BUCHER automation

Application Note 057



60886447_01

MCX-Axes

This document has been compiled by Bucher Automation AG with due diligence based on the state of the art as known to them. Any revisions and technical advancements of our products are not automatically made available in a revised document.
The Bucher Automation AG shall not be liable for any errors either in form or content, or for any missing updates, as well as for any damage or detriment resulting from such failure.

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach/Neckar, Germany
T +49 7141 2550-0
info@bucherautomation.com

Technical support
T +49 7141 2550-444
support@bucherautomation.com

Sales
T +49 7141 2550-663
sales@bucherautomation.com

www.bucherautomation.com

Translation of the german original User Manual

Revision	1.01
Date of issue	6/22/2026

Table of Contents

1 Requirements 5

2 Types of Axes 6

2.1 Real axis 6

2.2 Simulation axis 6

2.3 Virtual axis 7

2.4 External axis 7

2.5 Shadow axes 8

3 Mechanical Design 9

3.1 Linear - normal 9

3.2 Linear – modulo, direction: bidirectional 9

3.3 Linear – modulo, direction: unidirectional positive 9

3.4 Linear – modulo, direction: unidirectional negative 9

3.5 Rotational - normal 9

3.6 Rotational - modulo, direction: bidirectional 10

3.7 Rotational—modulo, direction: unidirectional positive 10

3.8 Rotational - modulo, direction: unidirectional negative 10

4 Mechanical measurement units 11

5 Using the axis 12

5.1 Creating an MCX Project 12

5.2 Real axis 13

5.2.1 Adding a servo amplifier 13

5.2.2 Configuring a Real Axis 14

5.2.3 Advanced axis parameters 15

5.2.4 Mechanical parameters 15

5.2.5 Axle commissioning 16

5.2.6 Manual operation 16

5.3 Simulation axis 19

5.4 Virtual Axes 21

5.5 External axis 23

5.5.1 JC-440Ext with EtherCAT® Drives 23

5.5.2 JC-365MC 24

5.5.3 Parameterization 26

5.5.4 Setting the Encoder Type 27

5.6 Shadow axis 28

5.6.1 MotionSetup Status 29

6 Programming axes 30
6.1 Ramp parameter settings 30
6.1.1 Ramp type 30
6.1.2 Ramp break mode 31
6.2 Enabling/disabling axes 32
6.2.1 Power.Enable(), Power.Disable(); 32
6.2.2 Drive.IsReadyForOperation 32
6.2.3 Power.QuickStop() 32
6.3 Referencing 33
6.3.1 Set a standard 33
6.3.2 Reference run 33
6.4 Positioning 36
6.4.1 Ramp Status 36
6.4.2 Point-to-Point (MovePtp) 38
6.4.3 PositioningMode - Positioning Modes 40
6.4.4 Endless positioning (MoveVelocity) 42
6.4.5 Stop (MoveHalt) 44

1 Requirements

Usage

– JetSym 5.3.6
– The description assumes a mouse with a right-handed configuration
  – Open submenu: Right-click
  – Double-click: Double-click the left mouse button quickly
  – Selection: A single click of the left mouse button

Naming Axes and Groups

– Users are generally free to name the axes and groups as they wish
– This Application Note uses the following prefixes for naming purposes

Parameters	Prefix
Real axes	x
Virtual Axes	xv
Shadow axes (Follower axes)	xs
technology group	tec
path group	geo

Tab. 1: Prefixes Used

– Prefixes have no logical effect, but they quickly indicate to the reader of a program the type of the object being used.

Safety Instructions

⚠ WARNING



Possible injury or property damage resulting from failure to follow the safety instructions

During the use, commissioning, programming, or testing of axes, hazardous situations may arise, whether due to moving mechanical parts or electrical voltages on the motor and servo amplifiers.

► Read and follow the safety precautions and operating restrictions for the servo amplifiers and control components outlined in the Safety chapter.

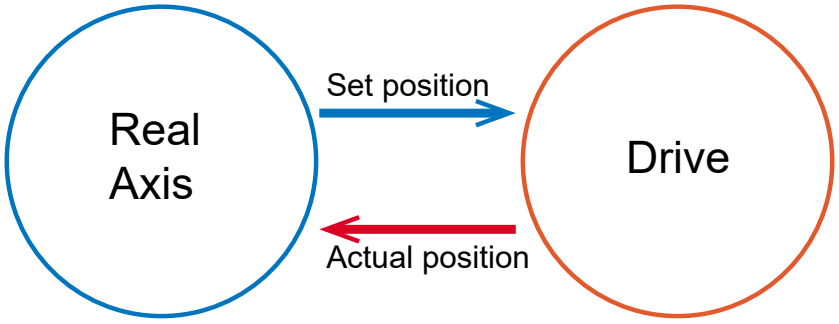
2 Types of Axes

Five types of axes can be defined in the MotionControl kernel:

- Real axis
- Simulation axis
- Virtual axis
- External axis
- Shadow axis

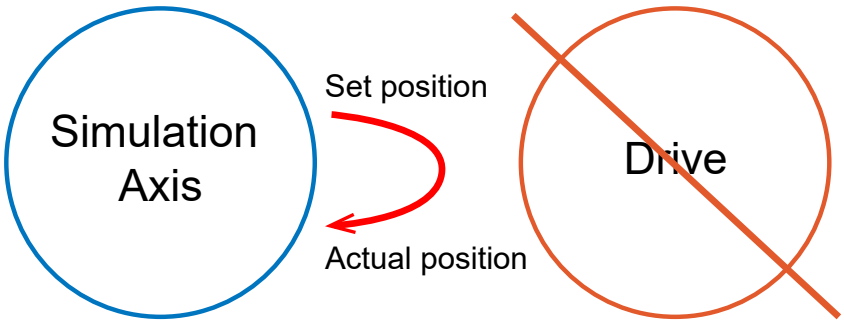
2.1 Real axis

- A real axis—as the name implies—actually exists.
- Servo amplifiers are connected via a system bus, through which setpoints and commands are transmitted.
- At the same time, actual values are sent from the servo amplifier to MotionControl.
- The setpoints are calculated in every cycle of the MotionControl kernel.



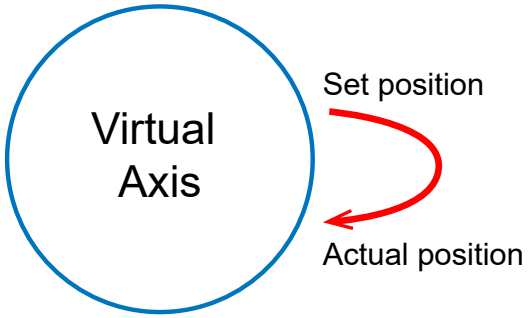
2.2 Simulation axis

- A simulation axis is a virtual representation of a real axis.
- Since the hardware required for application development is not necessarily connected to the controller, real axes can be reconfigured as simulation axes.
- This allows the processes to be put into operation even without a servo amplifier.



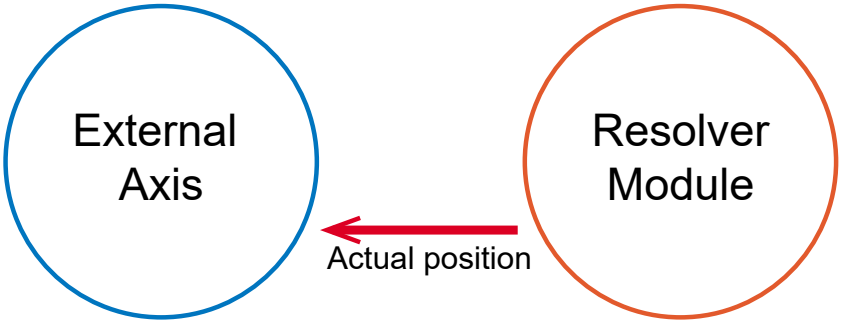
2.3 Virtual axis

– A virtual axis is used only on the MotionControl kernel and is not connected to a servo amplifier.
– It often serves as a central axis within a technology group.



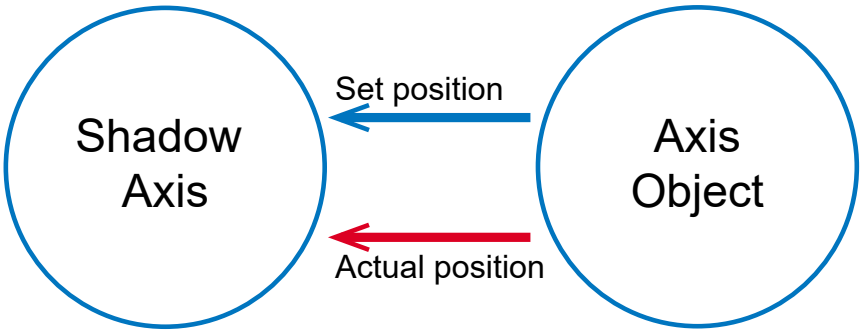
2.4 External axis

– For applications in which, for example, the position of a grinding wheel is required, an external axis is used.
– This usually uses an encoder module or an additional encoder input on a servo amplifier.
– In this process, only the actual values sent by the module are evaluated.
– No setpoints are generated for this axis.
– The operating status for this axis is always “inactive”!

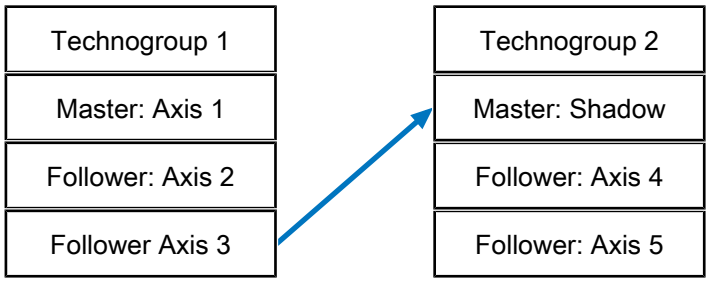


2.5 Shadow axes

– A trailing axis is a combination of an external axis and an internal axis.
– The follow-along axis is linked to an axis object of a different type.
– Consequently, this axis inherits the setpoint and actual values of the linked axis.



– A driven axis is always a master axis in a technology group.
– This axis is used when the source axis cannot be the lead axis of a technology group, for example, because this axis itself is a shadow axis in another group or because it is to be used independently of a technology group.



– In this example, Axis 3 is a slave axis in a technology group.
– However, since axes 4 and 5 are supposed to follow axis 3, a slave axis (axis 6) is used, which represents axis 3.
– Since an axis can only be active in one group at a time (it can be a member of multiple groups, but can only be active in one group at a time), Axis 3 can be operated independently of Group 2.

3 Mechanical Design

When using and programming the axes, the focus is primarily on the operating point. Therefore, the Motion-Control kernel must know how the mechanical system is configured.

3.1 Linear - normal



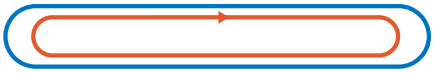
A linear normal mechanism involves rectilinear motion and has a beginning and an end, as is the case, for example, with linear axes.

3.2 Linear – modulo, direction: bidirectional



A linear mechanism moves in a straight line and can be moved indefinitely, as in the case of a conveyor belt, for example.

3.3 Linear – modulo, direction: unidirectional positive



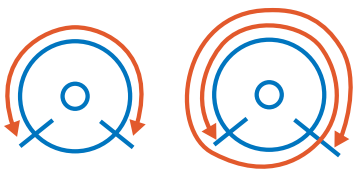
The linear mechanism may only be moved in the positive direction.

3.4 Linear – modulo, direction: unidirectional negative



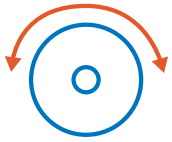
The linear mechanism may only be moved in the negative direction.

3.5 Rotational - normal



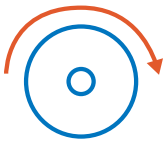
A mechanism that rotates in a circle with a limited range of motion. The travel range is not limited to the module area.

3.6 Rotational - modulo, direction: bidirectional



A mechanism that rotates endlessly in a circle and can be moved in either direction.

3.7 Rotational—modulo, direction: unidirectional positive



A mechanism that rotates endlessly in a circle and may only be moved in the forward direction.

3.8 Rotational - modulo, direction: unidirectional negative



A mechanism that rotates endlessly in a circle and may only be moved in the negative direction.

4 Mechanical measurement units

You can specify the unit of measurement used for each axis separately.

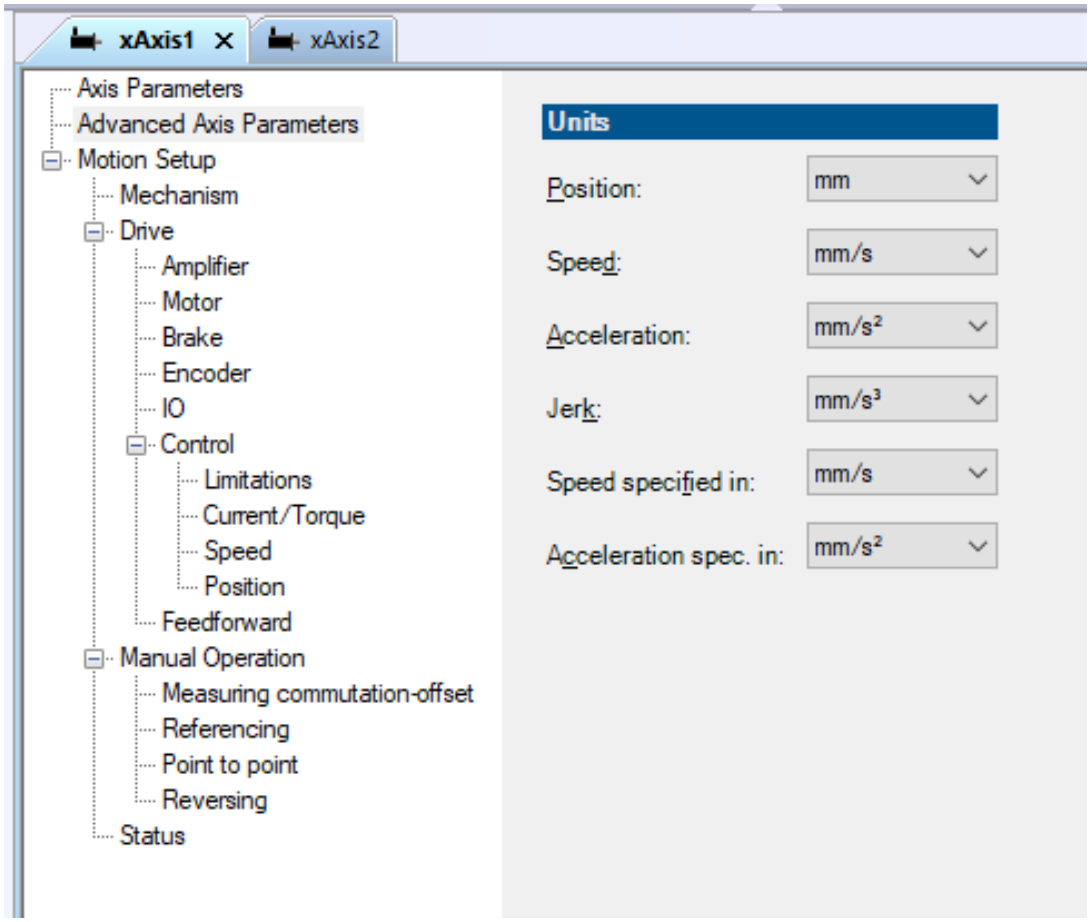


Fig. 1: Mech. Setting measurement units

These units form the basis for how MotionControl interprets the corresponding values.

