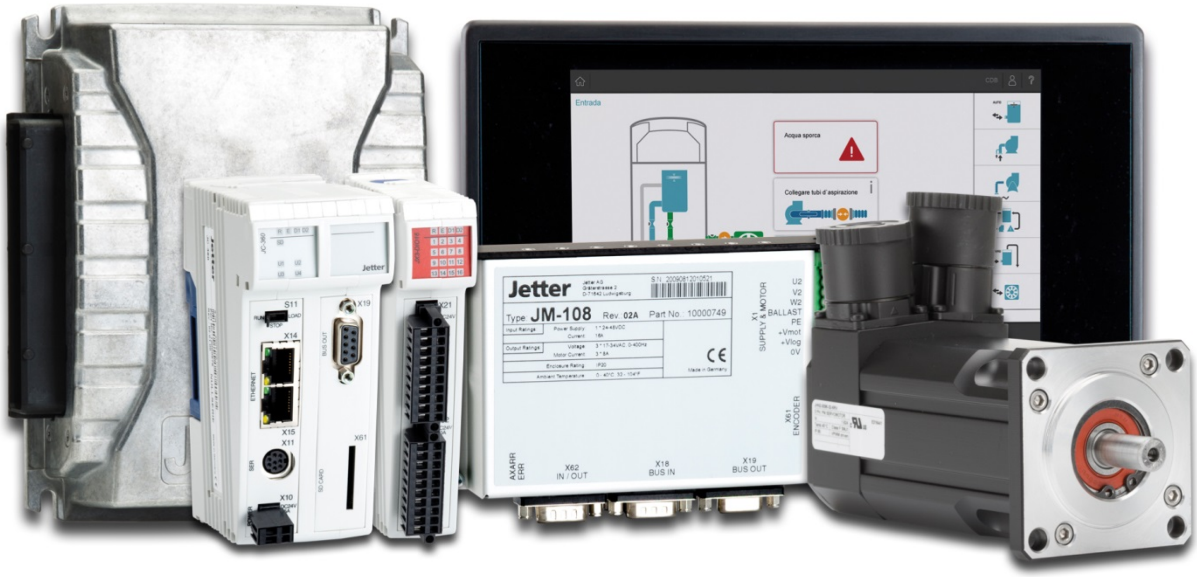


Motion Control - Technology Group

Application Note 049

608 847 48_01

We automate your success.

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg
Germany

www.jetter.de

Phone:

Switchboard	+49 7141 2550-0
Sales	+49 7141 2550-531
Technical hotline	+49 7141 2550-444

E-mail:	info@jetter.de
Technical hotline	hotline@jetter.de
Sales	vertrieb@jetter.de

Product name	Motion Control - Technology Group
Document type	Application Note 049
Translation of the original German language document	
Document revision	1.01
Item number	608 847 48_01
Date of issue	2021-06-23

Table of Contents

1 Introduction

1.1 Prerequisites

The following versions were used for the code and project examples as well as screenshots:

- JetSym 5.60

• JC-440MC, OS 1.08.0.00 • Motion API 2.0.0.4	• JC-365MC, OS 1.33.0.00 • Motion API 1.0.0.16

In this application note, the above controllers are used as examples to represent the different Motion APIs 1 + 2.
If there are differences in their use, these are mentioned in the document.

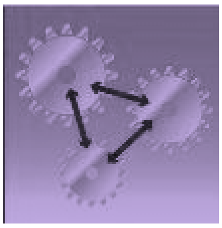
1.2 What is a technology group?

- In a technology group, individual axes can be combined to form a network in order to coordinate the movement of the individual axes. In a technology group, a master axis is combined with follower axes. The follower axes can be dynamically coupled and uncoupled.
- A group always consists of one master axis and 1 ... n follower axes.

1.3 Types of coupling

1.3.1 Electrical gearbox

The term "electrical gearbox" is used in the figurative sense for an electronic gear shift which, like a mechanical transmission, causes a change in speed and position. In contrast to a mechanical gearbox, an electrical gearbox can continuously change the transmission ratio. Dynamic coupling and decoupling processes as well as corrections of position offsets can be more easily realized with the electrical gearbox.



- An electrical gearbox is created when you define a technology group. The gear ratio between the master axis and a follower axis can be specified dynamically and individually with a quotient when the technology group is activated.
- Superimposed movements are possible for the follower axis.

1.3.2 Cam

Cam disk drives are used when the position of one or more follower axes depends on the position of a master axis and complex motion sequences in cyclically running machines have to be coordinated. Mechanical cam gears are usually used for this purpose.

- Electric cam discs offer a much greater flexibility, since it is possible to switch very quickly to other cam profiles and the moved mass is smaller.
- Higher dynamics and higher cycle rates are achieved, the drives are smaller and wear is reduced.
- An electronic cam disk can be switched on and off more flexibly and moved via any offset. With a mechanical cam disk, offset is only possible with greater effort by rotating the cam body on the drive shaft.

1.3.3 Actual value/setpoint coupling

An axis object usually has a position setpoint and an actual position value.

- The position setpoint is calculated cyclically by the MCX and transferred to the position controller of the axis for control.
- The actual position value describes the position at which the axis is currently located.

With ***actual value coupling***, the follower axes are coupled to the actual position value of the master axis.
If the master axis is a real axis or an external encoder, the actual position is not exactly constant even at standstill. Any fluctuations are transmitted to the follower axes as guide value, which can lead to unsteady behavior. To reduce this effect, an appropriate filter for smoothing can be set. However, a filter also delays the reactions of the follower axes.
Furthermore, the transmission of the actual value to the MC kernel causes a delay, which also delays the reaction of the follower axes.
These delays are mainly noticeable during speed changes of the master axis.
Actual value coupling is used to synchronize a follower axis, e.g. with the aid of a friction wheel (master axis), with a passing product or a transport subject to slip.

With ***setpoint coupling***, the follower axes are coupled to the setpoint position of the master axis.
Due to the calculation of the position setpoint by the MCX, this value is accurate and always constant, e.g. at standstill. A filter for smoothing the position setpoint is not necessary.
A new position setpoint is available to the follower axes in the same MCX calculation cycle so that no delays occur. Speed changes are thus immediately passed on to the follower axes.

Therefore, a setpoint coupling is the preferred solution in most cases. In cases where an external encoder is used, e.g. a friction wheel, only the actual value coupling can be used.

1.4 Types of master axes

- Real axis
  - Has a servo amplifier, except as simulation axis
  - The setpoints are calculated by the Motion Control
  - The actual values are returned by the servo amplifier
  - Can assume all MCX operating states, such as "Enabled" or "Running". See also the overview of the MCX operating states
- Virtual axis
  - Has no servo amplifier
  - The setpoints are calculated by the Motion Control
  - Actual value = setpoint
  - Can assume all MC operating states, such as "Enabled" or "Running". See also the overview of the MCX operating states
- External axis
  - Has a counter module, i.e. pure encoder evaluation
    - In case of JC-440MC, JC-945MC, or JC-975MC with EtherCAT the additional encoder input on a drive, such as JM-3000, serves as counter module
    - In case of JC-647MC, JC-365MC, JC-940MC, or JC-970MC an additional servo amplifier, such as JM-105, JM-108, JM-2xx, serves as counter module
  - Setpoints are not calculated
  - Actual values are reported back by the counter module
  - MC operating status is always "Inactive"
- Shadow axis
  - Has no servo amplifier
  - Actual values and setpoints are taken from the source axis
  - MC operating status is always "Inactive"

