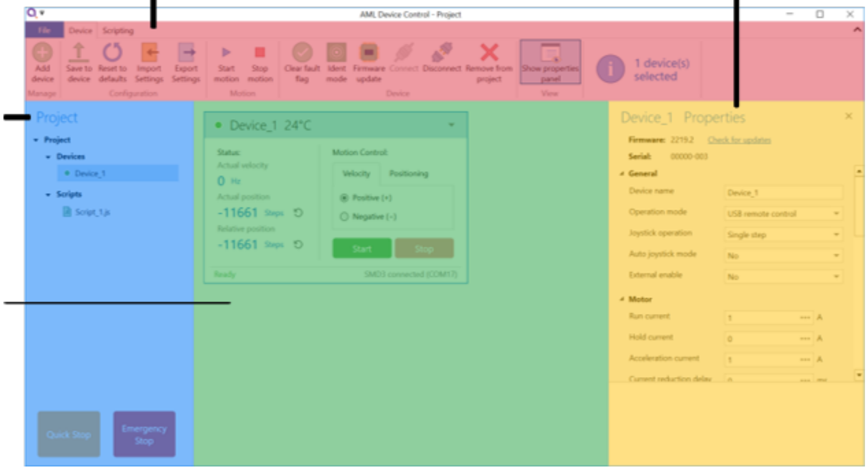
USB
Manually specify and connect a USB device

Serial
Manually specify and connect a Serial device via a COM port

Dummy Device
Manually add a test device for demo purposes

Overview

The default layout of the software is shown below.



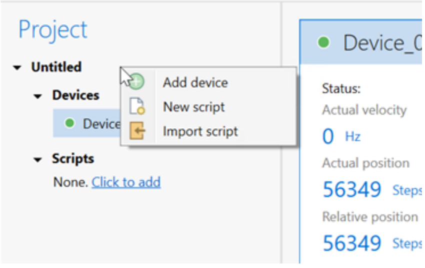
Ribbon
Buttons for device and scripting actions, and access to the "File" menu

Device properties panel
View and edit configuration for selected device(s).

Project panel
Add, remove and select devices or scripts. Re-arrange items by dragging.

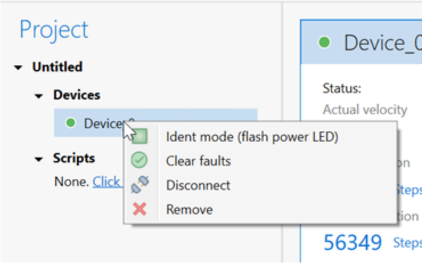
System work area
Controller windows for each device will appear here. They can be re-arranged by dragging.

Project panel



Shows a list of the devices and scripts in the project. Currently selected devices are highlighted. Multiple devices can be selected by holding down CTRL and clicking. The device properties panel shows the properties for the selected device(s).

Right-clicking on an empty area within the project panel presents a context menu, with options to add new devices, new scripts or importing scripts.



The right-click context menu on each device provides access to functions such as clearing faults or placing the selected device into ident mode in which the green status indicator flashes.

A status indicator next to each device shows the current status of each device according to these colours:

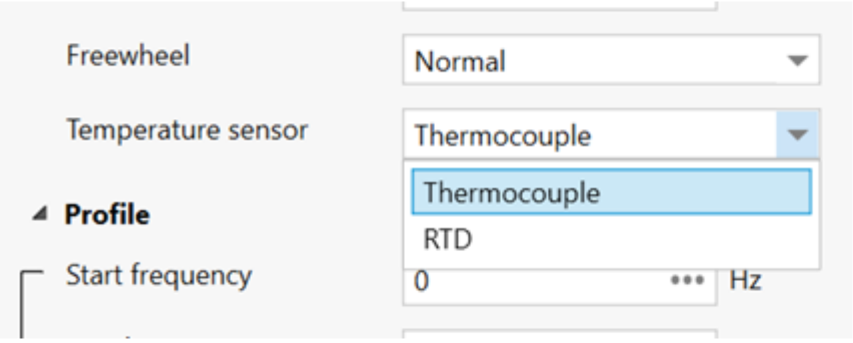
Colour	Description
🟢	Device connected and ready
🔵	stick connected