Example

Set unit	Value	Speed
mm/s	100	100 mm/s
m/s	100	100,000 mm/s

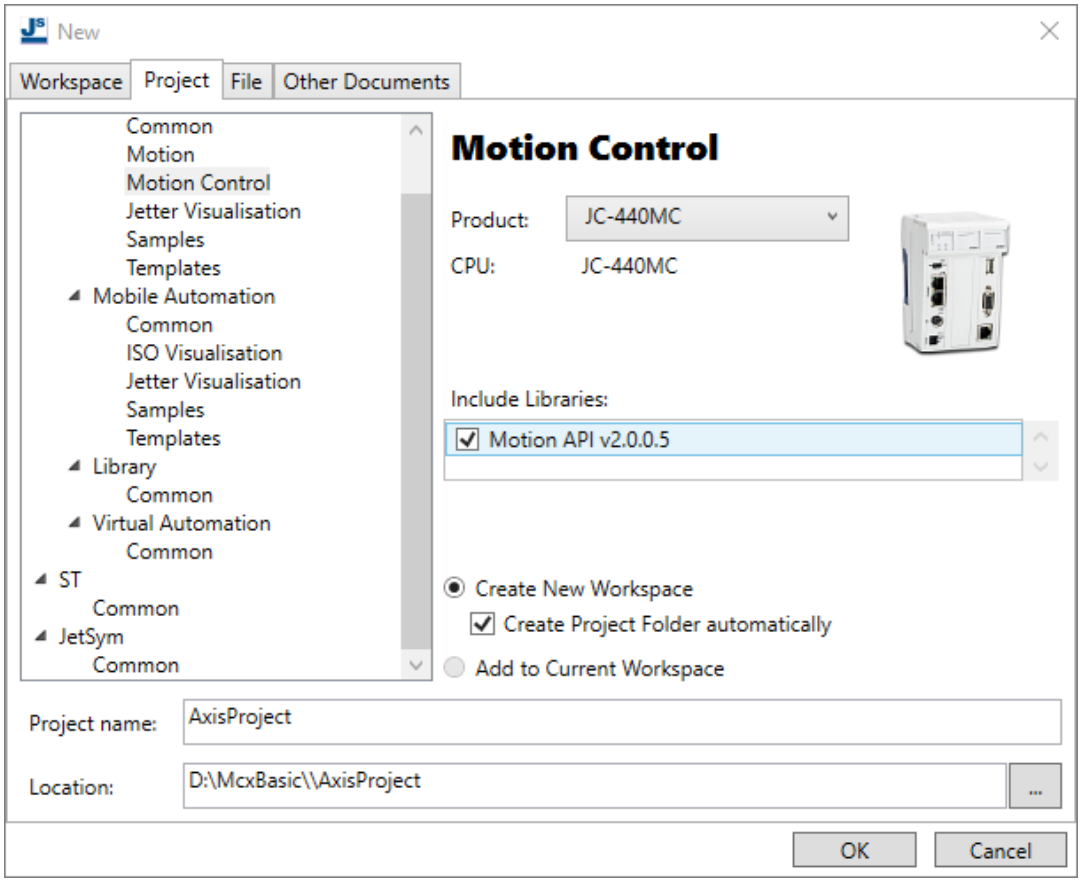
It is recommended that you continue to use the standard SI units.

5 Using the axis

5.1 Creating an MCX Project

To illustrate the different axis types, we will create a project to which we will gradually add and configure the various axes.

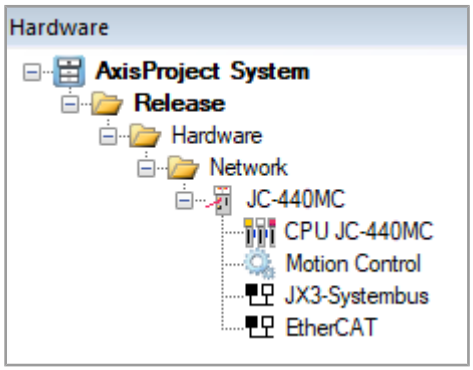
1. In JetSym, open the Project Wizard by selecting File -> New .



2. Select an MCX controller and integrate the MotionAPI.
   The system will always suggest the version that is currently installed and compatible with the controller.
3. Click OK to confirm.

   ⇨ The project tree is generated.

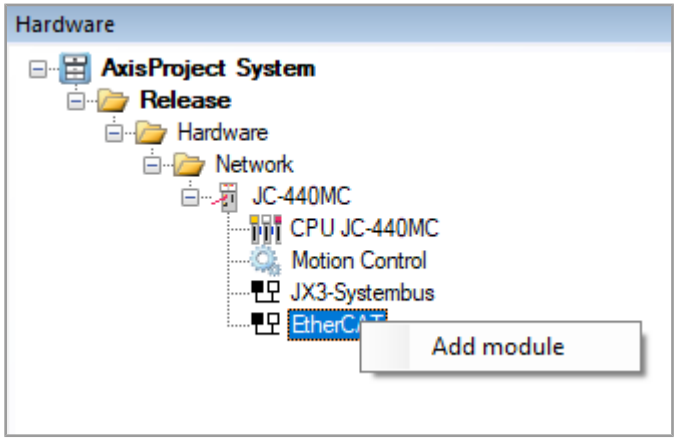
⇨ The controller and base nodes have now been created in the Hardware Manager for further configuration.



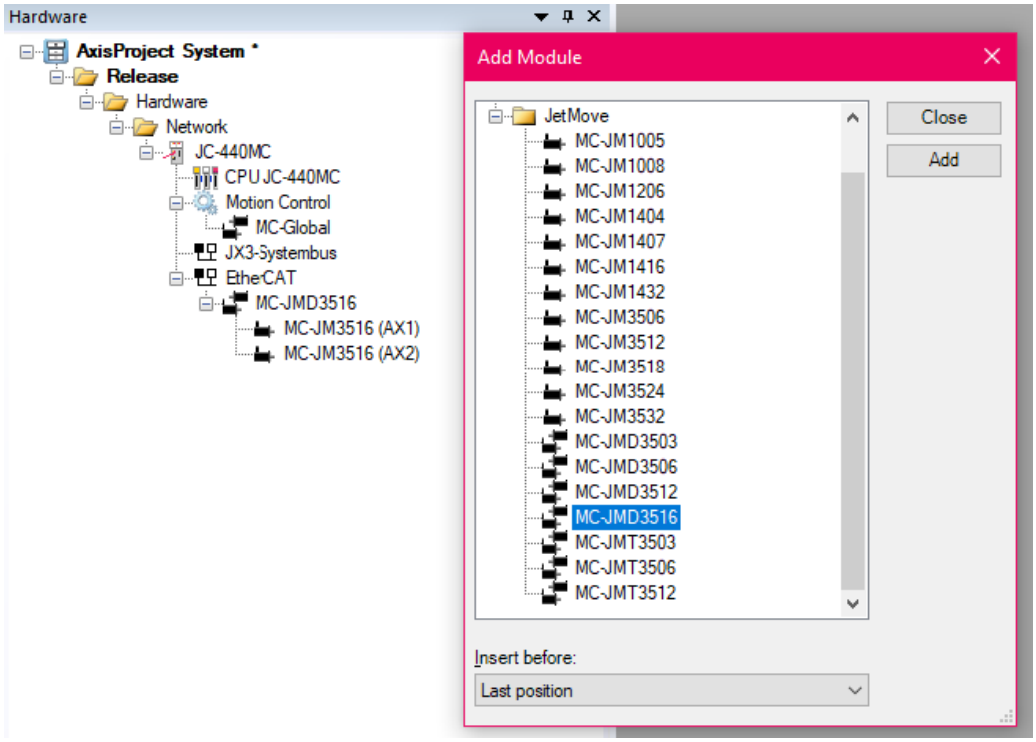
5.2 Real axis

5.2.1 Adding a servo amplifier

Servo amplifiers can now be connected to the EtherCAT here.



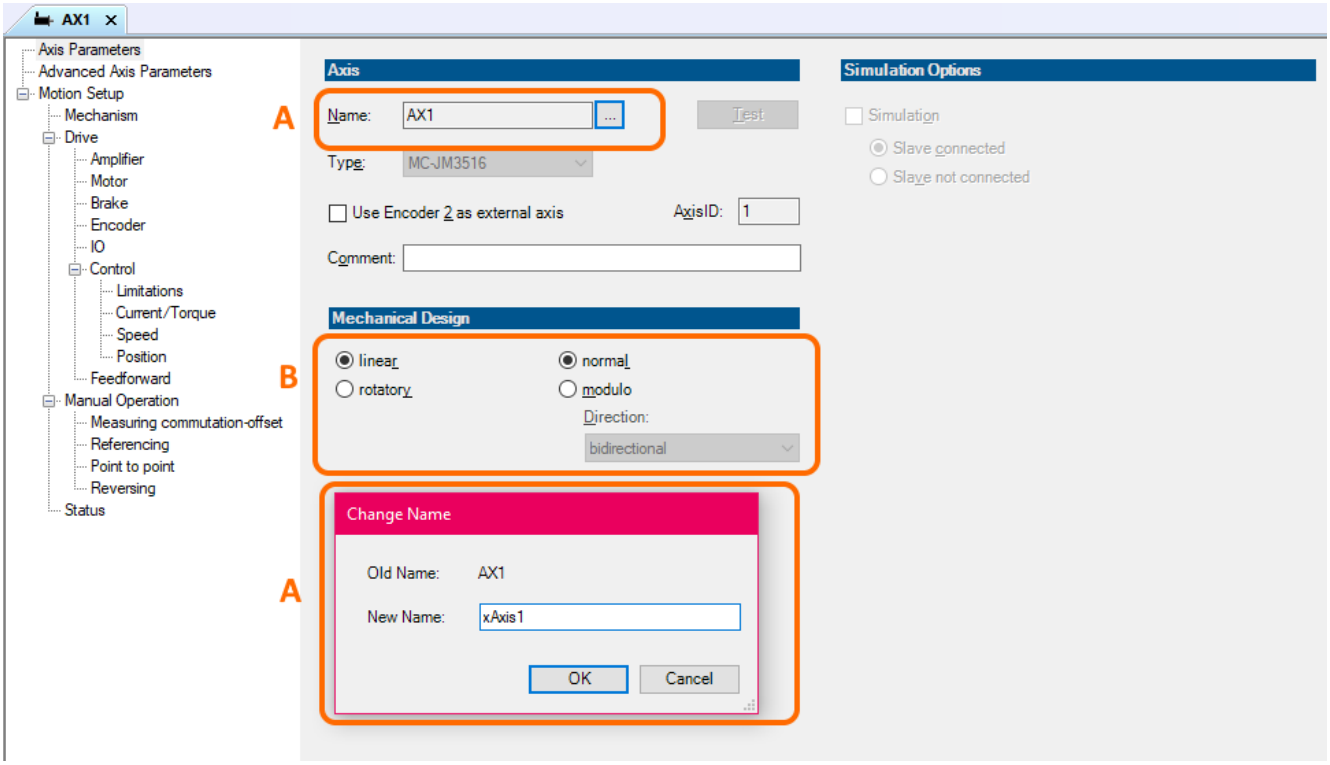
1. Use the submenu under the "EtherCAT" node to open the dialog box for selecting modules.
2. Use Add module to connect these to the node.



3. Now select the required servo amplifiers and add them to the hardware configuration by double-clicking or clicking Add.

5.2.2 Configuring a Real Axis

Double-clicking an axis object opens MotionSetup—a configuration and commissioning dialog.



In this example, the MotionSetup for axis "AX1" is open.

On the MotionSetup home page, you can configure the basic settings for the axis object.

A

It is recommended that you assign a name that is appropriate for the application instead of the suggested axis name “AX1.” This name is also used to refer to the respective axis object in the STX program. You can customize the axis name in this dialog box.

B

In the "Mechanical Design" dialog box, you specify whether the motion is rotary or linear and whether the axis is a modulo axis or a normal axis.

Standard axis

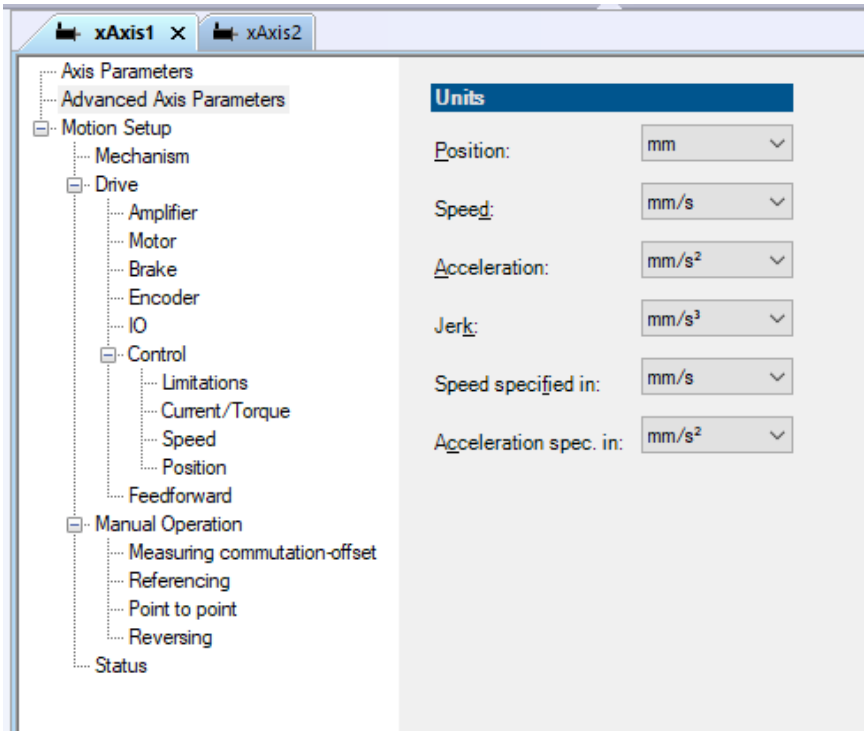
The axis has a fixed travel range that cannot be exceeded.

Modulo axis

When the module range is exceeded, the axis position resets to the beginning. In this context, the motion at the operating point is usually taken into account.

5.2.3 Advanced axis parameters

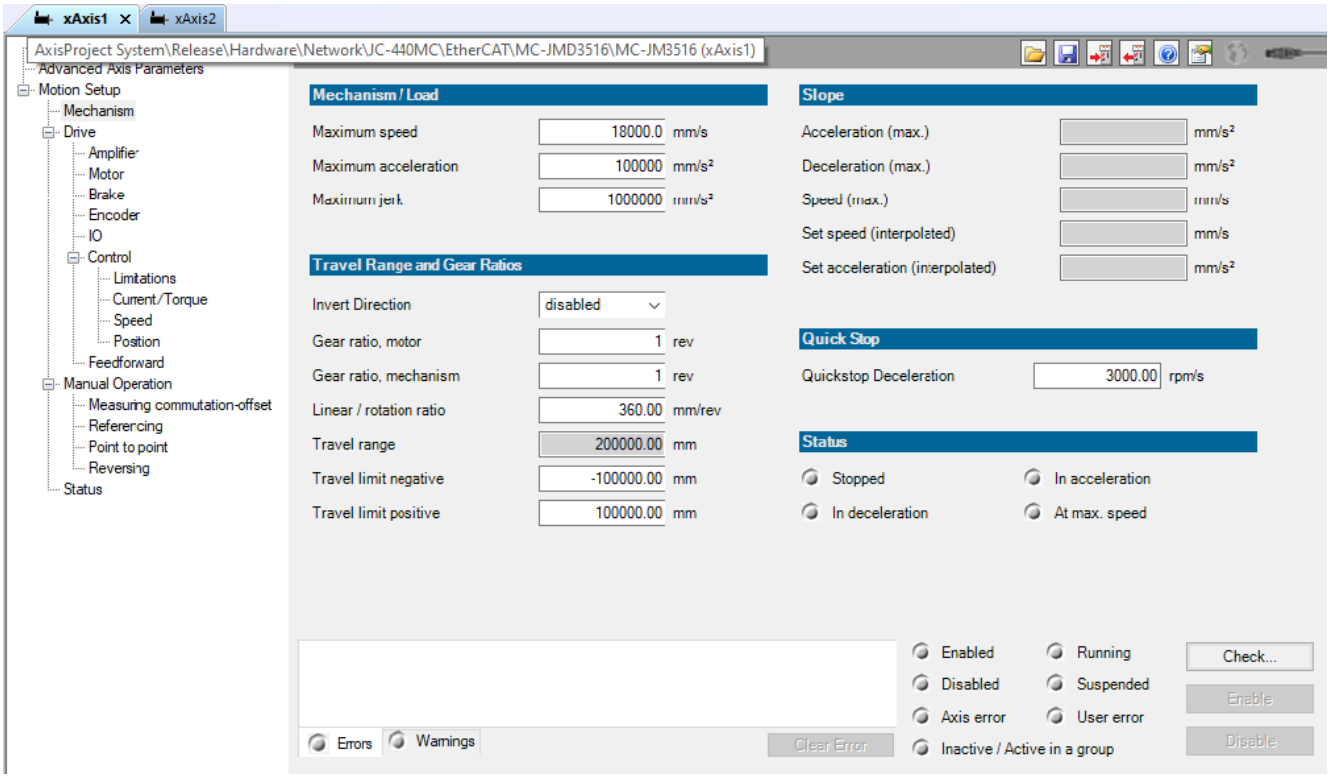
You can set the axis units here.



See Mechanical measurement units [▶ 11].

5.2.4 Mechanical parameters

On the MotionSetup page Mechanism, parameters important for the setpoint calculation are preset.



These parameters describe the mechanical conditions:

– How fast can or should the mechanism be moved?

– What accelerations are permitted?

– Does the direction of rotation need to be changed because, when the motor moves in the positive direction, the mechanism moves in the logically negative direction?
– What are the gear ratios?
– What is the permissible operating range?