1.5 Types of follower axes

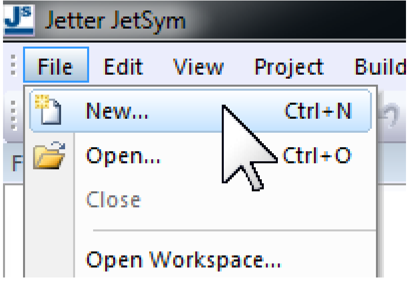
- Real axis
- Virtual axis

2 Creating a Technology Group

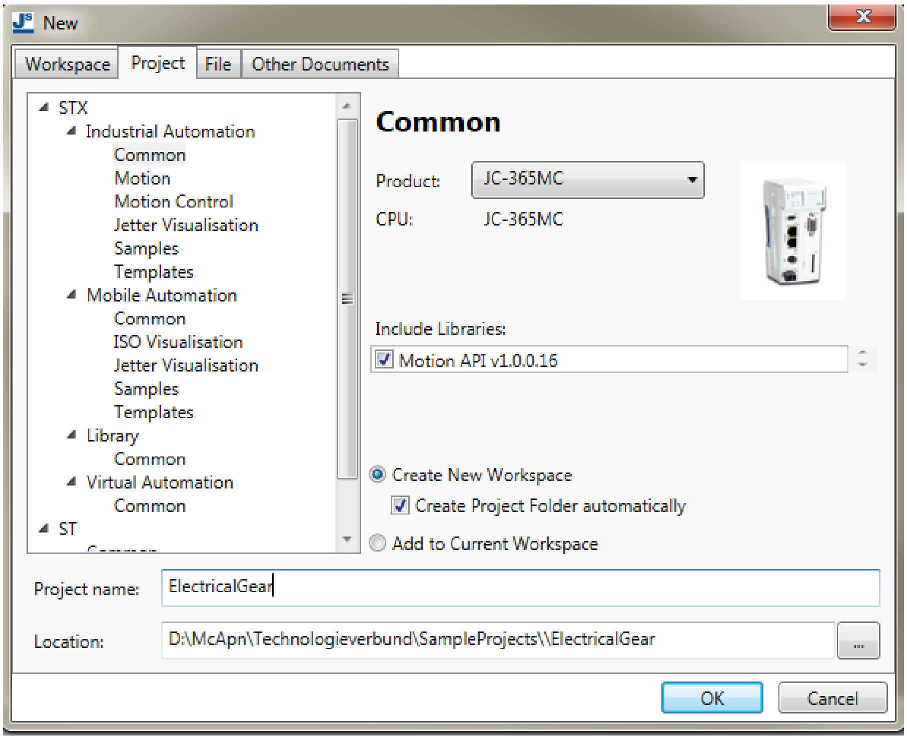
2.1 How to create a technology group

The example of a new project shows you how to create a technology group.

- Open JetSym
- Open the *"File*" menu and click the button *"New*"



- The dialog for creating a project opens.
  Select the "*JC-440MC*" or "*JC-365MC*" controller and include the proposed Motion API library.
  After entering a project name and selecting the appropriate folder in which the project is to be saved, confirm your entries by clicking *"OK*"



Common

Product: JC-365MC

CPU: JC-365MC

Include Libraries:

☑ Motion API v1.0.0.16

● Create New Workspace

☑ Create Project Folder automatically

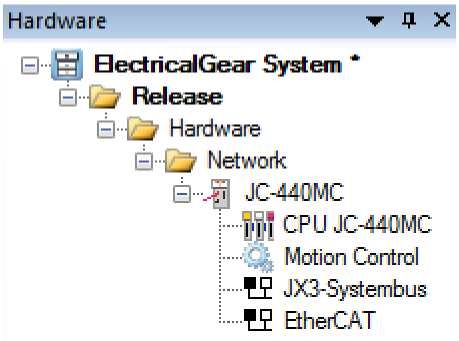
○ Add to Current Workspace

Project name: ElectricalGear

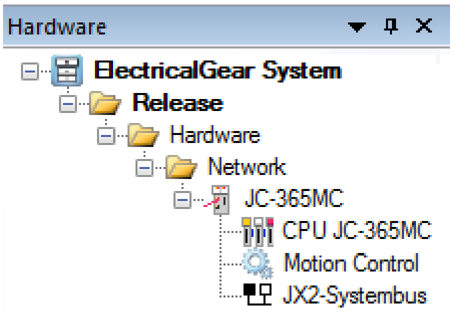
Location: D:\McApn\Technologieverbund\SampleProjects\\ElectricalGear

- A basic structure is created. The hardware tab contains this tree structure.

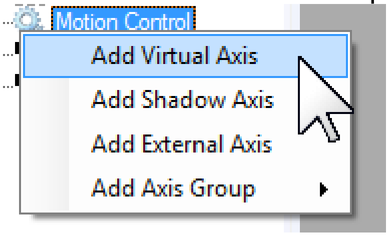
JC440MC:



JC365MC:

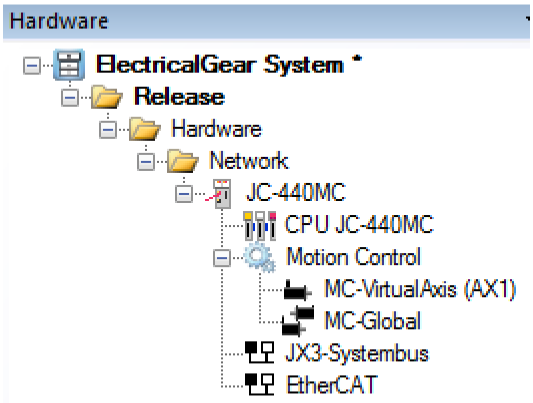


- First, create a virtual axis as the master axis. The virtual axis is created via the "*Add Virtual Axis*" shortcut menu which opens by right-clicking on the "*MotionControl*" node.

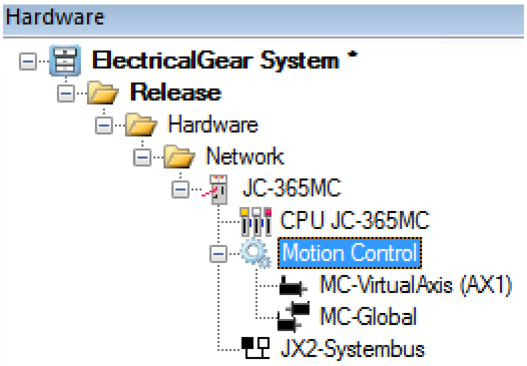


- This is now displayed in the hardware tree:

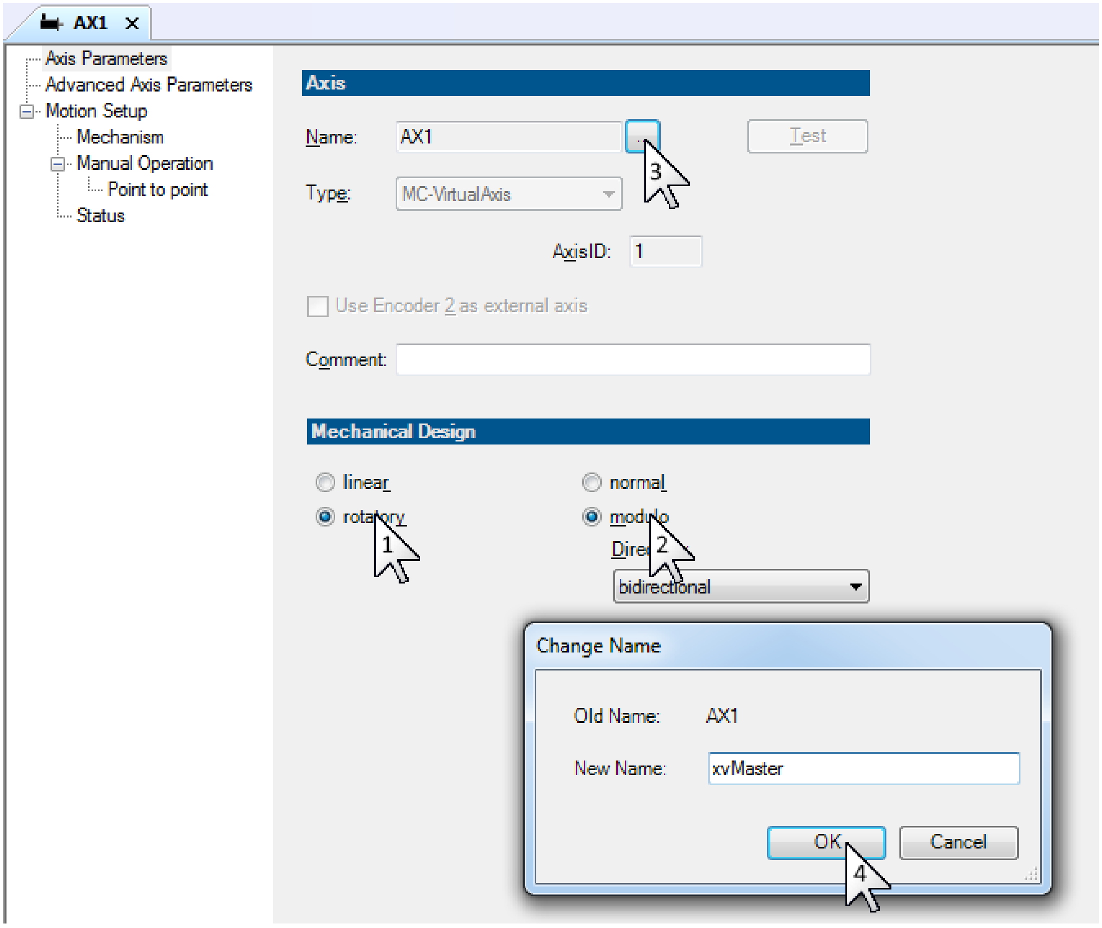
JC440MC:



JC365MC:



- Double-click on the "MC-VirtualAxis (AX1)" node to open the Motion Setup for this virtual axis.



Name: AX1

Type: MC-VirtualAxis

AxisID: 1

Comment:

Change Name

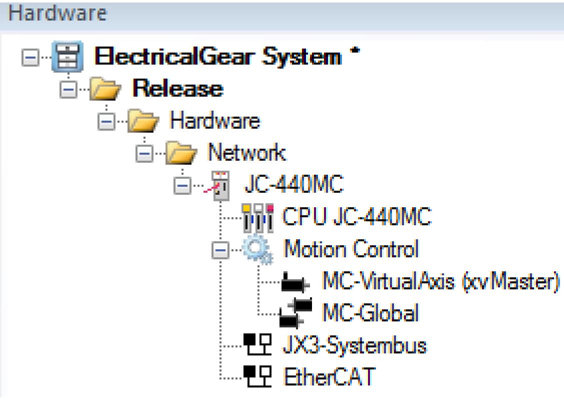
Old Name: AX1

New Name: xvMaster

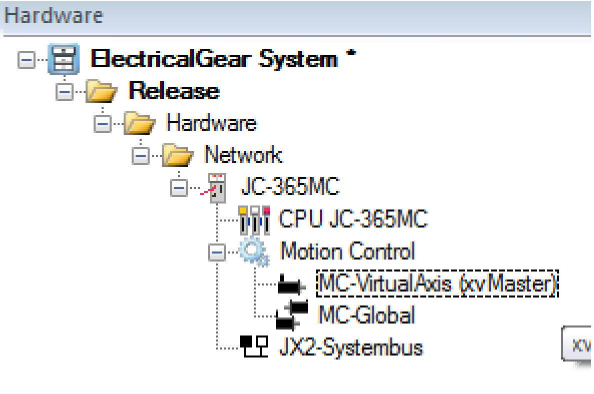
Set the *"Mechanical Design*" to *"rotatory*" (1) and *"modulo*" (2). Click the "*...*" (3) button to rename the axis. In this example, assign the axis the new name "*xvMaster axis*" and confirm this by clicking "*OK*" (4).
Confirm the further dialogs with *"Yes*".
Within the STX project, a MotionAPI object is created which lets you address the virtual axis.

- The new name of the axis now also appears in the hardware tree.

JC440MC:

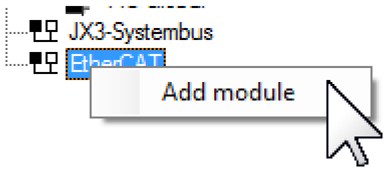


JC365MC:

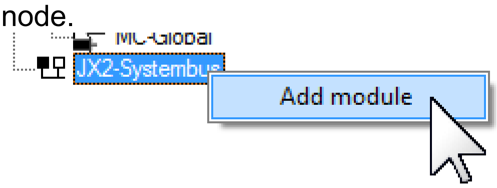


- Now add a real axis.

  JC440MC: To do this, open the shortcut menu by right-clicking on the "EtherCAT" node.
  JC365MC:

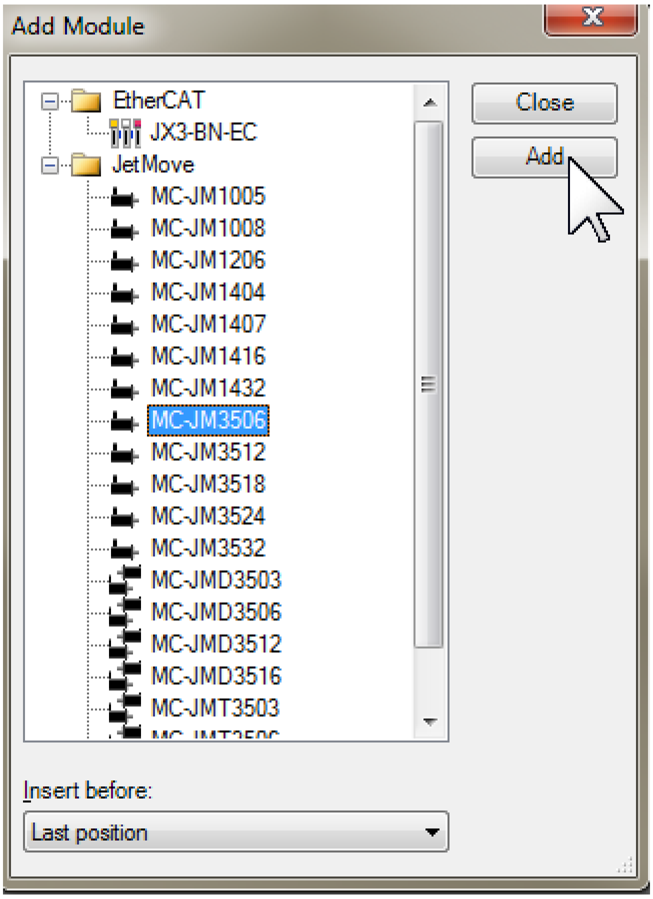


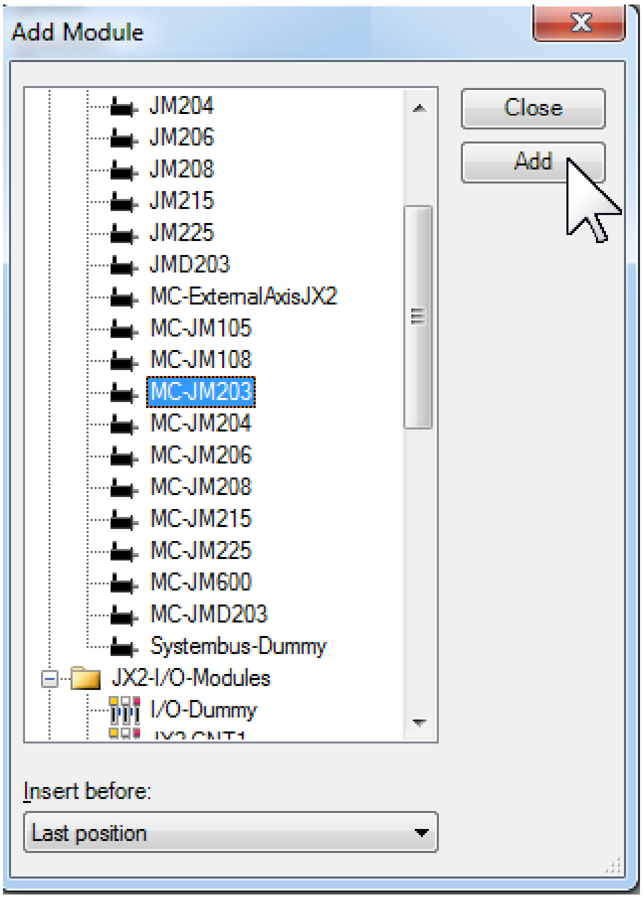
  JC365MC: To do this, open the shortcut menu by right-clicking on the "JX2 system bus" node.