🔴	Device in a fault state
●	Device disconnected

Device properties panel

Select one or more devices in the project panel; their properties are displayed and can be edited here. A blank is shown for properties which are different across the selected devices. As each property is selected, help text appears at the bottom of the panel describing the configuration option in more detail.

- Some properties allow selection of one of several choices, for example, temperature sensor selection:



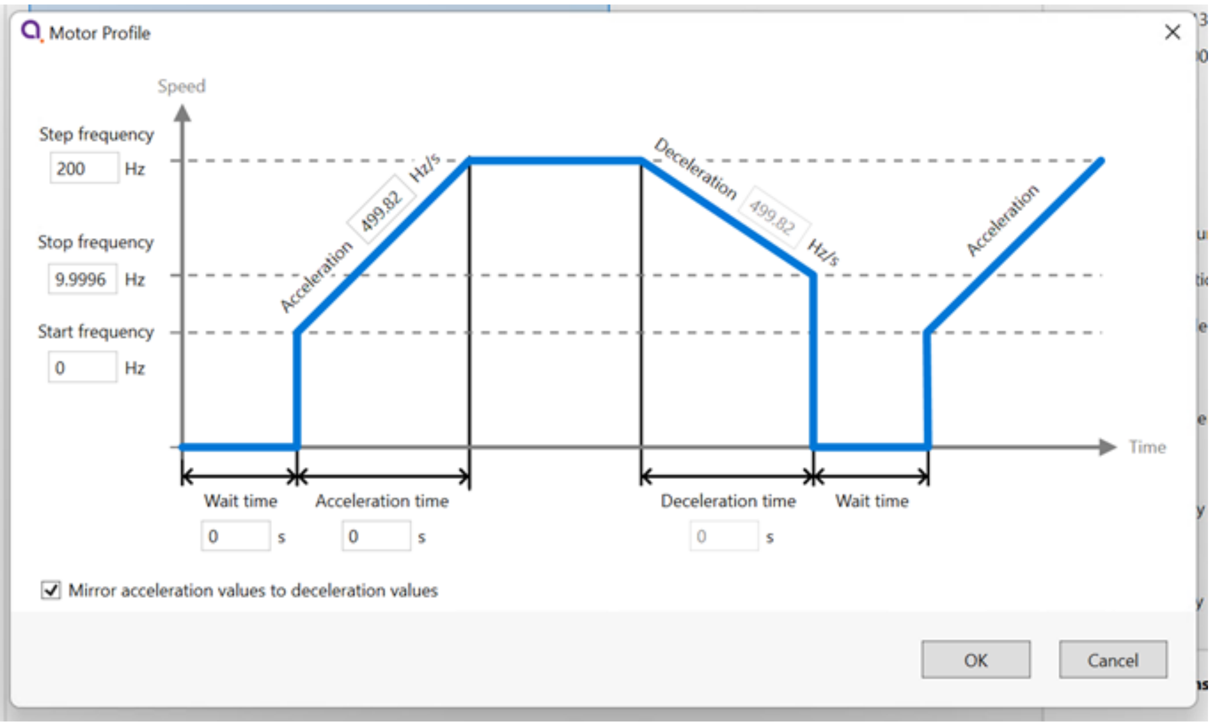
Freewheel: Normal

Temperature sensor: Thermocouple

Profile

Start frequency: 0 Hz

- Others such as ‘Run current’ simply require a numeric value to be entered
- Others allow values to be entered directly, or the three dots button to the right to be clicked. This opens a window, allowing a more complex value to be input, for example, ‘Acceleration’ allows input in the native units or seconds:



Motor Profile

Step frequency: 200 Hz

Stop frequency: 9.9996 Hz

Start frequency: 0 Hz

Wait time: 0 s

Acceleration time: 0 s

Deceleration time: 0 s

☑ Mirror acceleration values to deceleration values

System work area

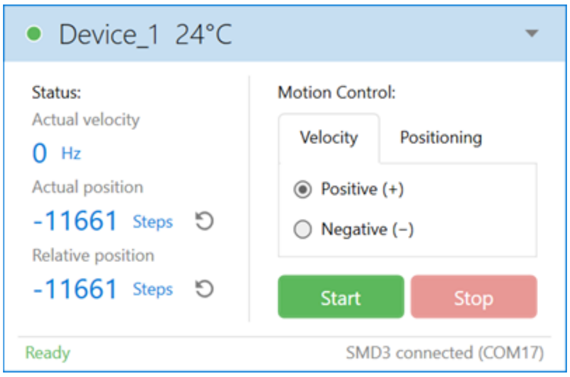
Controller windows for each device appear in this area. Windows can be arranged as desired and will automatically ‘snap’ to a grid making it easy to keep them neatly organised.

Controller window

Shows a status summary for the selected device, providing essential information such as actual velocity, actual (absolute) position, relative position and error status.

Absolute and relative position counters may be reset using the ⟲ icons.

For controlling an SMD3, choose the type of motion and click start or stop. Multiple SMD3 devices can be controlled using the motion controls on the ribbon:

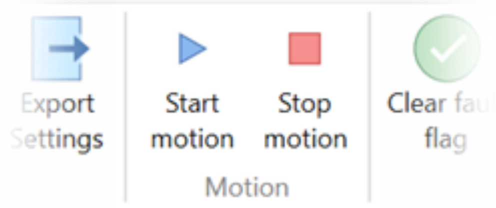


● Device_1 24°C

Status:
Actual velocity
0 Hz
Actual position
-11661 Steps
Relative position
-11661 Steps

Motion Control:
Velocity | Positioning
● Positive (+)
○ Negative (−)

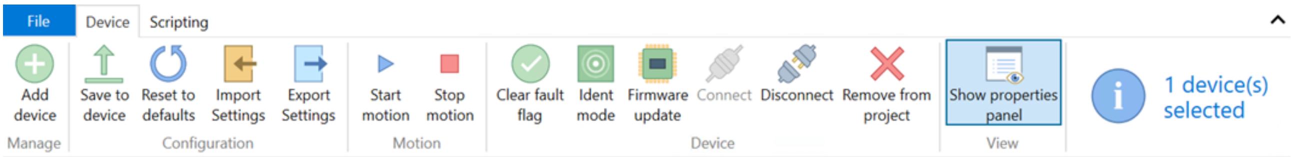
Ready
SMD3 connected (COM17)



INFORMATION: Be aware that the synchronisation between multiple SMD3s is not specified or guaranteed when controlling them in this way; delays within the computer, software, and USB connection to the SMD3 mean that each SMD3 will start or stop its motion at a slightly different moment, therefore this option is not suitable for performing complex co-ordinated movements across multiple axes.

Ribbon

Contains buttons for all actions. Within the software, buttons can be hovered over for more information.



Saving projects

SMD3 configuration is maintained in two locations:

1. The SMD3 itself, with the use of the ”Save to device” command. If the “Save to device” command is not used, settings will revert to their previous values on power cycling.
2. In the software project file.

The behaviour of the software in relation to this is as follows:

If the serial number of a connected SMD3 matches that of one in the project file that is open: The configuration given in the project file prevails, and the SMD3 configuration is synchronised to match that in the project file. Note that unless you use the save changes to hardware function, the original configuration of the SMD3 is not overwritten, and will be restored on power cycle or by using the load command to restore the configuration from flash.