5.2.5 Axle commissioning

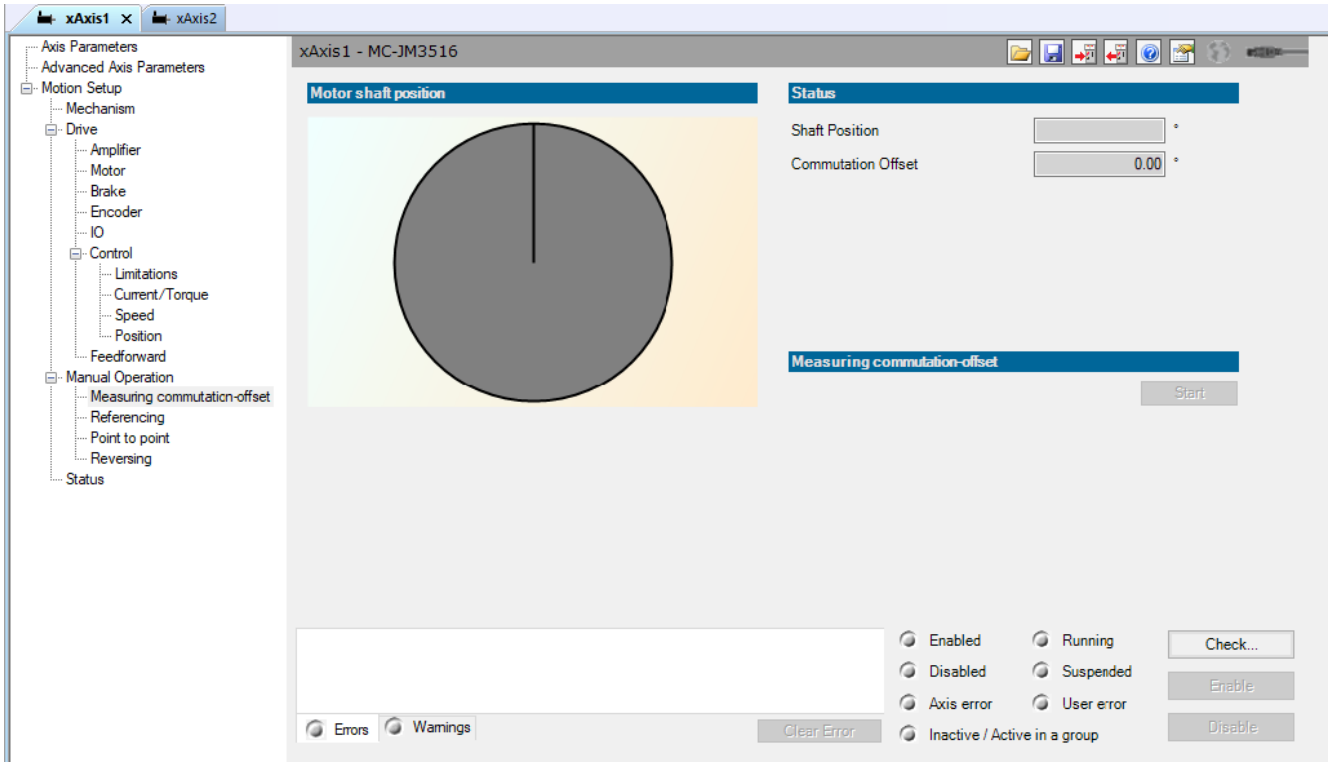
Additional MotionSetup pages are available for commissioning the axis.

► Please refer to the servo amplifier manuals for the commissioning procedure.

5.2.6 Manual operation

Measuring the commutation offset

A commutation measurement can be performed here to determine the commutation angle.

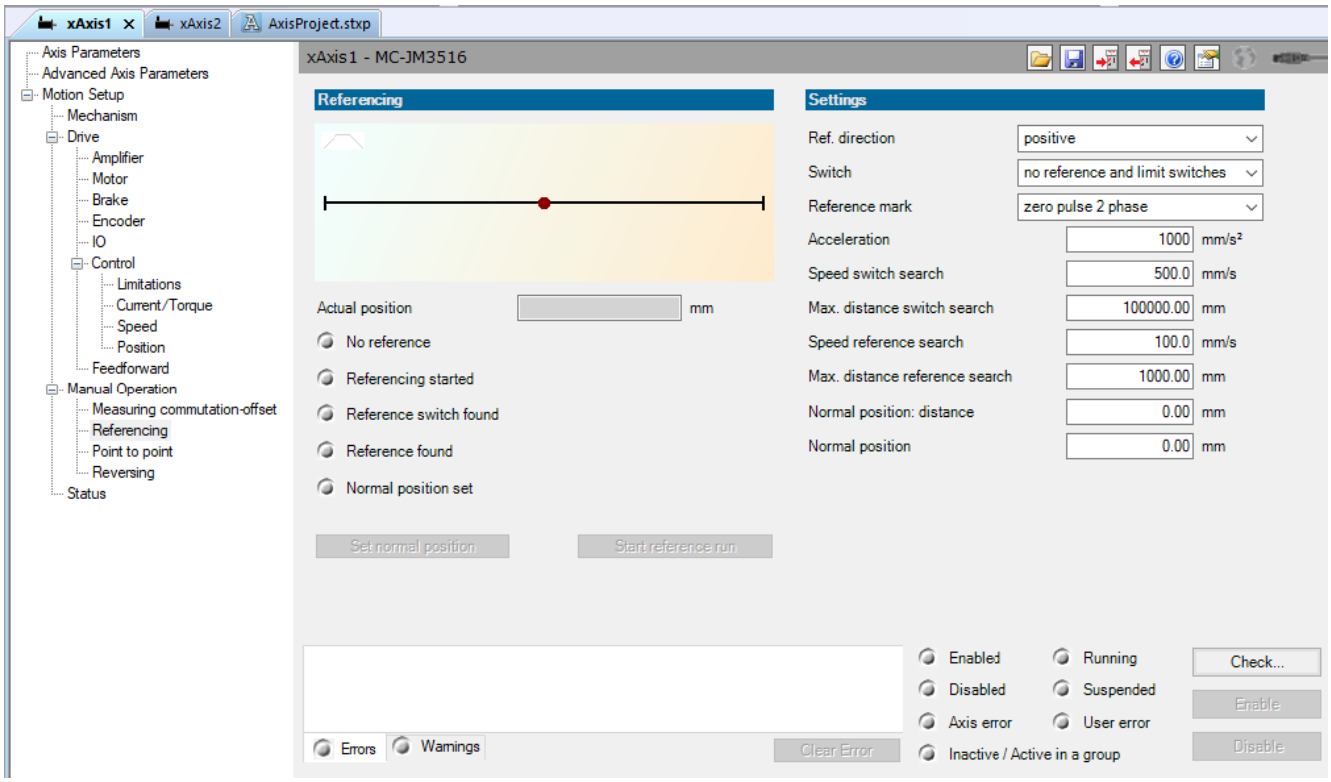


– By default, servomotors from Bucher Automation have a commutation offset of 0°. However, this angle may vary for engines from other manufacturers. This measurement can be used to determine the commutation.
– The calculated commutation is saved in the axis' ini file and loaded when the controller is restarted.
– A commutation measurement can also be started from within the application:

```
xAxis1.Drive.Motor.MeasureCommutationOffset(DriveCommOffsetMeasurement-
Modes.DynamicMode);
when xAxis1.State.IsDisabled continue;
```

Referencing

On this page, you can configure and test the reference run.



– The set reference run parameters are saved in the axis' ini file and loaded when the controller is restarted.
– If the referencing run is started in the application without any transfer parameters, the parameters set in the ini file are used, provided that the reference parameters have not been overwritten by the application.
– **Note:**
  If you start the referencing run without any transfer parameters, the most recently set parameters will be used. If the reference run parameters have been changed in the application, these values are used.

```
xAxis1.MoveHome.Start();
```

– However, the reference motion parameters can also be set separately within the application.

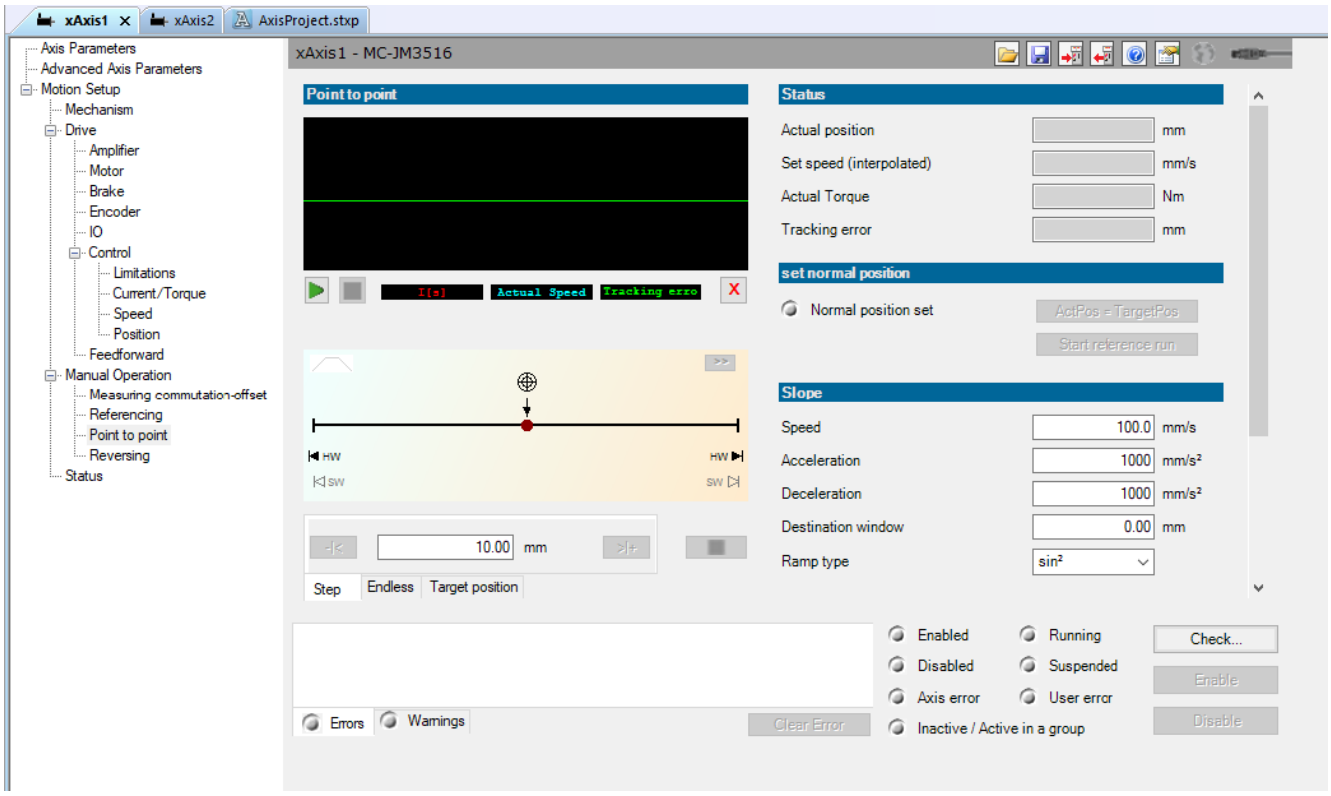
ⓘ Info

Additional information

For a detailed description of the referencing parameters and referencing options, please refer to the servo amplifier manuals and the JetSymHelp.

Point-to-Point

On this page, you can configure and test point-to-point positioning.



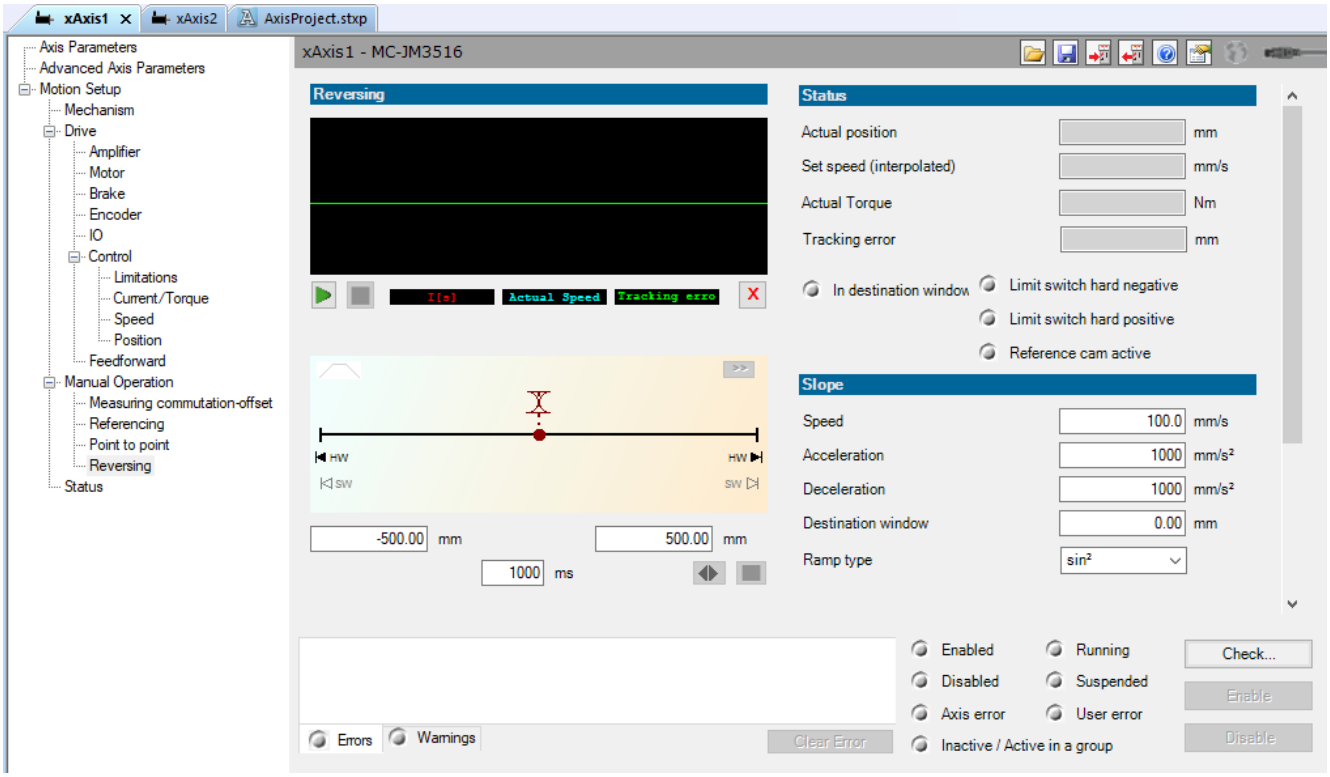
– The configured point-to-point parameters are saved in the axis' ini file and loaded when the controller is restarted.
– If point-to-point positioning is started in the application without any transfer parameters, the parameters set in the ini file are used, provided that the point-to-point parameters have not been overwritten by the application.
– **Note:**
  If you start point-to-point positioning without specifying any transfer parameters, the most recently set parameters will be used. If point-to-point parameters have been changed in the application, those values are used.

```
xAxis1.MovePtp.Start();
```

– However, the point-to-point parameters can also be set separately within the application.

Reverse

On this page, you can set the unit to reverse operation. This is often used to calibrate the control parameters.



– The set parameters—such as speed, acceleration, deceleration, target window, and ramp type—are identical to the point-to-point parameters. These are saved in the axis' ini file and loaded when the controller is restarted.