- Now add the required number of axes. For this purpose, please select motion control axes with the prefix "*MC*", such as "*MC-JM203*" and click “Add”. One axis is sufficient for this example. Then close the dialog box.

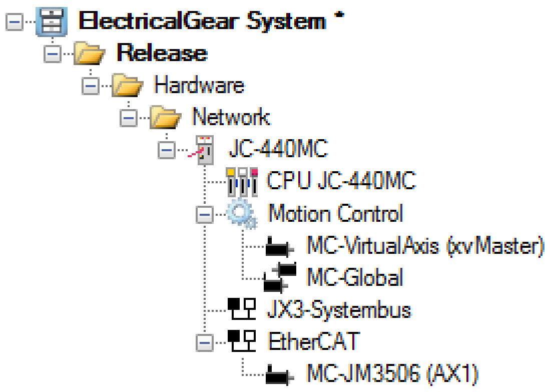
  JC440MC: JC365MC:



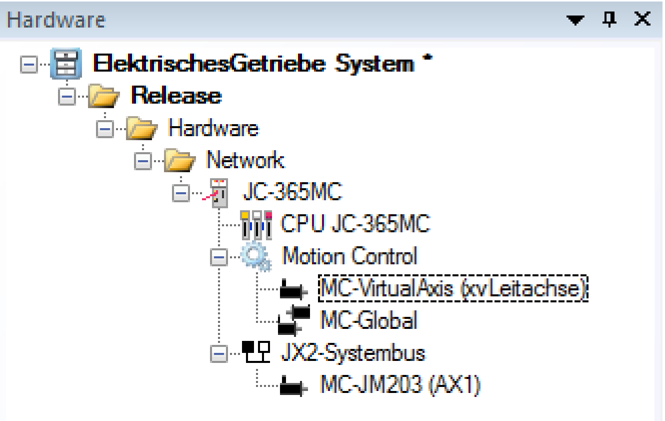


The hardware tree now shows the created axes as subnodes of the selected JX2 bus.

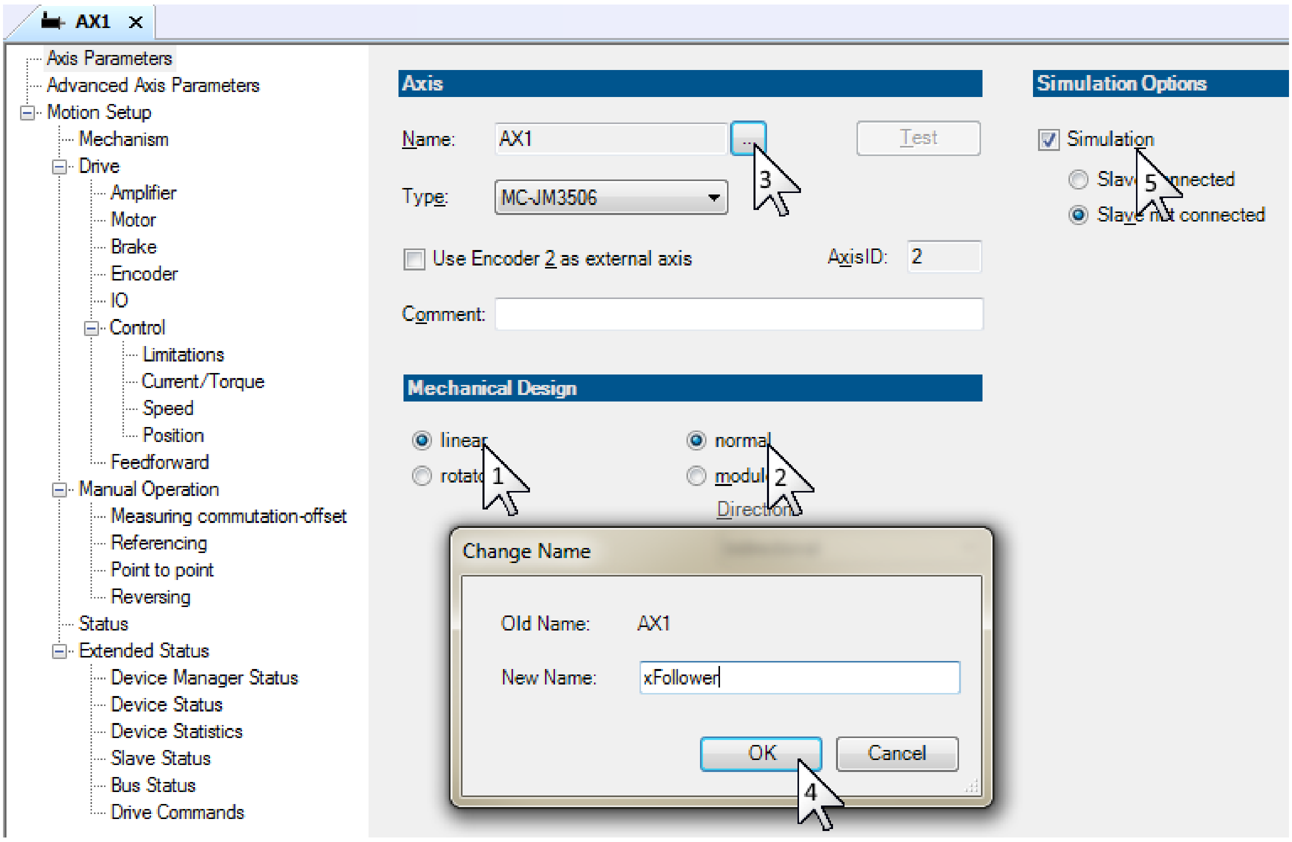
JC440MC:



JC365MC:



- To rename and configure the axis, open the motion setup of the "*AX1*" axis by double-clicking on the node.
- JC440MC:



Name: AX1

Type: MC-JM3506

☐ Use Encoder 2 as external axis

AxisID: 2

Comment:

● linear ○ rotatory

● normal ○ modulo

☑ Simulation

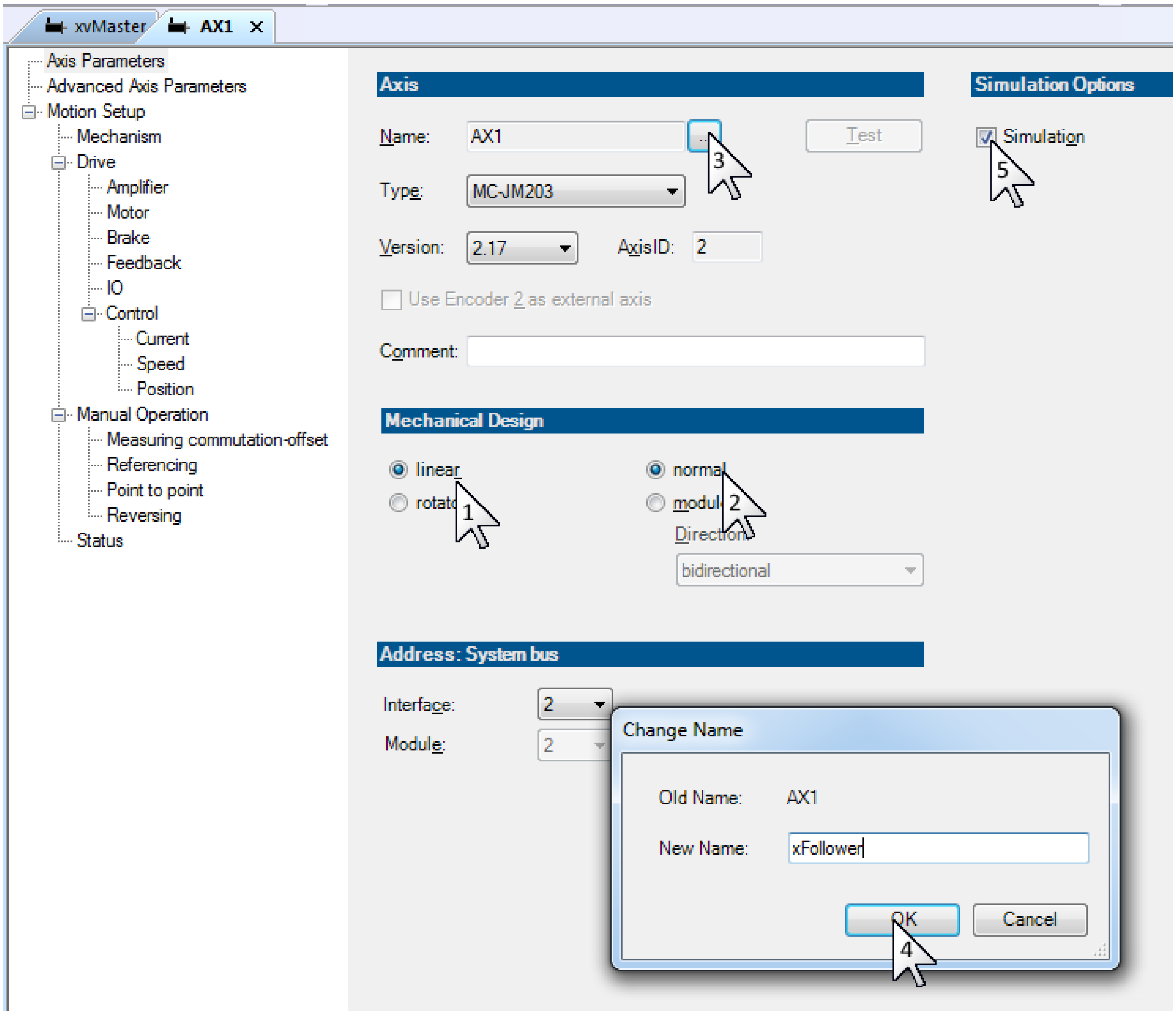
○ Slave connected

● Slave not connected

Old Name: AX1

New Name: xFollower

- JC365MC:

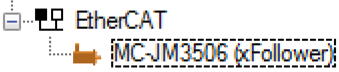


Name: AX1
Type: MC-JM203
Version: 2.17
AxisID: 2
☐ Use Encoder 2 as external axis
Comment:
☑ Simulation
● linear
○ rotatory
● normal
○ modulo
Direction: bidirectional
Interface: 2
Module: 2
Old Name: AX1
New Name: xFollower

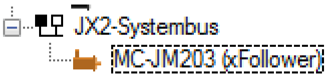
Select the appropriate "Mechanical Design" according to your application, i.e. "*linear*" or "*rotatory*" (1) and "*normal*" or "*modulo*" (2).
Click the "..." (3) button to rename the axis. In this example, name the axis "*xFollower*" and confirm it by clicking "*OK*" (4).
Confirm the further dialogs with *"Yes*".
In addition, check *"Simulation*" (5) to select whether this is a simulation axis or not.
Within the STX project, a MotionAPI object is created which lets you address the axis.

- The changed name is now also displayed in the hardware tree.

JC440MC:

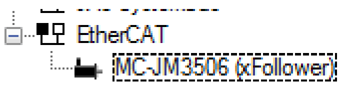


JC365MC:

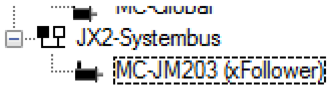


- The color of the axis symbol in the hardware tree shows whether the axis is a simulation axis (orange) or a real axis (black).

JC440MC:

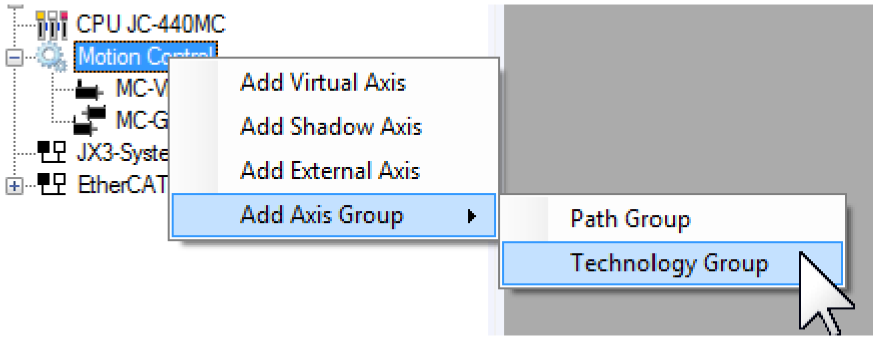


JC365MC:

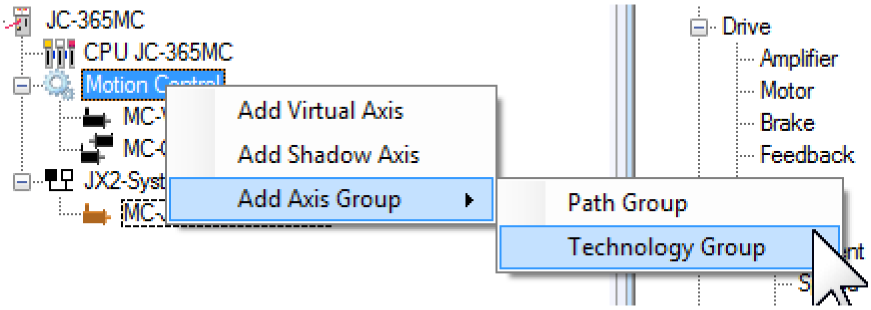


- Open the shortcut menu by right-clicking on the *"MotionControl*" node and select *"Add Axis Group*". Then select "*Technology Group*".

JC440MC:

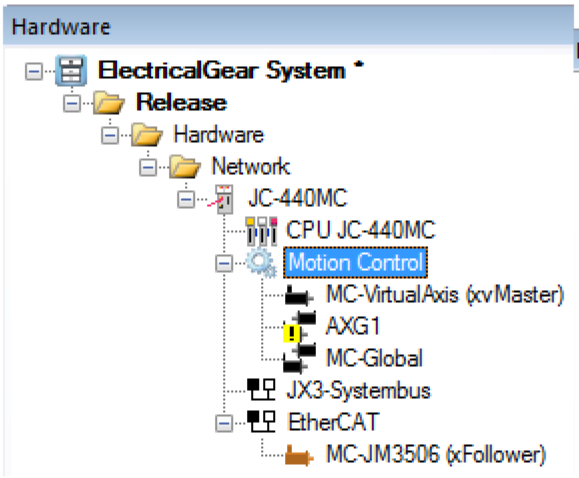


JC365MC:

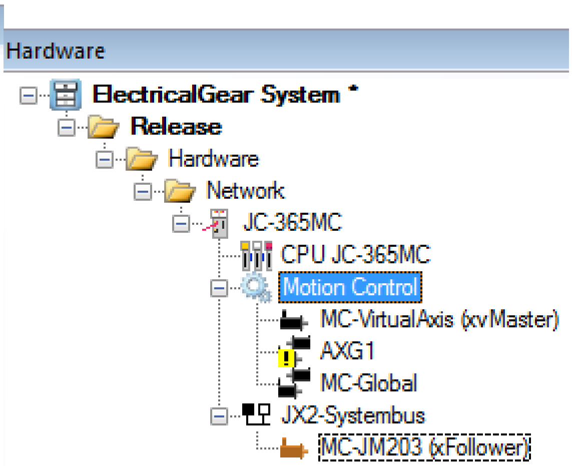


- The new technology group is now displayed in the hardware tree. Since the technology group has not yet been configured, it is marked with a warning sign in the hardware tree. Once a valid configuration is available, this warning sign disappears.