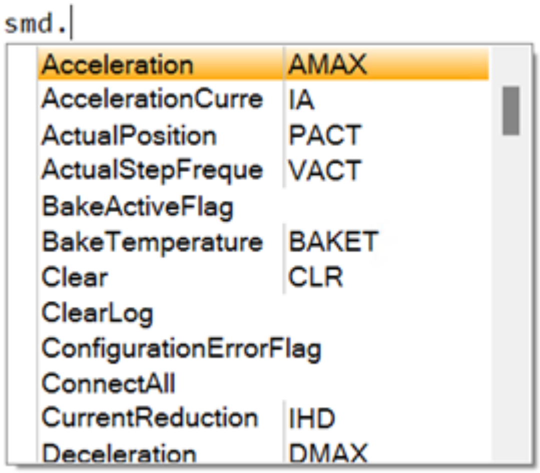
If the serial number of a connected SMD3 does not match any of those in the project file that is open: The SMD3 is considered a new device in the project, and the project file will be initialised from the configuration found in the SMD3 itself. After this point, the first behaviour outlined above applies.

INFORMATION: In the first case above, configuration items that require the SMD3 to be in standby will not be correctly synchronized if the SMD3 is not in standby when the software connects.

Scripting

The software includes an easy to use script editor, that allows for sequences to be programmed and executed on multiple connected SMD3 devices, as well as system level operations such as adding and removing SMD3 devices from the project.

The scripting language used is JavaScript; this is powerful, easy to use and extensively documented. A global ‘smd’ object is made available from which you perform all interactions with the SMD3s. Type ‘smd.’ and an auto completion popup appears, showing all available commands, as well as help documentation for each. Press the enter key to select an option, then provide any arguments required.



```
smd.
```

Acceleration	AMAX
AccelerationCurre	IA
ActualPosition	PACT
ActualStepFreque	VACT
BakeActiveFlag	
BakeTemperature	BAKET
Clear	CLR
ClearLog	
ConfigurationErrorFlag	
ConnectAll	
CurrentReduction	IHD
Deceleration	DMAX

A brief description of each function/command is presented below the scripting section. Here is an overview of the scripting area:

Ribbon

Buttons for device and scripting actions, and access to the "File" menu.

Scripting help panel

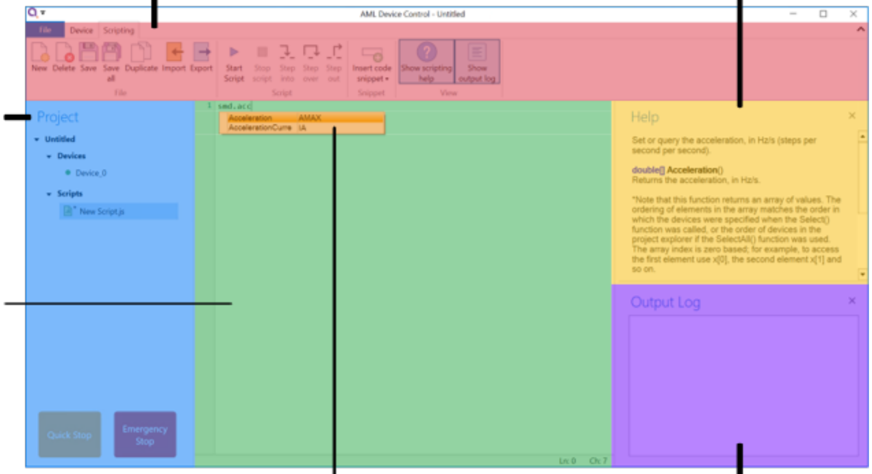
Help text relating to a selected function is displayed here.

Project panel

Add, remove and select devices or scripts. Re-arrange items by dragging.

Scripting work area

Scripts selected on the project panel are displayed here.



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

double[] Acceleration()
Returns the acceleration, in Hz/s.