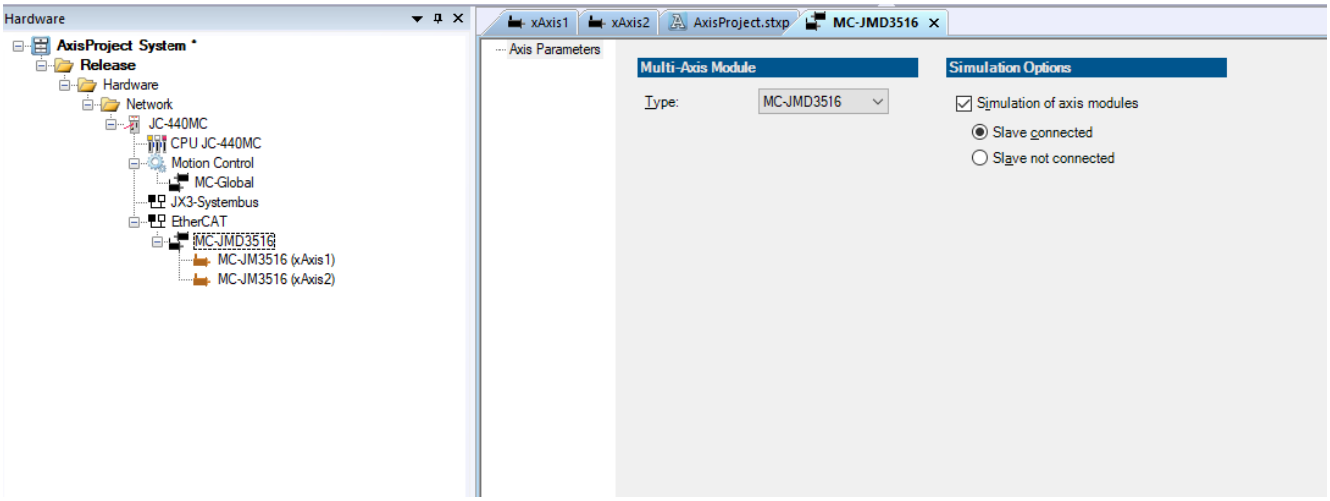
– The application does not include a separate reverse operation, but it can be programmed accordingly and, in simplified terms, looks something like this:

```
loop
xAxis1.MovePtp.Start(,Target1);
when xAxis1.State.IsEnabled continue;
xAxis1.MovePtp.Start(,Target3);
when xAxis1.State.IsEnabled continue;
end_loop;
```

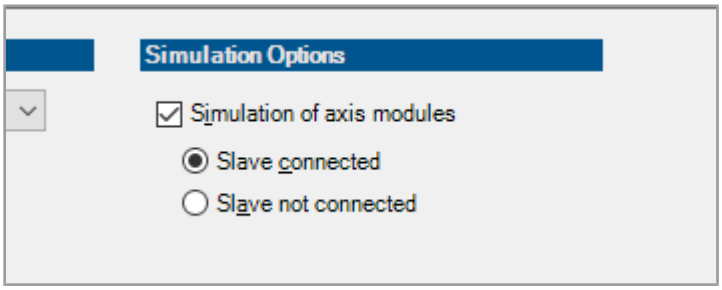
5.3 Simulation axis

The use of simulation axes allows the application program to be developed with no hardware connected, or with only some hardware connected.

If the machine is not available or is only partially available, the missing drives can be simulated. It is possible to safely test processes that are under development or need to be reviewed. To do this, select the Simulation option in the axis parameters of the respective servo amplifier.

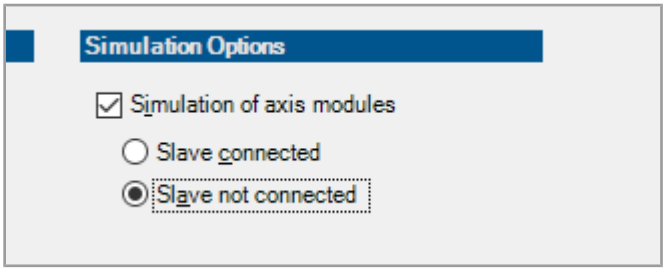


– **Note:**
  Onlyentire modules can be simulated at a time. If multiple-axis amplifiers are used, the simulation option is set in the superimposed parent node and applies to all axes of this module.
– If the servo amplifier is connected, select the Slave Connected option.



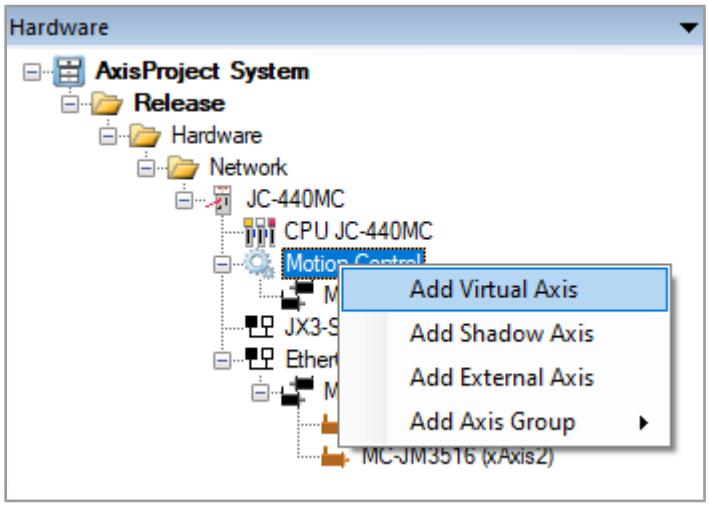
This option is useful when the hardware is connected but, for example, no motors are attached or the mechanical components are not to be moved during the test.

If no hardware is connected, select the Slave not connected option.

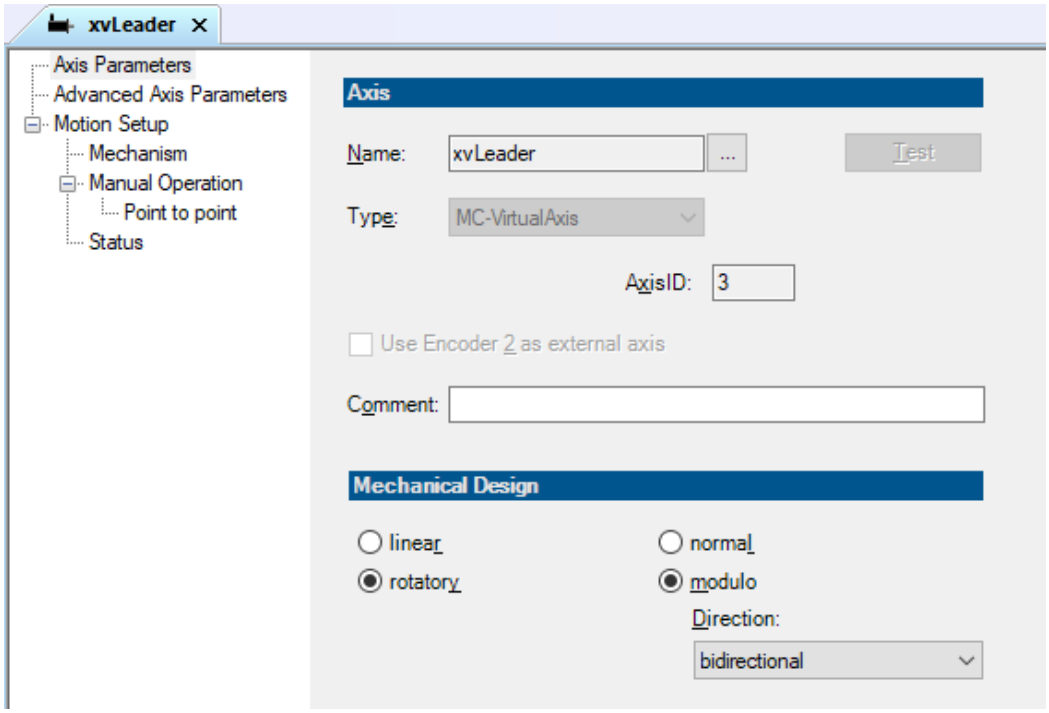


– This distinction is important because, in order to initialize the system bus, it must be clear which modules are connected.
– **Usage:**
  Simulation axes are used in the application just like real axes. However, it should be noted that the axis cannot be accessed directly. For example, reading the DC link voltage would not yield the expected result.
– External reactions may also be absent. If, for example, a light barrier is triggered on the actual machine while it is in motion, this will not occur in the simulation.

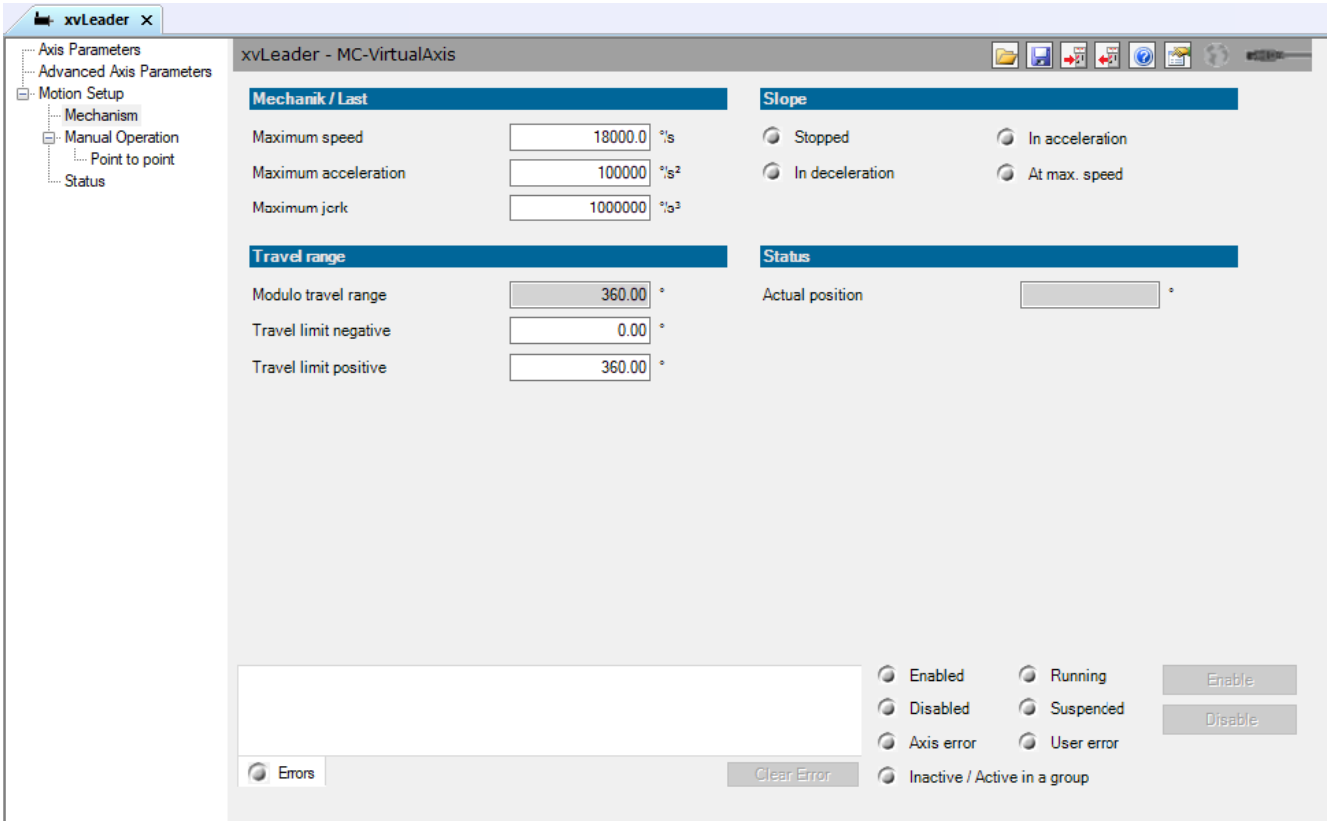
5.4 Virtual Axes



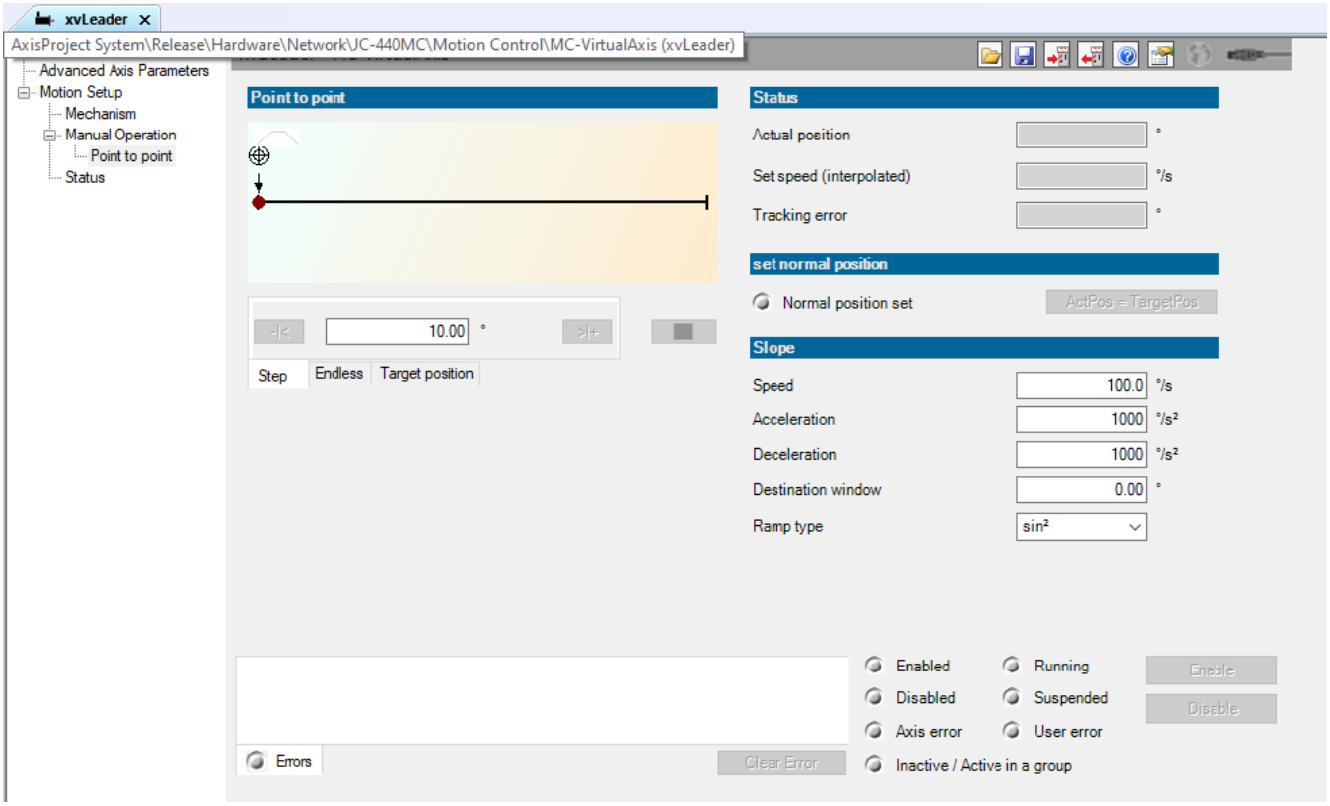
1. Use the submenu on the "Motion Control" node to open the dialog box for selecting axes.
2. Use the Add Virtual Axis option to add it to the hardware configuration.
   A virtual axis is configured in the same way as a real axis, except that this axis has no hardware and therefore no motor parameters need to be set.



3. During configuration, select the required mechanical design and set the travel range and maximum values.



For testing purposes, you can also use MotionSetup to position a virtual axis.



– The configured point-to-point parameters are saved in the axis' ini file and loaded when the controller is restarted.
– If point-to-point positioning is started in the application without any transfer parameters, the parameters set in the ini file are used, provided that the point-to-point parameters have not been overwritten by the application.

– **Note:**
When starting point-to-point positioning without transfer parameters, the most recently set parameters are used. If point-to-point parameters have been changed in the application, those values are used.

```
xvLeader.MovePtp.Start();
```

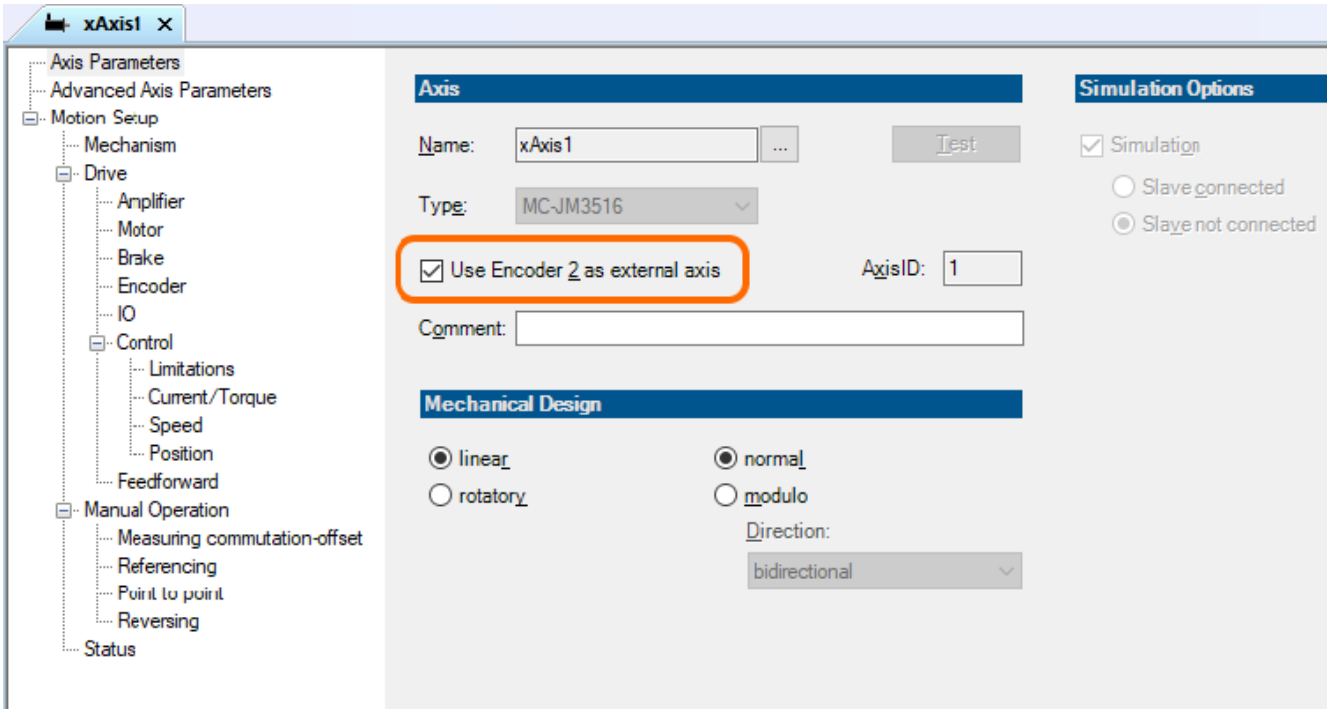
– However, the point-to-point parameters can also be set separately within the application.