JC440MC:

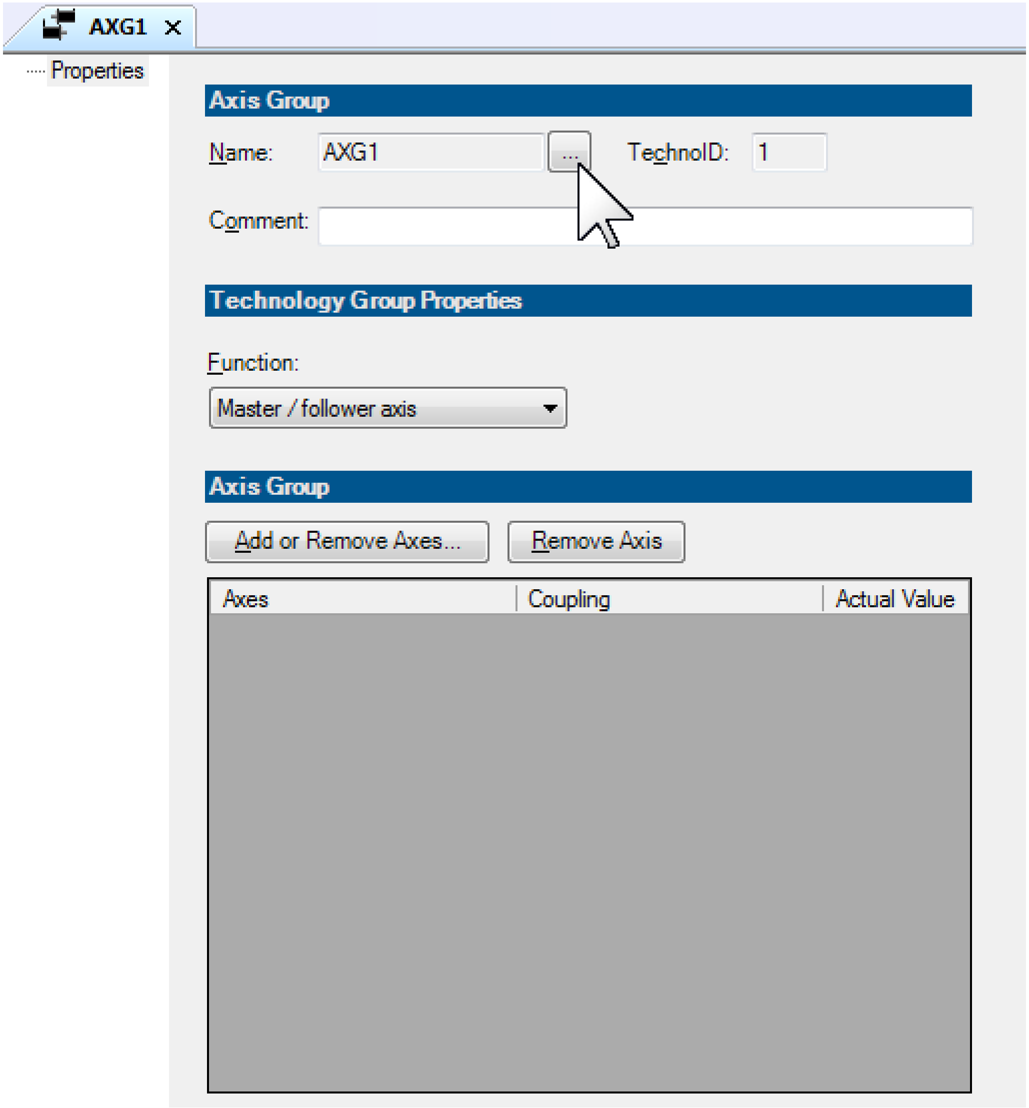


JC365MC



2.2 Configuring the technology group

Double-click on the newly created group to open the settings for the technology group.



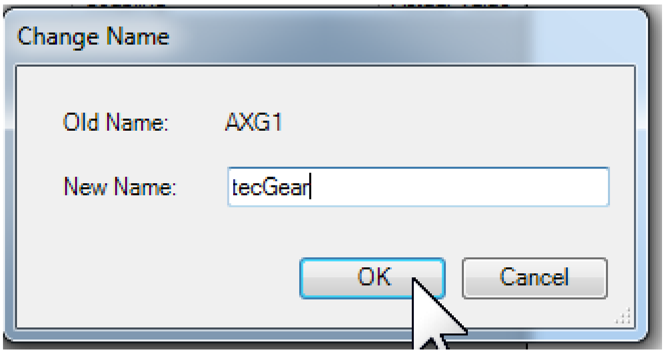
Name: AXG1

TechnoID: 1

Comment:

Function: Master / follower axis

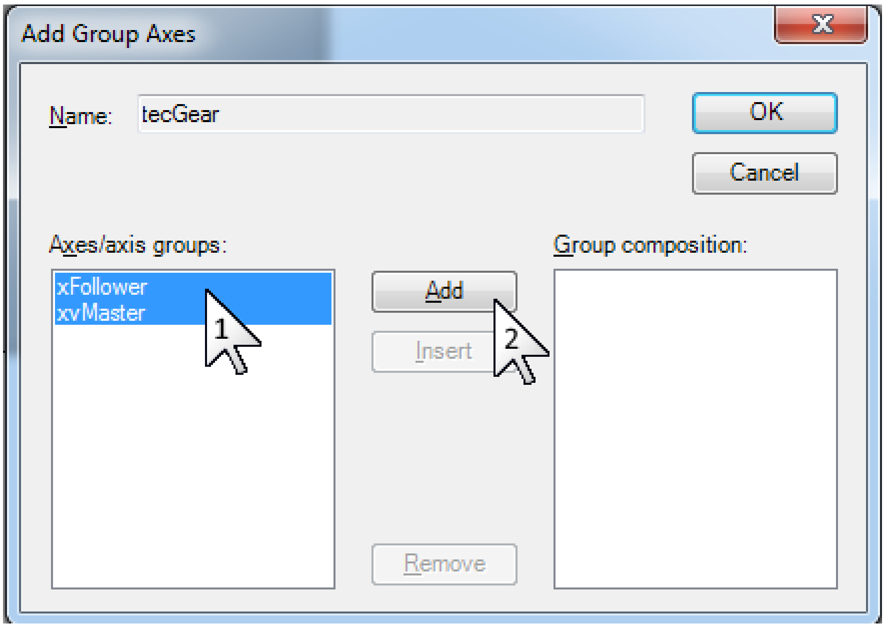
By clicking on the "…" button, you can now assign a descriptive name to the technology group. A MotionAPI object with this name is created within the STX project. Confirm the renaming of all related objects by JetSym so that the new name is also applied in the STX project.



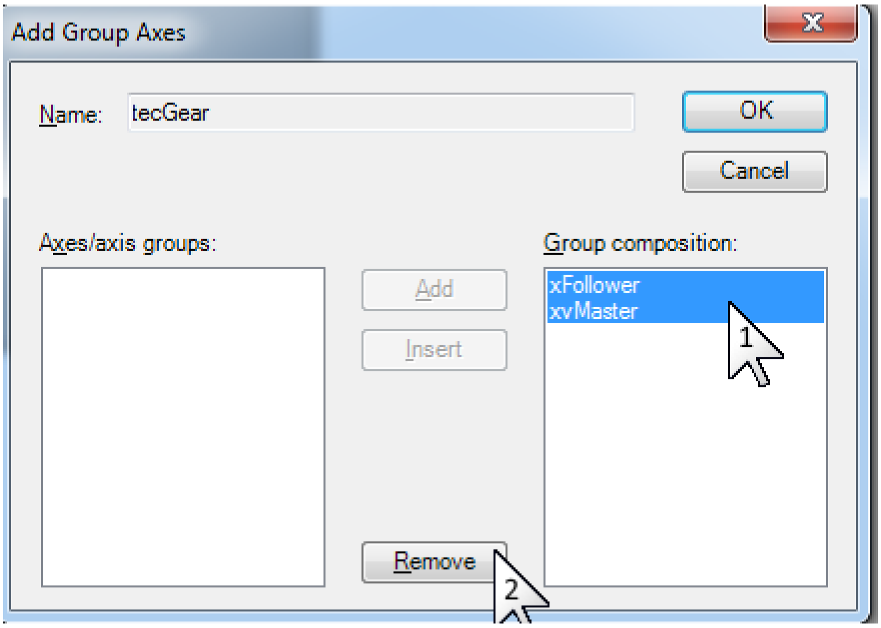
Old Name: AXG1

New Name: tecGear

In the Axis Group" section click on *"Add or Remove Axes*" to specify the axes involved in the group. The following dialog box appears:



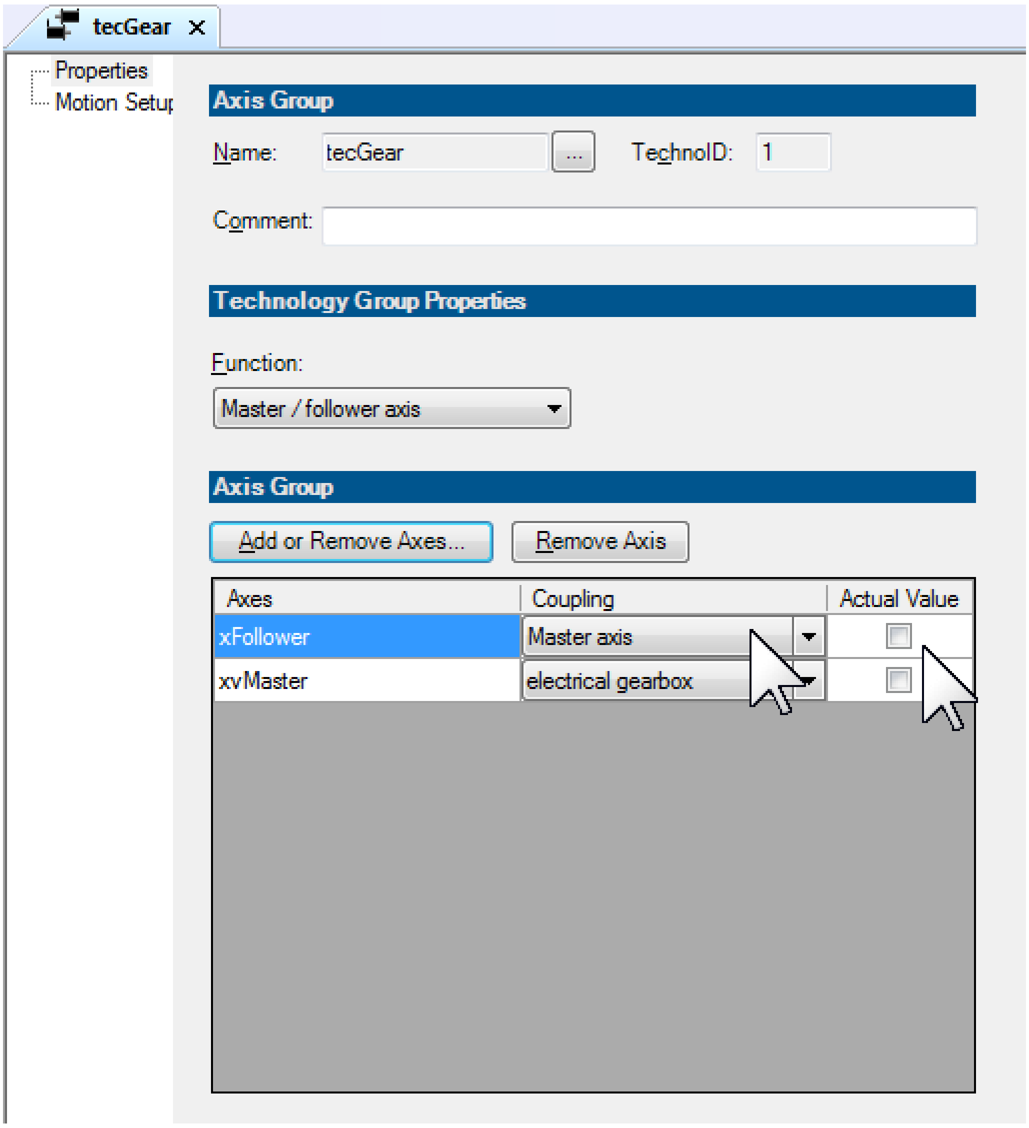
Select the axes (1) to be included by clicking on "*Add*" or by double-clicking on the respective axis. This moves them to "Group composition:" which lists all axes involved in the group. The assignment of the respective tasks takes place in the higher-level view.



If you want to remove any axes from group, you can select them (1) and delete them from the list by "*Remove*()" (2) or double-clicking them in the “Group composition” window.

When you configure a technology group for the first time, the first axis in the list is suggested as the master axis.

Confirm your selection by clicking on "OK" (3).



Axis Group

Name: tecGear TechnoID: 1

Comment:

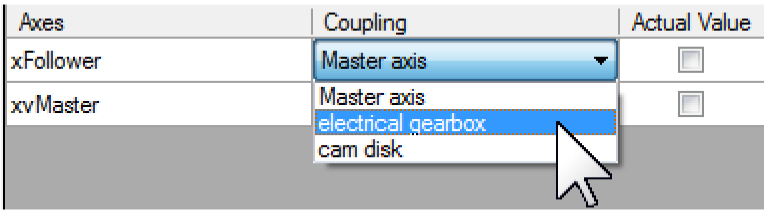
Technology Group Properties

Function: Master / follower axis

Axis Group

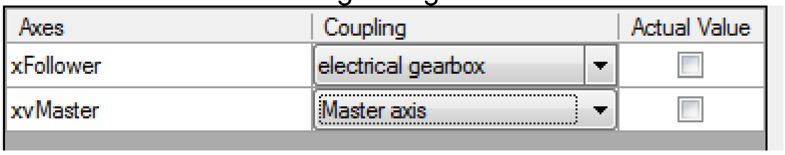
Axes	Coupling	Actual Value
xFollower	Master axis	☐
xvMaster	electrical gearbox	☐

Now you set the coupling type and whether an actual value coupling is to be used for each included axis.
In the screen below, set the "*xFollower*" axis coupling type to "*Eelectrical gearbox*" and the "*xvMaser*" to "*Master axis*".



Axes	Coupling	Actual Value
xFollower	Master axis	☐
xvMaster		☐

This results in the following configuration:



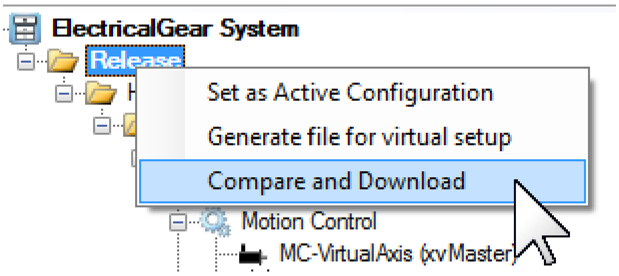
Axes	Coupling	Actual Value
xFollower	electrical gearbox	☐
xvMaster	Master axis	☐

Please note:
The master axis does not necessarily have to be in the first position. The order is arbitrary.

The configuration supports only one master axis. JetSym will inform you of such a configuration error in the hardware tree by a warning symbol and a message in the tooltip. This message also appears in the "Compare and Download" dialog box.