*Note that this function returns an array of values. The ordering of elements in the array matches the order in which the devices were specified when the Select() function was called, or the order of devices in the project explorer if the SelectAll() function was used. The array index is zero based; for example, to access the first element use x[0], the second element x[1] and so on.

Auto completion pop-up

Beginning to type a command will bring up the auto completion pop-up.

Output log panel

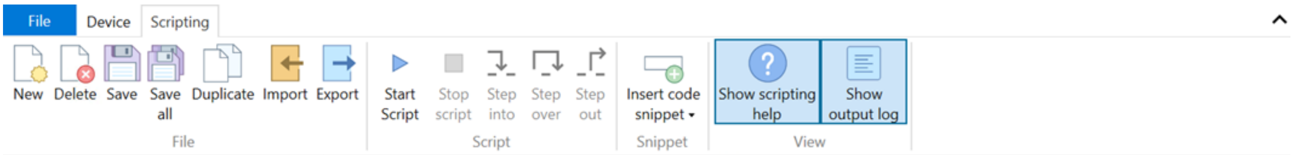
Output from the script when the *smd.Log()* function is used is displayed here.

The auto completion popup can be shown using the ‘Ctrl-K’ keyboard shortcut.

Information on the available SMD3 device specific commands can be found in section USB Interface of this manual. Serial command mnemonics will be auto completed by the script editor. The arguments of each scripting function are identical to those shown in section Command Reference, however, the format of querying and commanding is different. The example below shows how the SMD3 mode can be set and queried:

```
smd.Mode(2);                    // Set the SMD3 mode to 2 (remote)
smd.Mode();                     // Query state of mode
```

The ribbon contains scripting specific buttons.



Function specific to the SMD3 software

Function	Description
Add	bool **Add**(string **serial**) Add a new device to the project. Returns true if the device has been detected and added to the project. serial: Device serial number.

ClearLog	void **ClearLog**( ) Clear command line.
ConnectAll	void **ConnectAll**( ) Connect all devices in the project.
Delayms	void **Delayms**(int **value**) Delay in milliseconds. value: Minimum: 0 Maximum: 2^31 -1
DelaySeconds	void **DelaySeconds**(int **value**) Delay in seconds. value: Minimum: 0 Maximum: 2147483
DisconnectAll	void **DisconnectAll**( ) Disconnect all devices in the project.
Log	void **Log**(string **value**) Print value to the command line.
Name	bool **Name**(string **serial**, string **name**) Change name of the device. Returns true if the device has been found and name changed. serial: Device serial number. name: Device new name.
Remove	bool **Remove**(string **serial**) Remove a device from the project. Returns true if the device has been detected and removed from the project. serial: Device serial number.
RemoveAll	void **RemoveAll**( ) Remove all devices from the project.
Select	bool **Select**(string[ ] devices) Returns true if all the requested devices have been selected. devices: Name of the device(s).

SelectAll	void **SelectAll**( ) Select all devices.
SelectNone	void **SelectNone**( ) Deselect all devices.

Functions that query the SMD3 and that are not also available as a serial command are listed below. Note that these all return an array rather than a single value, with each array element corresponding to the data from one SMD3. The order of the array elements corresponds to the order in which the SMD3 devices are selected. For example, suppose the “X-axis”, “Y-axis” and “Z-axis” named devices were selected with the command “smd.Select(“Z-axis”, ”X-axis”, ”Y-axis)”, to check the standby flag of the “X-axis” device, use MotorStandbyFlag()[1].