5.5 External axis

1. Use the submenu on the "Motion Control" node to open the dialog box for selecting axes.
2. Use Add External Axis to add it to the hardware configuration.

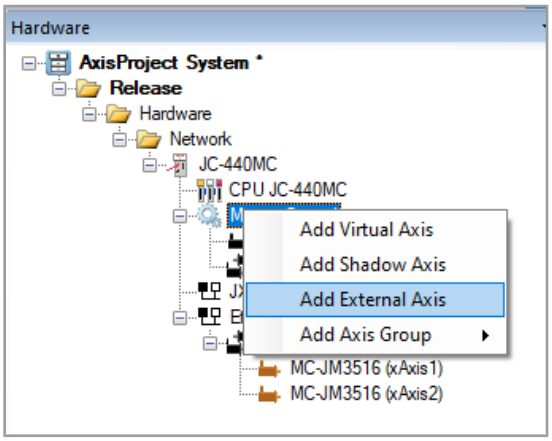
5.5.1 JC-440Ext with EtherCAT® Drives

An EtherCAT® drive with an additional encoder input is required.

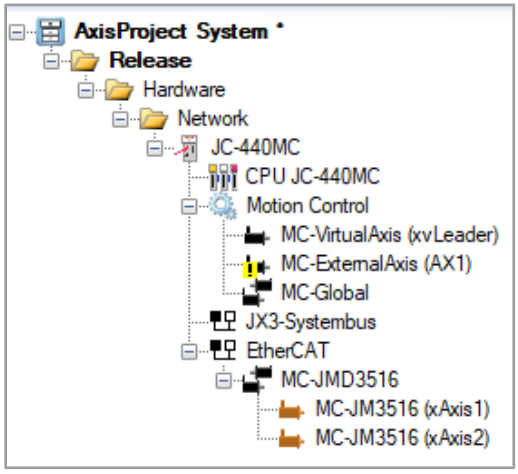


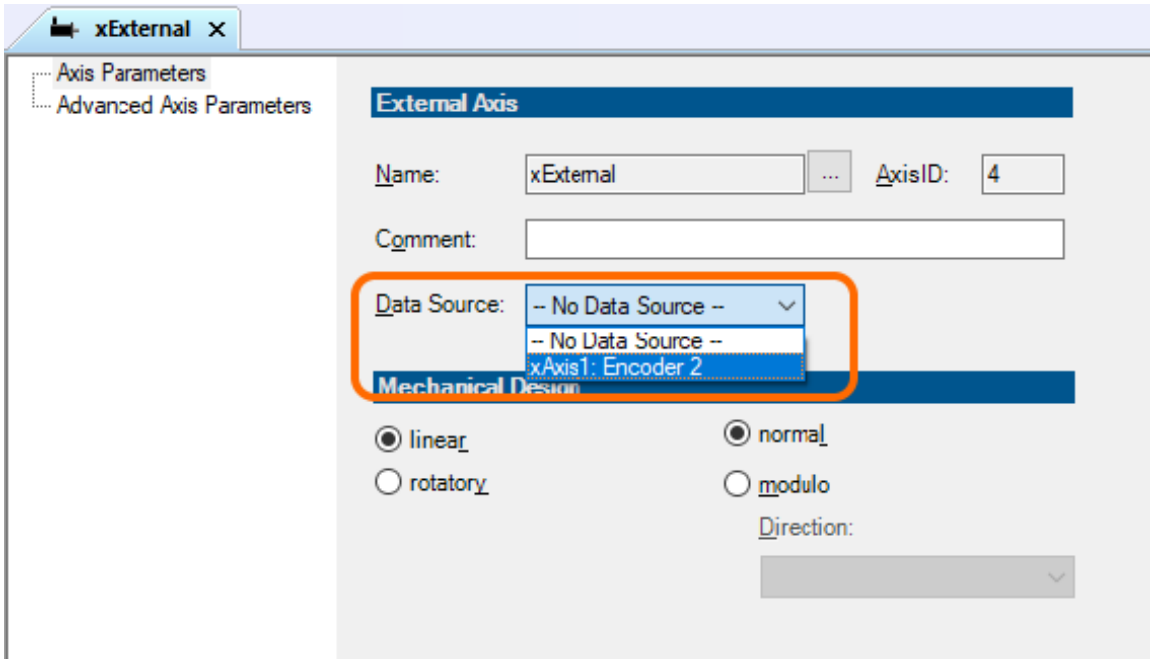
1. Select the option to use the second encoder input for an external axis.

   ⇨ This means that this input is not available as a second sensor for controlling the actual axis.



2. From the context menu of the "MotionControl" node, select Add External Axis .





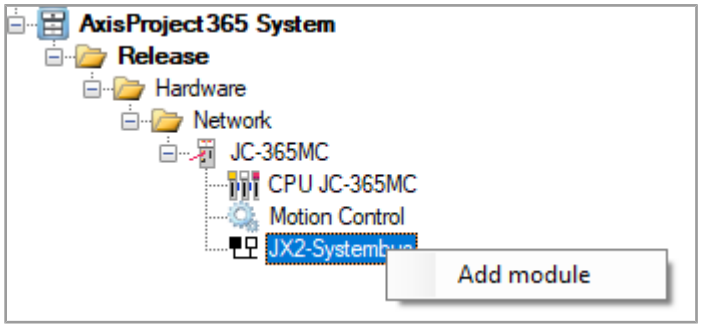
3. Select the encoder enabled for the external axis here and assign it to this axis object.
4. Set the Mechanical Design here as well.

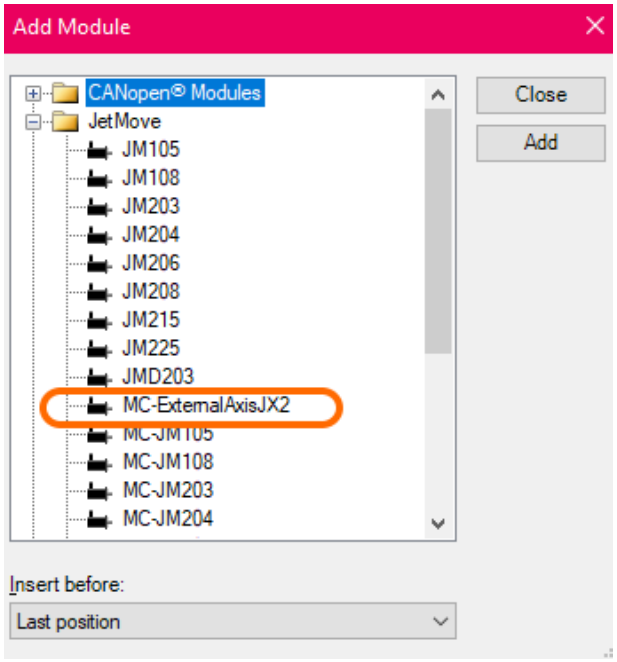
⇨ This allows you to specify whether you have a linear or rotational axis and whether it is normal or modulo.

See the Mechanical Design [▸ 9] chapter for more information.

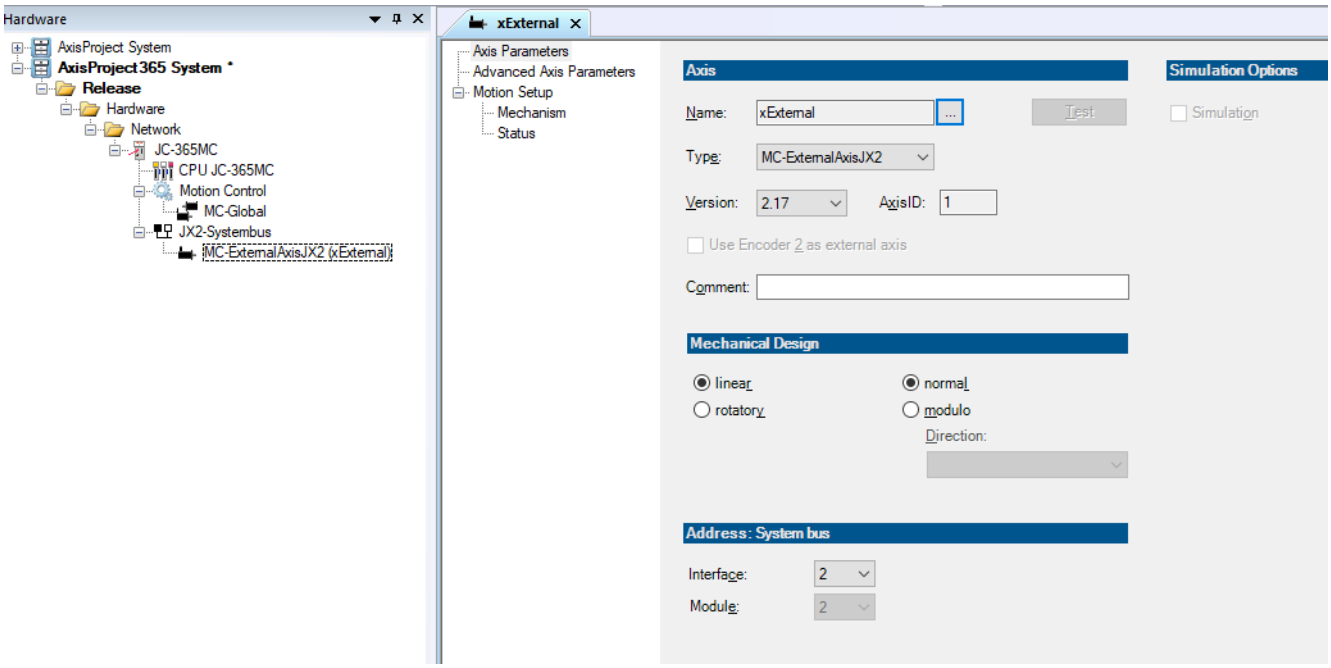
5.5.2 JC-365MC

– On the JM-200 and JM-105/108, the second encoder input—if present—cannot be used as an external axis.
– To acquire an additional encoder signal for an external axis, a JM-200 or JM-105/JM-108 is required on the JX2 system bus.





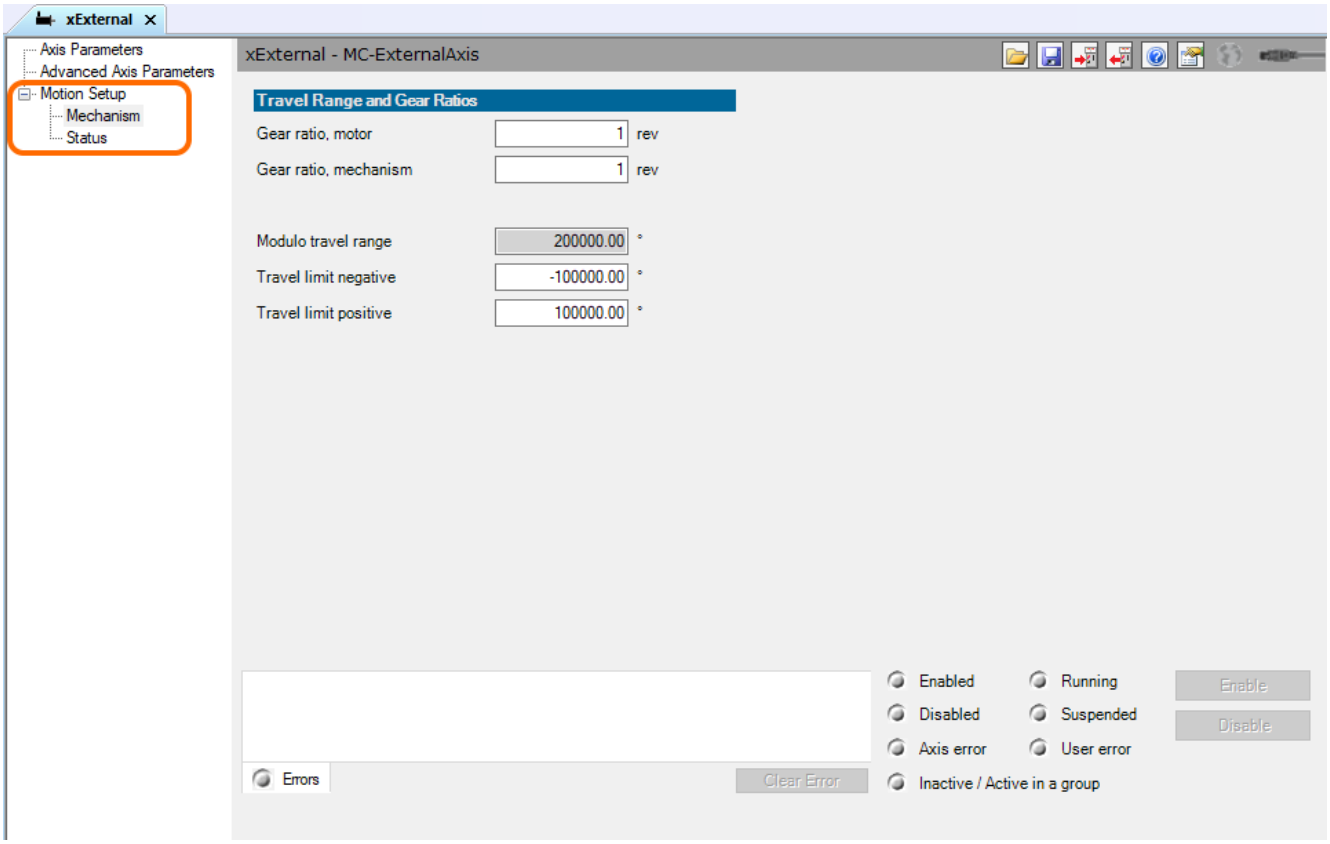
1. To do this, add the "MC-ExternalAxisJX2" module to the system bus.



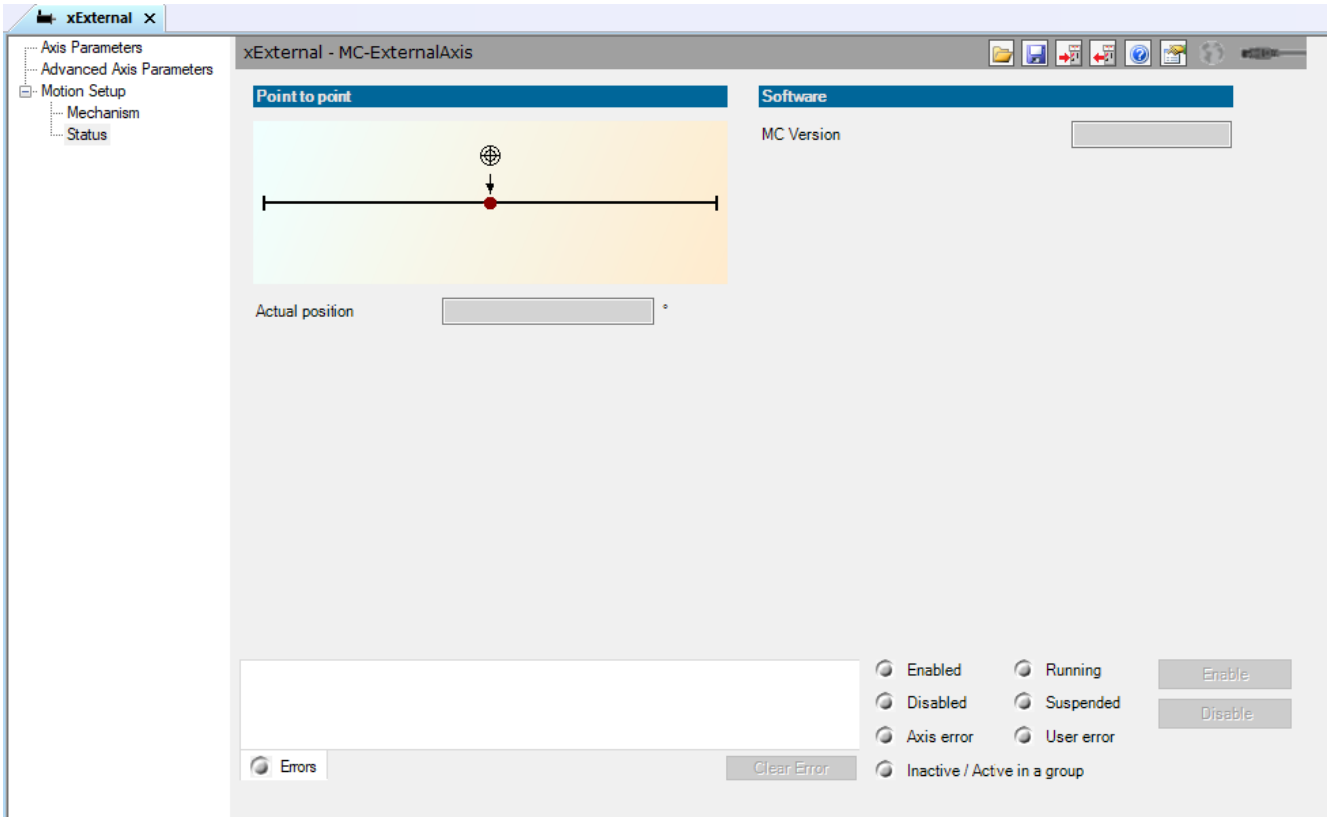
2. You can now set the same parameters as when using the JC-440Ext and JM-3000.

5.5.3 Parameterization

✓ If the basic settings are valid, the MotionSetupoption appears in the menu.



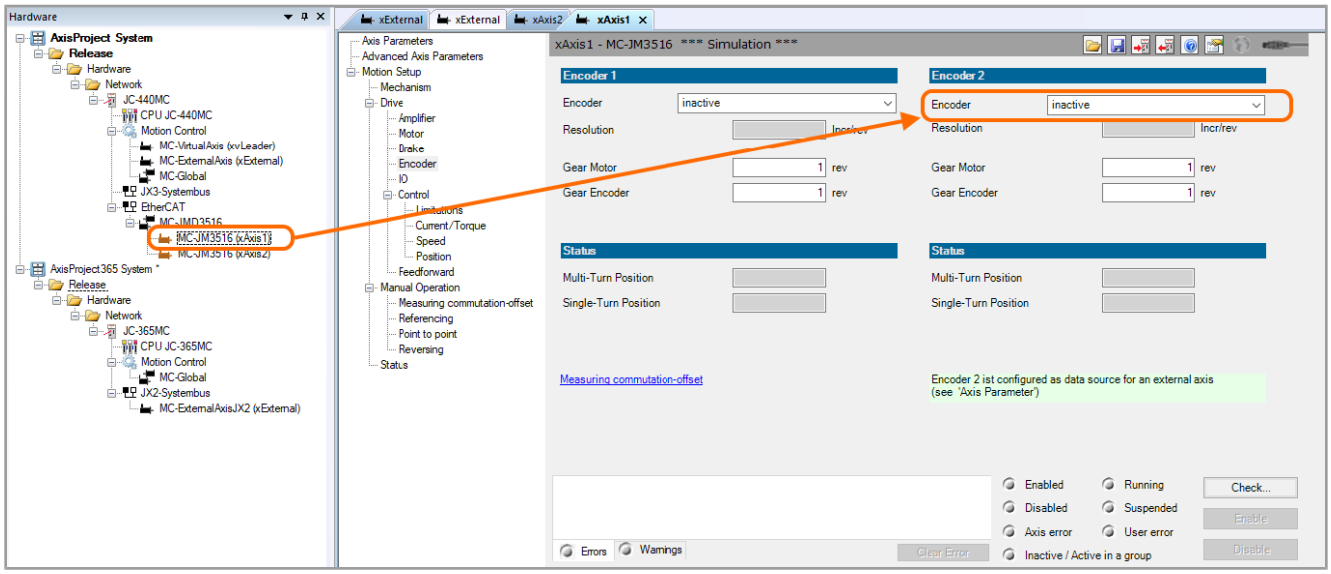
1. On the mechanism page, set the gear ratio and the travel range.



2. On the Status page, check the operation of the external encoder by viewing the current position.

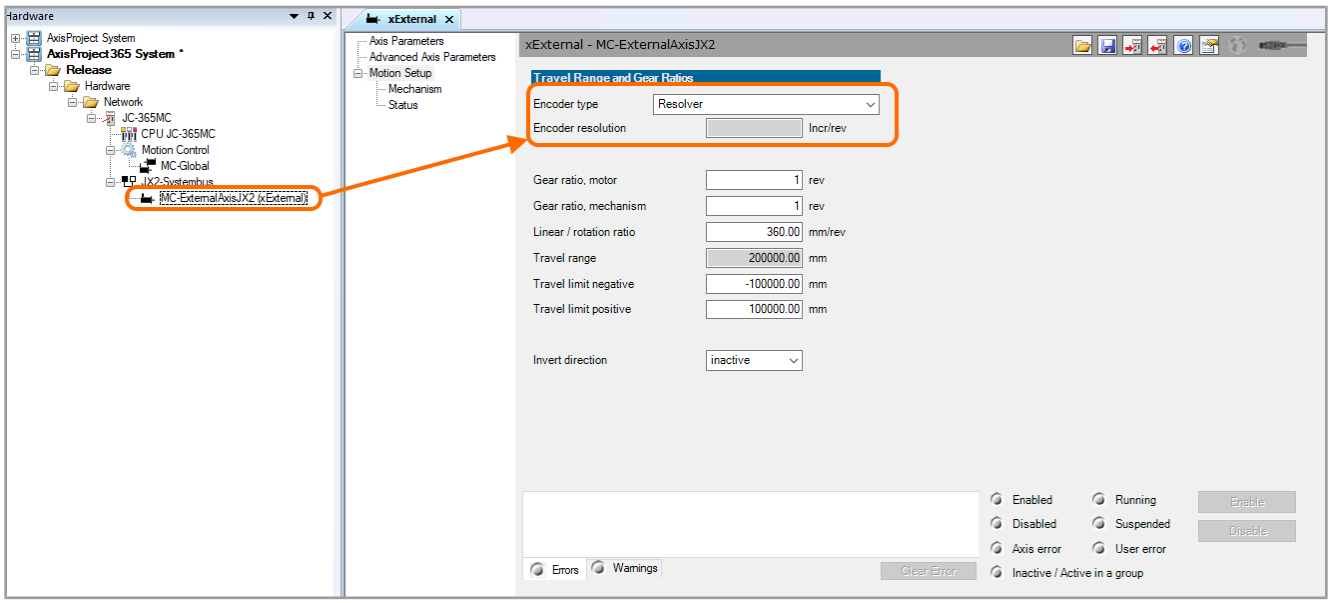
5.5.4 Setting the Encoder Type

JM-10xx, JM-2xxx, JM-3xxx



► Set the encoder type on the encoder page of the physical axis whose second encoder is used for the external axis.

JM-2xx/JM-1xx

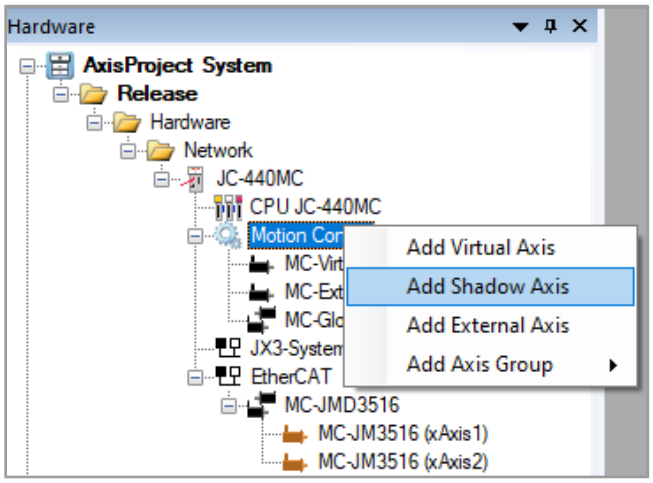


► Set the encoder type on the mechanical side of the external axis.

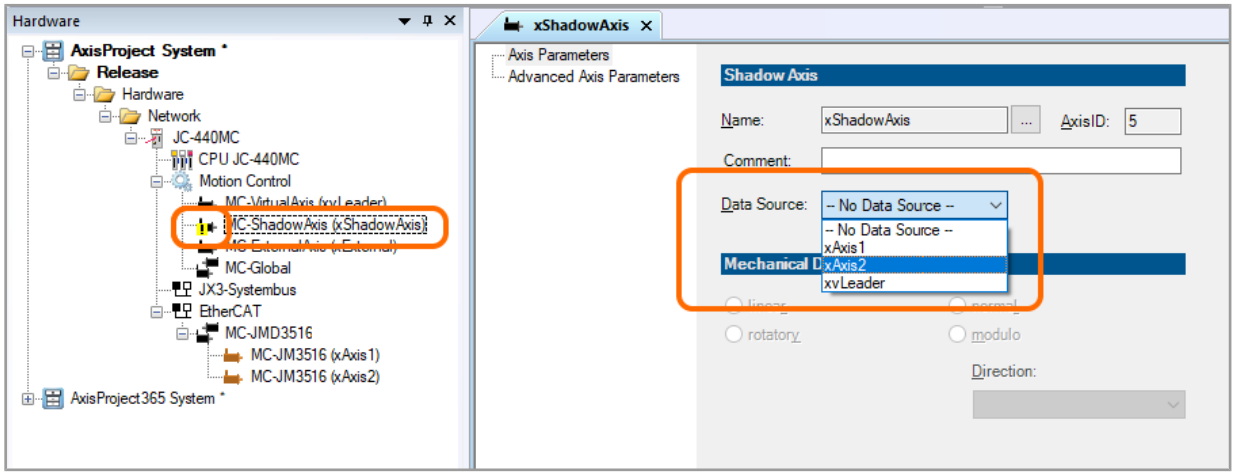
– The external axis does not require any additional settings, such as control parameters, motor type, etc.

– Furthermore, an external axis is always in the “inactive” operating state. This eliminates the need for additional views in MotionSetup.

5.6 Shadow axis



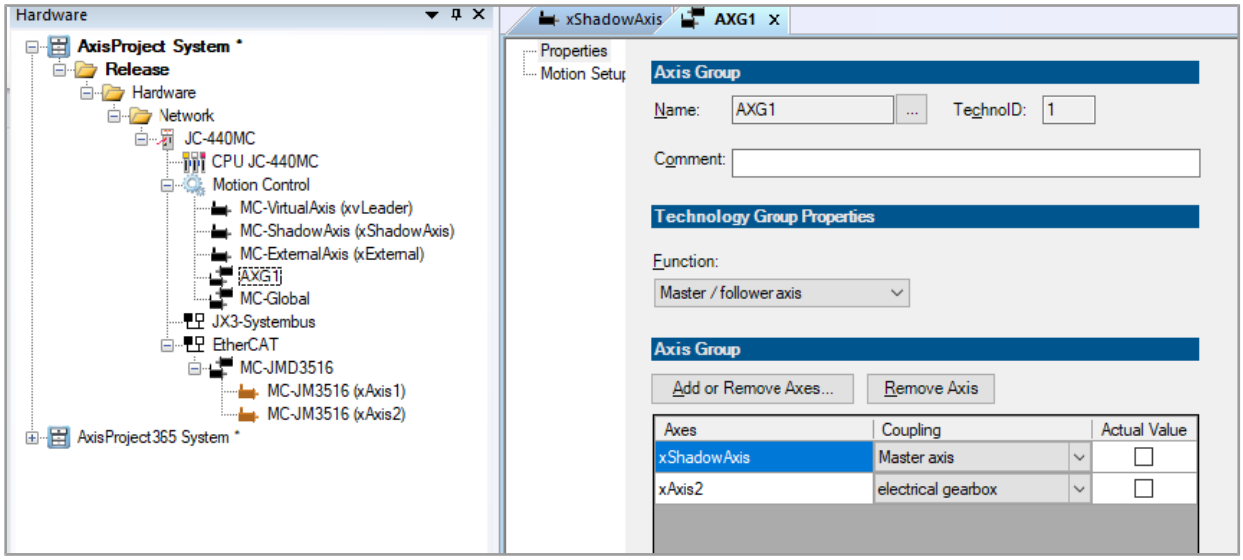
1. Use the submenu on the "Motion Control" node to open the dialog box for selecting axes.
2. Use the Add Shadow Axis option to add it to the hardware configuration.



⇨ A shadow axis is created. By default, this displays an engine icon with an exclamation mark, indicating that the configuration is still incomplete or incorrect.

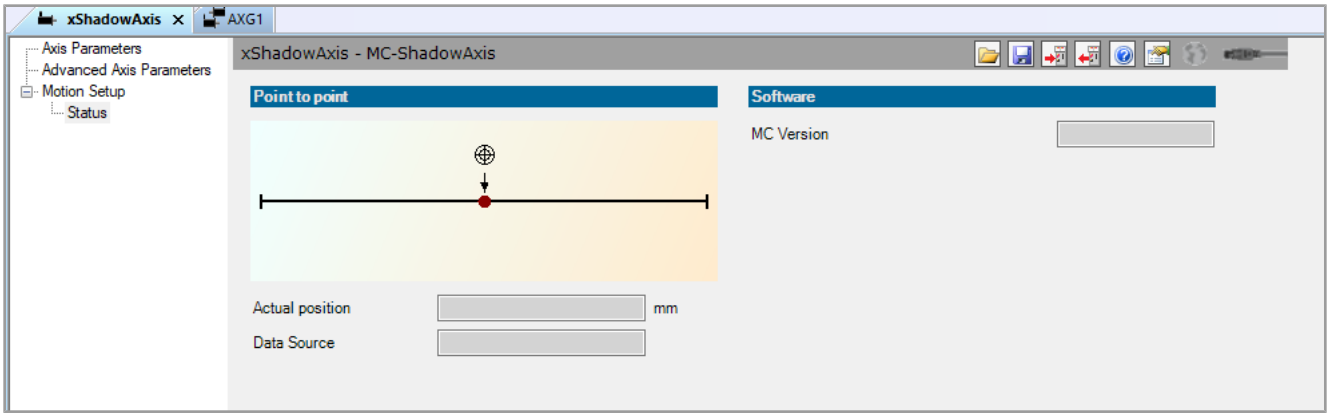
3. Set the data source for the shadow axis on the Axis Parameters page .

Since a shadow axis can only be used as a master axis in a technology group, the configuration is not complete until this shadow axis is integrated as a master axis in a technology object.



5.6.1 MotionSetup Status

Once the configuration of the shadow axis is complete, you can view the actual position in MotionSetup.



– The shadow axis inherits all settings from the source axis, such as the travel range.
– Furthermore, a shadow axis is always in the “inactive” operating state.
– This eliminates the need for additional views in MotionSetup.

6 Programming axes

6.1 Ramp parameter settings

- Real
- Simulation
- Virtual
- External
- Follower

6.1.1 Ramp type

Here you can set which ramp type should be used for positioning during acceleration and deceleration.

```
xAxis1.Mechanism.Slope.SlopeType := AxisSlopeTypes.Sine;
xAxis1.Mechanism.Slope.SlopeType := AxisSlopeTypes.Linear
```

- For the **AxisSlopeTypes.Sine** ramp type, acceleration uses a sine ramp and deceleration uses a sine² ramp.
- For the **AxisSlopeTypes.Linear** ramp type, acceleration uses a linear ramp and deceleration uses a sine ramp.