Function	Description
BakeActiveFlag	bool[] **BakeActiveFlag**() Returns true if the bake mode is running.
ConfigurationErrorFlag	bool[] **ConfigurationErrorFlag**() Returns true if the motor configuration is corrupt.
EmergencyStopFlag	bool[] **EmergencyStopFlag**() Returns true if the motor is disabled by software.
ExternalEnableFlag	bool[] **ExternalEnableFlag**() Returns true if the external enable input is high.
ExternalInhibitFlag	bool[] **ExternalInhibitFlag**() Returns true if the external enable input is disabling the motor.
IdentModeActiveFlag	bool[] **IdentModeActiveFlag**() Returns true if the ident mode is active.
JoystickConnectedFlag	bool[] **JoystickConnectedFlag**() Returns true if the joystick is connected.
LimitNegativeFlag	bool[] **LimitNegativeFlag**() Returns true if the negative limit is active.
LimitPositiveFlag	bool[] **LimitPositiveFlag**() Returns true if the positive limit is active.
MotorOverTemperatureFlag	bool[] **MotorOverTemperatureFlag**() Returns true if the motor temperature is greater than 190 °C.
MotorShortFlag	bool[] **MotorShortFlag**() Returns true if a motor phase to phase or phase to ground short has been detected.

MotorStandbyFlag	bool[] **MotorStandbyFlag**() Returns true if the motor is stationary.
TargetVelocityReachedFlag	bool[] **TargetVelocityReachedFlag**() Returns true if the motor is at the target step frequency.
TemperatureSensorOpenFlag	bool[] **TemperatureSensorOpenFlag**() Returns true if the selected temperature sensor is open circuit.
TemperatureSensorShortedFlag	bool[] **TemperatureSensorShortedFlag**() Returns true if the selected temperature sensor is shorted (not applicable to thermocouple)

Example scripts

Add device, rename and set device properties

```
// Add device with serial number 20054-027
smd.Add("20054-027");

// Set name of device with serial number 20054-027 to MyDevice
smd.Name("20054-027","MyDevice");

// Select device with name MyDevice
smd.Select("MyDevice");

// Set the acceleration and deceleration rate in Hz/s
smd.Acceleration(100);
smd.Deceleration(100);

// Set the acceleration current
smd.AccelerationCurrent(1.044);

// Set the hold current
smd.HoldCurrent(0);

// Set the run current
smd.RunCurrent(0.5);

// Set the start frequency
smd.StartFrequency(10);

// Set the step frequency
```

```
smd.StepFrequency(1000);

// Set the frequency at which the drive transitions to full step
smd.MicrostepTransition(500);

// Set the micostep resolution
smd.Resolution(64);
```

Execute a series of movements, illustrating synchronous move commands

```
// Select device and set mode to remote
smd.Select("MyDevice");
smd.Mode(2);
// Move +100 steps, then -500 steps, and finally +100 steps
// "MoveRelative" executes synchronously, i.e. the motor must arrive at its destination position before the next line of code executes
smd.MoveRelative(500);
smd.MoveRelative(-500);
smd.MoveRelative(100);

// Log the finishing step position to the command line output area
smd.Log(smd.ActualPosition()[0]);
```

Execute movement on multiple drives, illustrating synchronous and asynchronous commands

```
// Select first device and start a movement
smd.Select("Device_0");
smd.Mode(2);
smd.MoveRelativeAsync(10000); // Execution continues to the next line of code immediately, regardless of how long this move will
                                    // requires to complete.

// Select second device and start a movement
smd.Select("Device_1");
smd.Mode(2);
smd.MoveRelativeAsync(5000);

// Both motors are completing their moves at the same time.
```

Get value of actual position counter and log to command line

```
// Select device with name MyDevice
smd.Select("MyDevice");

// Store actual position in a variable
pact = smd.ActualPosition();

// Log result to command line
smd.Log(pact[0]);
```

Check if the motor is in standby

```
// Select device with name MyDevice
smd.Select("MyDevice");

// Store status flags in a variable
status = smd.StatusFlags();

// Log a specific bit status of the selected device to the command line
if(status[0] = status[0] & (0x1 << 6)){
    smd.Log("Motor is in standby");
} else {
    smd.Log("Motor is running");
}
```