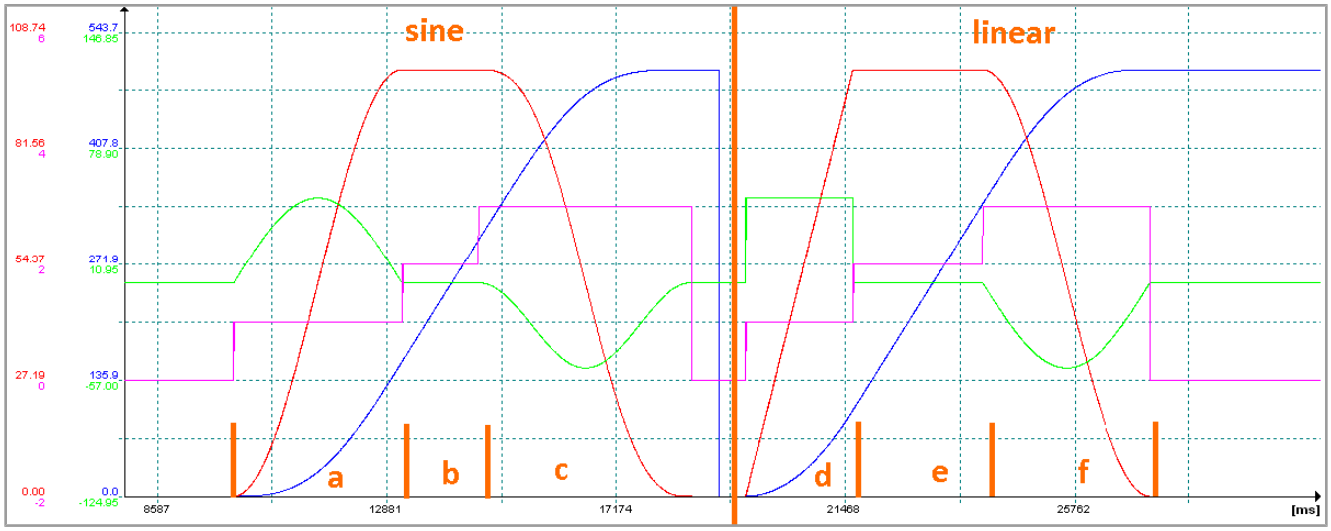


Fig. 2: Ramp type

Legend

- Position setpoint
- Set speed
- Set acceleration
- Ramp (slope) status

***Sine*positioning**		***Linear*positioning**	
a	Sinusoidal acceleration → sinusoidal increase in velocity	**d**	Linear acceleration → linear increase in velocity, constant acceleration
b	Steady-state phase	**e**	Steady-state phase
c	Sine² delay → smooth stopping	**f**	Sine delay

Tab. 2: Positioning Ramp Type

6.1.2 Ramp break mode

Use the **Config.RampBreakMode** property to change the behavior of setpoint generation when a new MovePtp.Start command is issued while the motion is in the deceleration slope.

Ramp configuration	There is still enough distance to come to a stop	Behavior
None	Yes	The first position setpoint is approached. The second position setpoint is approached directly afterwards.
None	No	The first position setpoint is approached. The second position setpoint is approached directly afterwards.
Immediate	Yes	The deceleration ramp is aborted with the newly transferred max. deceleration and the second position setpoint is approached directly.
Immediate	No	The first position setpoint is approached. The second position setpoint is approached directly afterwards.

Tab. 3: Ramp interruption configuration

```
xAxis1.Mechanism.Slope.Config.RampBreakMode := MCAxisRampBreakModes.None;
xAxis1.Mechanism.Slope.Config.RampBreakMode := MCAxisRampBreakModes.Immediate;
```

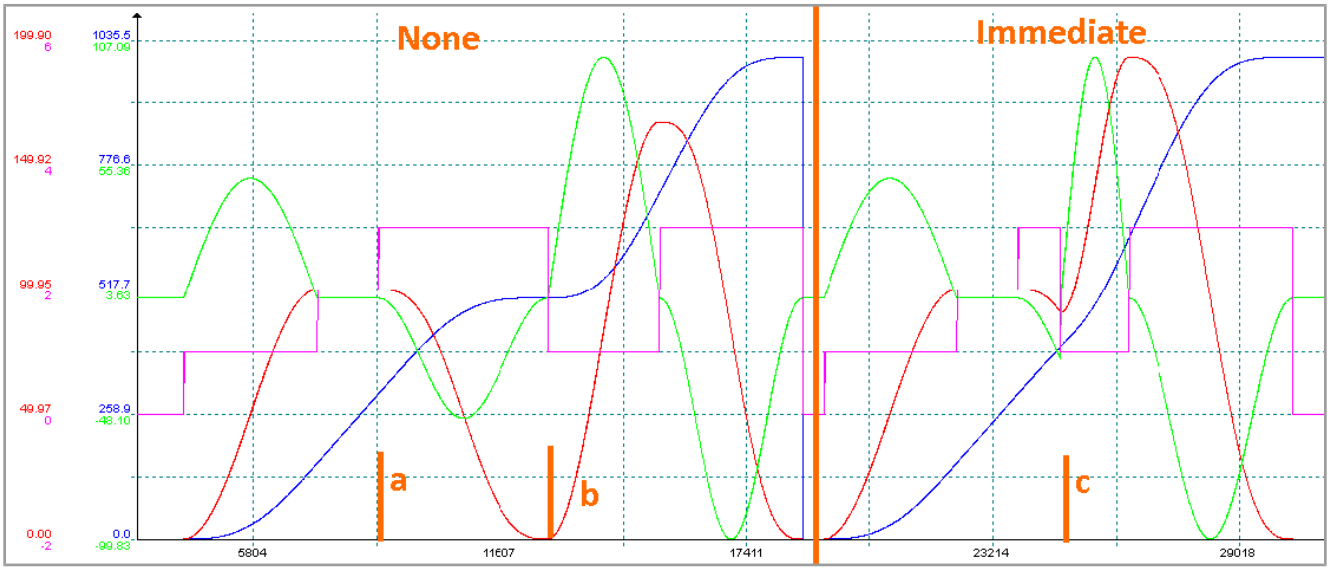


Fig. 3: Ramp interruption configuration

Legend

- Position setpoint
- Set speed
- Set acceleration
- Ramp (slope) status

Position *: None*		*Immediate*Placement	
a	Time at which the new positioning is started in the code.	**c**	Time at which the new positioning is started in the code and in reality. The system cancels the current deceleration and switches to acceleration of the new positioning.
b	Time at which the new positioning is started in reality.		

Tab. 4: Positioning Ramp Break

6.2 Enabling/disabling axes

- Real
- Simulation
- Virtual
- External
- Follower

The MotionAPI's Power object is used to enable and disable axes.

6.2.1 Power.Enable(), Power.Disable();

You can then check the operating status to ensure that the action was successful.

```
xAxis1.Power.Enable();
    when xAxis1.State.IsEnabled continue;
xAxis1.Power.Disable(MCPowerDisableModes.Forced);
    when xAxis1.State.IsDisabled continue;
```

6.2.2 Drive.IsReadyForOperation

- For real axes, it is recommended to check whether the servo amplifier is ready for operation before giving the go-ahead.
- The reason why the servo amplifier is not ready for operation can then be analyzed in more detail in advance and prepared for error output.
- Although it is also possible to send an enable command to a servo amplifier that is not ready for operation, this will result in an axis error.
- Especially in multi-axis applications, all axes should be enabled only when they are ready to be enabled.

```
    if xAxis1.Drive.IsReadyForOperation then
xAxis1.Power.Enable();
    when xAxis1.State.IsEnabled continue;
    else
    // Inform the user that the axis is not operational
    end_if;
```

6.2.3 Power.QuickStop()

As an alternative to direct locking, a quick-stop feature is also available.
In this process, the system stops using the emergency stop ramp in the speed control loop and then locks.

```
    xAxis1.Power.QuickStop();
    when xAxis1.State.IsDisabled continue;
```

6.3 Referencing

6.3.1 Set a standard

- Real
- Simulation
- Virtual
- External
- Follower

With <Axis>.MoveHome.SetReference() an axis is set at the designated position.

```
xAxis1.MoveHome.SetReference(0.0);
when xAxis1.MoveHome.IsReferenceSet continue;
```

After the position is set, the reference status changes to “IsReferenceSet.”

6.3.2 Reference run

- Real
- Simulation
- Virtual
- External
- Follower

With **<Axis>.MoveHome.Start** a reference run is started. Example:

```
xAxis1.MoveHome.Start(
Directions.Positive, // Direction
ReferencingSwitchConfigs.RefAndLimitSwitch, // SwitchConfig
ReferencingModes.SwitchingEdge_2Phase, // Fashion
100.0, // Acceleration
50.0, // SpeedSwitchSearch
10000.0, // MaxDistanceSwitchSearch
10.0, // SpeedReferenceSearch
100.0, // MaxDistanceReferenceSearch
200.0, // ReferenceOffset
0.0 // NormalPosition
);
when xAxis1.MoveHome.IsReferenceSet continue;
```

- All parameters are optional.
- If the referencing run was also tested and the necessary parameters were set during commissioning using MotionSetup, these are also saved in the initialization file.
- After the file is transferred to the controller, the calculated reference parameters are set.

The following command is then sufficient:

```
xAxis1.MoveHome.Start();
when xAxis1.MoveHome.IsReferenceSet , continue;
```

If only some of the reference parameters are to be changed, while the rest are to remain as they were during commissioning or as previously configured, then only the changed parameters are specified.

In this example, the distance to the reference switch and the reference position are to be changed:

```
xAxis1.MoveHome.Start(
    , // Direction
    , // SwitchConfig
    , // Fashion
    , // Acceleration
    , // SpeedSwitchSearch
    , // MaxDistanceSwitchSearch
    , // SpeedReferenceSearch
    , // MaxDistanceReferenceSearch
    m_nNewReferenceOffset, // ReferenceOffset
    m_nNewNormalPosition // NormalPosition
    );
    when xAxis1.MoveHome.IsReferenceSet continue;
```

ⓘ Info

Further information

For more information on this subject, refer to the JetSym Online Help.

Reference run parameters

The reference point can be either the position of the zero pulse or the position of the switch edge if referencing is performed without a zero pulse.

The respective variables are indicated in parentheses.

Direction (<Axis>MoveHome.Direction)

– Directions.Positive

– Directions. Negatives

Specifies the direction in which the reference switch should be searched for.

SwitchConfigs (<Axis>MoveHome.SwitchConfig)

– ReferencingSwitchConfigs.NoSwitch
  – Select this option if you are referring exclusively to K0.

– ReferencingSwitchConfigs.RefAndLimitSwitch
  – Referencing with reference switch and limit switch. If a limit switch is triggered during the reference switch search, the system stops and searches for the reference switch in the opposite direction.

– ReferencingSwitchConfigs.RefSwitch
  – During referencing, only the reference switch is monitored. This is often the case with rotary axes or when limit switches are missing.

– ReferencingSwitchConfigs.LimitSwitch
  – A reference is made to the limit switch. Which limit switch is used depends on the direction of the referencing process. If referencing is started in the positive direction, the system references the positive limit switch.

Mode (<Axis>MoveHome.Mode)

– ReferencingModes.ZeroPulse_2Phase

There are two reference phases:

1. Coarse search phase: usually configured more quickly (SpeedSwitchSearch) to quickly locate the zero pulse K0
2. Fine-tuning phase: usually configured to run slower (SpeedReferenceSearch) in order to more accurately locate the edge of the zero pulse K0

– ReferencingModes.SwitchingEdge_2Phase

There are two reference phases:

1. Preliminary search phase: usually configured more quickly (SpeedSwitchSearch) to find the reference switch promptly
2. Fine-tuning phase: usually configured to run slower (SpeedReferenceSearch) in order to more accurately locate the edge of the reference switch

– ReferencingModes.ZeroPulse_1Phase
  – A search phase: usually configured for slow operation (SpeedReferenceSearch) to more accurately locate the edge of the zero pulse K0

– ReferencingModes.SwitchingEdge_1Phase
  – A search phase: usually configured to run slowly (SpeedReferenceSearch) in order to more accurately locate the edge of the reference switch

Acceleration (<Axis>MoveHome.Acceleration)

Acceleration and deceleration during the referencing process.

SpeedSwitchSearch (<Axis>MoveHome.SpeedSwitchSearch)

Speed during the "initial" search phase. This value is usually set higher than SpeedReferenceSearch in order to locate the reference point quickly.

MaxDistanceSwitchSearch (<Axis>MoveHome.MaxDistanceSwitchSearch)

Maximum distance over which the reference point is to be located during the "rough" search phase. If this distance is reached during the referencing process, an error is generated.

SpeedReferenceSearch (<Axis>MoveHome.SpeedReferenceSearch)

Speed during the "refined" search phase. This value is usually set lower than the SpeedSwitchSearch value in order to determine the edge of the reference point.

MaxDistanceReferenceSearch (<Axis>MoveHome.MaxDistanceReferenceSearch)

Maximum distance over which the reference point is to be located during the "fine" search phase. If this distance is reached during the referencing process, an error is generated.

ReferenceOffset (<Axis>MoveHome.ReferenceOffset)

Distance from the found reference point.
If the flank of the reference point was determined during the referencing, the axis is positioned by this ReferenceOffset. For example, if ReferenceOffset = 0 mm, the tool is positioned to the edge.

NormalPosition (<Axis>MoveHome.NormalPosition)

The position the axis should be at after reaching the ReferenceOffset.

Example:
ReferenceOffset = 100 mm, NormalPosition = 20 mm:
During the reference search, the switch edge is detected and the position is adjusted by 100 mm. The normal position is now set there so that, at the end of the referring run, the axis is positioned at 20 mm.

Monitoring of the reference run

The different phases of the referencing process are represented by a state and can be queried accordingly.

<Axis>MoveHome.IsReferenceRunStarted

Referencing begins, and the system searches for the edge of the reference point.

<Axis>MoveHome.IsSwitchFound

The reference point's transition edge was found during the "rough" search phase. The axis now looks for the cross again during the "fine-tuning" phase.

<Axis>MoveHome.IsReferenceFound

The transition at the reference point was found during the "fine" search phase. The axis now moves to the position determined by the reference offset.

<Axis>MoveHome.IsReferenceSet

The referencing run is complete, and the home position (NormalPosition) has been set.

6.4 Positioning

– Real
– Simulation
– Virtual
– External
– Follower

An axis can be positioned either point-to-point (MovePtp) or continuously (MoveVelocity).

6.4.1 Ramp Status

Positioning consists of four phases:

– Stopped
– Acceleration phase
– Steady-state phase
– deceleration phase

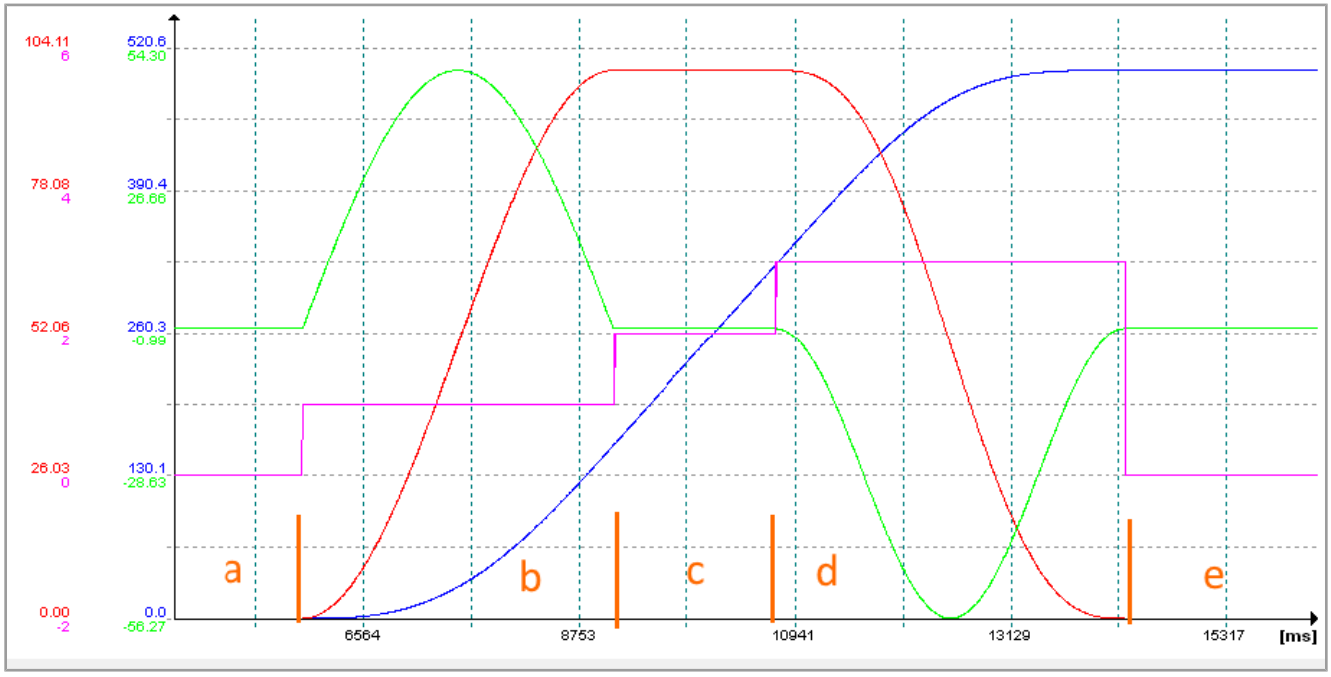


Fig. 4: Ramp Status

Legend

- Position setpoint
- Set speed
- Set acceleration
- Ramp (slope) status

a
Stopped

– The axis has stopped
– Target position does not change
– Setpoint speed and setpoint acceleration are both zero
– The ramp status is paused (0: (IsStopped)

b
Acceleration phase

– Axis accelerates
– Target position changes
– Target speed is increasing
– Target acceleration increases and then decreases again until the target speed is reached
– The ramp status is "accelerating" (1: IsAccelerating)

c
Constant-speed phase

– The axis travels at a constant speed
– The target position is now rising linearly
– Target speed remains constant
– Target acceleration is 0
– The ramp status is "constantly moving" (2: (IsAtConstantSpeed)

d
Delay phase

– Axis delayed
– Target position changes
– Target speed is decreasing
– Target acceleration increases and then decreases again until the target is reached
– The ramp status is "delayed" (3: IsDecelerating)

e
See a.

Depending on the settings of the positioning parameters, constant-speed motion may not be necessary if the system transitions directly from acceleration to deceleration.

Query in STX

```
if xAxis1.Mechanism.Slope.IsAccelerating then
elseif xAxis1.Mechanism.Slope.IsAtConstantSpeed then
elseif xAxis1.Mechanism.Slope.IsDecelerating then
elseif xAxis1.Mechanism.Slope.IsStopped then
elseif xAxis1.Mechanism.Slope.SlopeState == AxisSlopeStates.Stopped then
end_if;
```

6.4.2 Point-to-Point (MovePtp)

– A point-to-point positioning can be achieved with **<Axis>.MovePtp.Start** oder **<Axis>.MovePtp.Start-NewTargetPosition**.
  In this process, the axis moves to the specified destination according to the ramp configuration and the entered parameters.
– By using **<Axis>.MovePtp.Start** is is possible to enter mode, target, speed, acceleration, deceleration and target window.
  If you do not want to change the speed, acceleration, or deceleration, you can alternatively use **<Axis>.MovePtp.StartNewTargetPosition**.

Example: <Axis>.MovePtp.Start

```
xAxis1.MovePtp.Start(
    AxisPositioningModes.AbsNormal, // PositioningMode
    500.0, // TargetPosition
    100.0, // Speed
    50.0, // Acceleration
    50.0, // Deceleration
    1.0 // TargetWindow
);
    when xAxis1.State.IsEnabled and xAxis1.IsInTargetWindow continue;
```

Example: <Axis>.MovePtp.StartNewTargetPosition

```
xAxis1.MovePtp.StartNewTargetPosition (
    AxisPositioningModes.AbsNormal, // PositioningMode
    500.0, // TargetPosition
    1.0 // TargetWindow
);
    when xAxis1.State.IsEnabled and xAxis1.IsInTargetWindow continue;
```

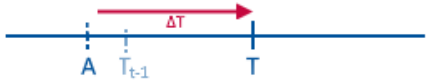
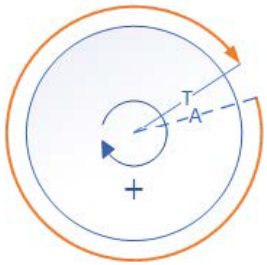
Since the parameters of both functions are optional, the following example can also be used to start a new positioning operation in which the speed, acceleration, and deceleration remain unchanged.

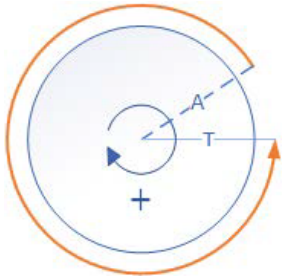
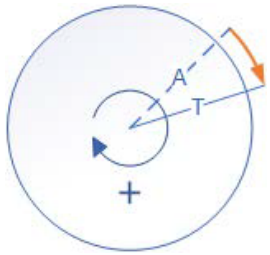
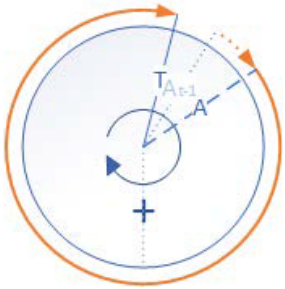
```
xAxis1.MovePtp.Start(
    AxisPositioningModes.AbsNormal, // PositioningMode
    500.0, // TargetPosition
    , // Speed
    , // Acceleration
    , // Deceleration
    1.0 // TargetWindow
);
    when xAxis1.State.IsEnabled and xAxis1.IsInTargetWindow continue;
```

When should I use MovePtp.Start, and when should I use MovePtp.StartNewTargetPosition?

In principle, the functions in the example mentioned above are redundant and should be considered equivalent. It is therefore up to the program developer to decide which function to use. The author of this Application Note uses only MovePtp.Start.

6.4.3 PositioningMode - Positioning Modes

Positioning Type	Description	Graphic	Example
AxisPositioningModes.AbsNormal	– The position setpoint is absolute – Only for standard axes (not modulo)		
AxisPositioningModes.RelTarget	– The new position setpoint is relative to the last position setpoint – For example, if point-to-point positioning was paused before reaching the target, you can move to the last position setpoint using “Start(AxisPositioningModes.RelTarget, 0.0….)”		
AxisPositioningModes.RelActual	– The new position setpoint is relative to the current position – Corresponds to the route being traveled		
AxisPositioningModes.AbsModuloPos	– For Modulo axes – The direction of travel is positive		The axis is at 50.0°; the target is 40.0°, so the axis moves 350° in the positive direction.

Positioning Type	Description	Graphic	Example
AxisPositioningModes.AbsModuloNeg	– For Modulo axes – The direction of travel is negative		The axis is at 40.0°; the target is 50.0°, so the axis moves 350° in the negative direction.
AxisPositioningModes.AbsModuloAuto	– For Modulo axes – The direction of travel is the shortest route		The axis is at 40.0°; the target is 50.0°, so the axis moves 10° in the positive direction.
AxisPositioningModes.AbsModulo-KeepDirection	– For Modulo axes – The direction of travel corresponds to the direction of the last positioning		The axis is currently at 40.0° and positioned at 50°; now it should move to 20° using AbsModuloKeepDirection: It moves 330° in the positive direction.

Tab. 5: Positioning Methods

Change speed (<Axis>.MovePtp.NewSpeed)

– To change the axis speed during a movement, a new speed is specified using the **<Axis>.MovePtp.NewSpeed()** command.
– The vehicle accelerates or decelerates to the new speed at the set acceleration rate.

```
xAxis1.MovePtp.NewSpeed(
    100.0 // Speed
);
```

The new speed is specified using the *Speed* parameter.

Change acceleration (<Axis>.MovePtp.NewAcceleration)

– The **<Axis>.MovePtp.NewAcceleration()** command specifies a new maximum acceleration for point-to-point positioning.
– This will be used for the next positioning.
– An acceleration ramp that is currently active is not changed.

```
xAxis1.MovePtp.NewAcceleration(
    100.0 // Acceleration
);
```

Change deceleration (<Axis>.MovePtp.NewDeceleration)

To change the axis deceleration while deceleration is in progress, a new deceleration value is specified using the
<Axis>.MovePtp.NewDeceleration() command.

```
xAxis1.MovePtp.NewDeceleration(
    100.0 // Deceleration
);
```

6.4.4 Endless positioning (MoveVelocity)

Endless positioning is performed using **<Axis>.MoveVelocity.Start()**. In this process, the axis accelerates to the target speed at the set acceleration.

```
xAxis1.MoveVelocity.Start(
    Directions.Positive, // Direction
    100.0, // Speed
    100.0, // Acceleration
    150.0 // Deceleration
);
```

Direction

Direction of endless positioning.

Speed

Target speed for endless positioning. The value must always be greater than 0.

Acceleration

This value is used to accelerate to the target speed. The value must always be greater than 0.

Deceleration

When endless positioning is paused, this delay is used as long as it is not overridden by the pause command. The value must always be greater than 0.

Reverse direction (<Axis>.MoveVelocity.Reverse)

To reverse the direction of motion, you can use the **<Axis>.MoveVelocity.Reverse()** command. The system then switches to the same speed in the opposite direction using the set acceleration.

```
xAxis1.MoveVelocity.Reverse();
```

As an alternative to the previous command, you can also call **<Axis>.MoveVelocity.Start()** again, but this time specifying the opposite direction.

```
xAxis1.MoveVelocity.Start(
    Directions.Negative, // Direction
    100.0, // Speed
    100.0, // Acceleration
    150.0 // Deceleration
);
```

Change speed (<Axis>.MoveVelocity.NewSpeed)

To change the axis speed while a movement is in progress, a new speed is specified using the **<Axis>.MoveVelocity.NewSpeed()** command. The vehicle accelerates or decelerates to the new speed at the set acceleration rate.

```
xAxis1.MoveVelocity.NewSpeed(
    100.0 // Speed
);
```

The new speed is specified using the *Speed* parameter. Since this value must always be greater than 0, the speed can be changed, but not the direction of motion!

Change acceleration (<Axis>.MoveVelocity.NewAcceleration)

To change the acceleration of the axis during an ongoing acceleration, a new acceleration is specified using the **<Axis>.MoveVelocity.NewAcceleration()** command.

```
xAxis1.MoveVelocity.NewAcceleration(
    100.0 // Acceleration
);
```

Change deceleration (<Axis>.MoveVelocity.NewDeceleration)

To change the deceleration of the axis while deceleration is in progress, a new deceleration value is specified using the
<Axis>.MoveVelocity.NewDeceleration() command.

```
xAxis1.MoveVelocity.NewDeceleration(
    100.0 // Deceleration
);
```

6.4.5 Stop (MoveHalt)

– An ongoing motion can be paused using **<Axis>.MoveHalt.Start**.
– The parameters are optional. If these are not specified, the parameters currently in effect will be used.
– It will pause for the specified delay. If this parameter is left blank, the system will use, for example, the delay that was used in the previous motion command.

```
xAxis1.MoveHalt.Start(
    MCAxisHaltModes.Normal, // Fashion
    85.0 // Deceleration
);
when xAxis1.State.IsEnabled continue;
```

You can check whether the pause has ended by querying **<Axis>.State.IsEnabled**.

```
when xAxis1.Mechanism.Slope.IsStopped continue;
```

The difference is this:

– If **<Axis>.State.IsEnabled** is true, the axis is enabled.
– When **<Axis>.Mechanism.Slope.IsStopped** is true, the axis is indeed stopped, but its operating status is ambiguous. This could be, for example, `IsEnabled`, but it could also be `IsDisabled` or something else.

Mode

MCAxisHaltModes.Normal

The current motion will be paused using the active delay starting at this point.

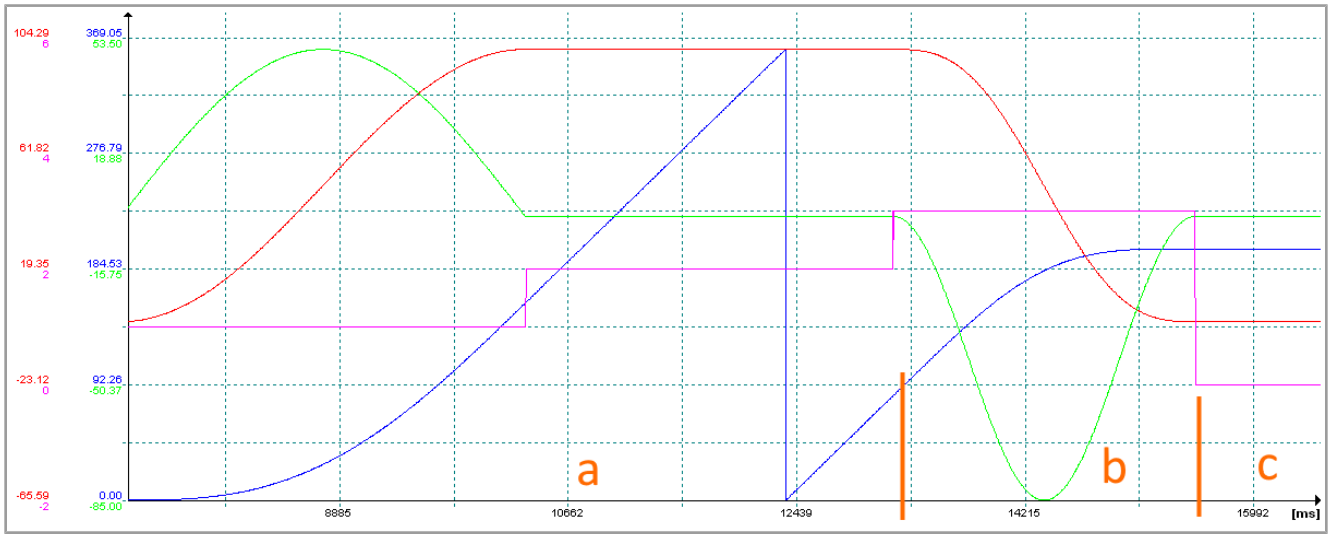


Fig. 5: Normal mode

Legend

- Position setpoint
- Set speed
- Set acceleration
- Ramp (slope) status

a

Starting continuous positioning.

b

– The axis is stopped using MCAxisHaltModes.Normal.
– The axis decelerates until it comes to a stop.
– The ramp status changes to "IsDecelerating".

c

– The axis has now stopped.
– The ramp status is now "IsStopped".

MCAxisHaltModes.AtActualPosition

The current movement is paused, and the robot returns to the point where the command was issued.

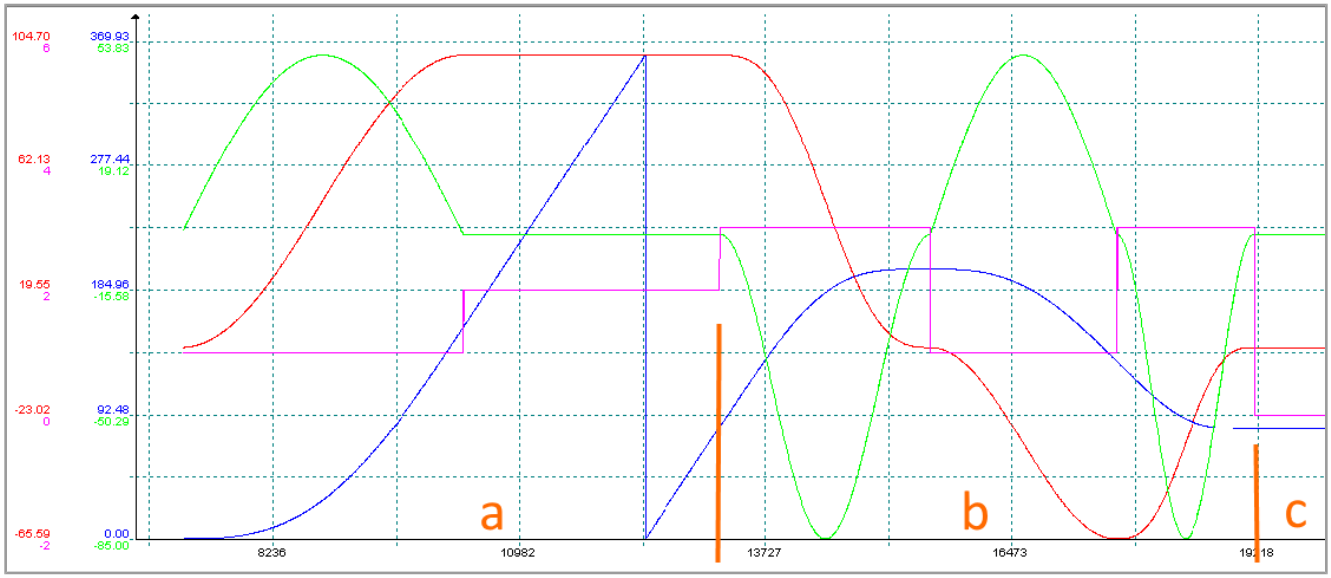


Fig. 6: Fashion: Current Location

Legend

- Position setpoint
- Set speed
- Set acceleration
- Ramp (slope) status

a

Starting continuous positioning.

b

– The axis is stopped using MCAxisHaltModes.AtActualPosition.
– First, the axis decelerates until it comes to a stop. Next, the object is moved to the position at the beginning of **b**.
– The ramp status changes to “IsDecelerating,” followed by “IsAccelerating” and “IsDecelerating” during repositioning.

c

– The axis has now stopped at the position at the beginning of **b**.
– The ramp status is now "IsStopped".

List of Figures

Fig. 1 Mech. Setting measurement units 11
Fig. 2 Ramp type 30
Fig. 3 Ramp interruption configuration 31
Fig. 4 Ramp Status 37
Fig. 5 Normal mode 44
Fig. 6 Fashion: Current Location 45

List of Tables

Tab. 1 Prefixes Used 5
Tab. 2 Positioning Ramp Type 30
Tab. 3 Ramp interruption configuration 31
Tab. 4 Positioning Ramp Break 32
Tab. 5 Positioning Methods 40

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach/Neckar, Germany
T +49 7141 2550-0
info@bucherautomation.com

www.bucherautomation.com