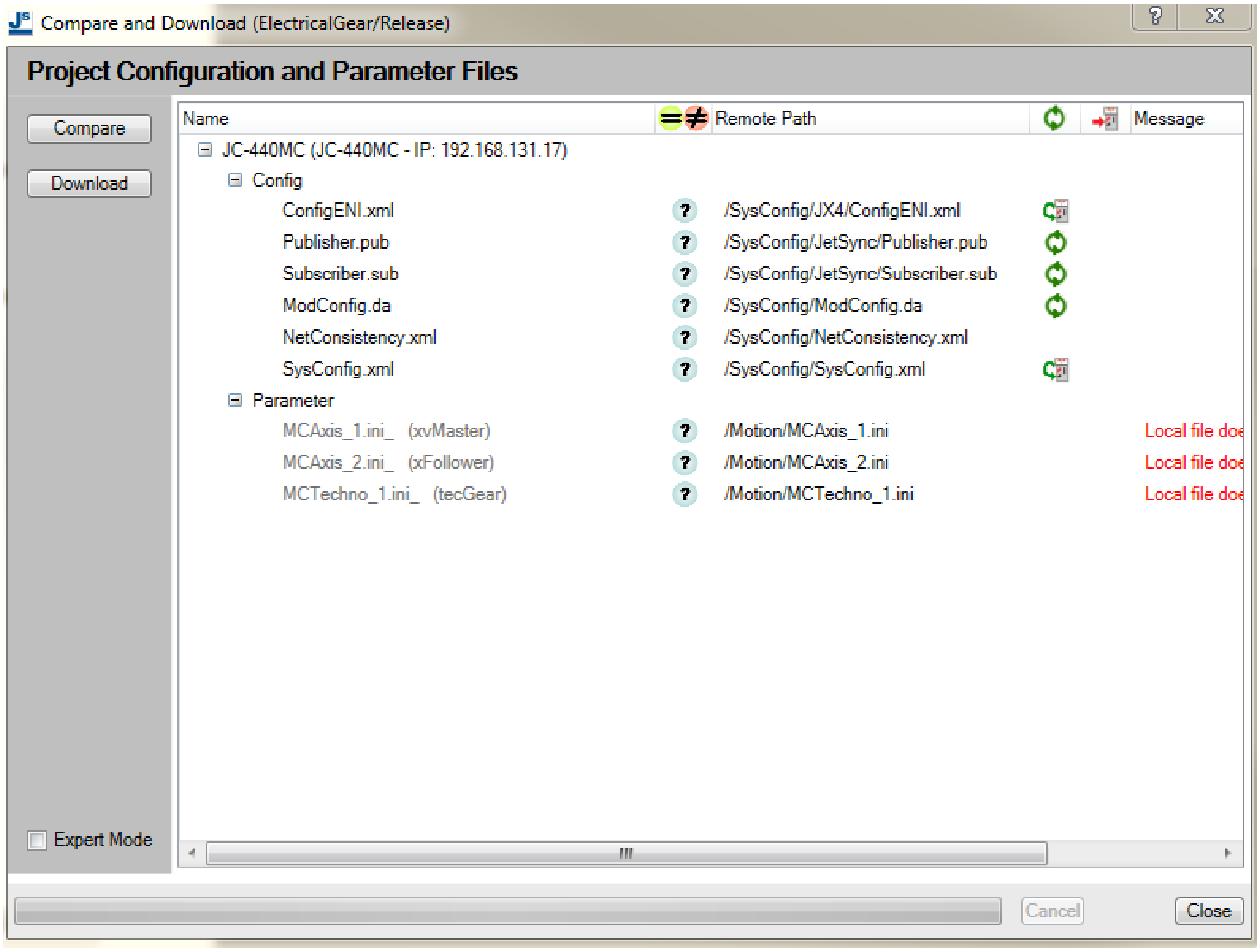
2.3 Downloading the MC files to the controller

The new motion control configuration must be made known to the controller by creating and downloading the necessary motion control files. After a reboot, the new configuration can be used by the controller.


ElectricalGear System
Release
Set as Active Configuration
Generate file for virtual setup
Compare and Download
Motion Control
MC-VirtualAxis (xvMaster)

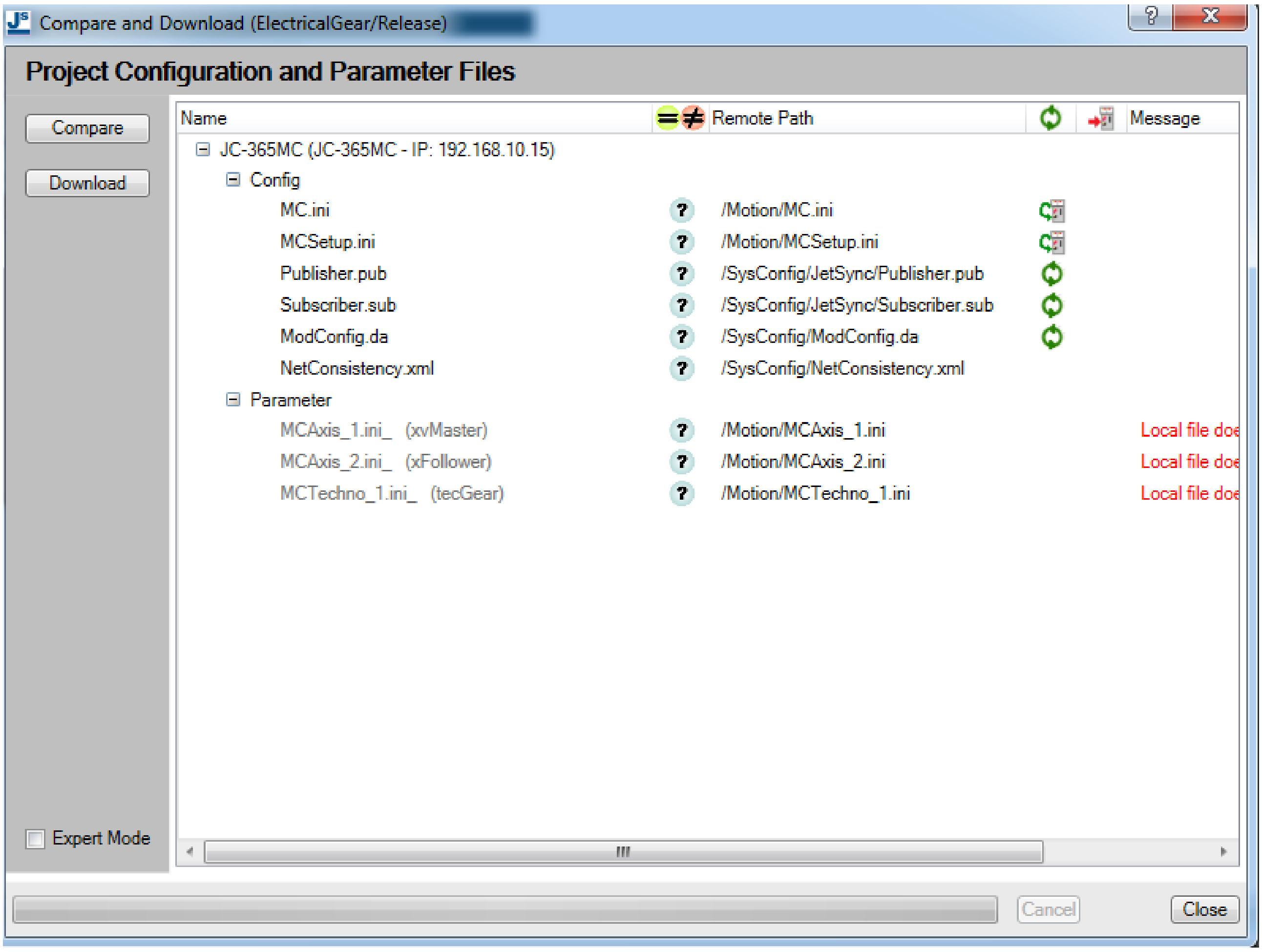
Select "*Compare and Download*" in the shortcut menu of the configuration folder (here e.g. "Release").

JC440MC


Compare and Download (ElectricalGear/Release)
Project Configuration and Parameter Files
Compare
Download
☐ Expert Mode
Cancel
Close

Name	=≠	Remote Path	↻	Message
⊟ JC-440MC (JC-440MC - IP: 192.168.131.17)				
⊟ Config				
ConfigENI.xml	?	/SysConfig/JX4/ConfigENI.xml	↻	
Publisher.pub	?	/SysConfig/JetSync/Publisher.pub	↻	
Subscriber.sub	?	/SysConfig/JetSync/Subscriber.sub	↻	
ModConfig.da	?	/SysConfig/ModConfig.da	↻	
NetConsistency.xml	?	/SysConfig/NetConsistency.xml		
SysConfig.xml	?	/SysConfig/SysConfig.xml	↻	
⊟ Parameter				
MCAxis_1.ini_ (xvMaster)	?	/Motion/MCAxis_1.ini		Local file doe
MCAxis_2.ini_ (xFollower)	?	/Motion/MCAxis_2.ini		Local file doe
MCTechno_1.ini_ (tecGear)	?	/Motion/MCTechno_1.ini		Local file doe

JC365MC:



Compare and Download (ElectricalGear/Release) [redacted]

Project Configuration and Parameter Files

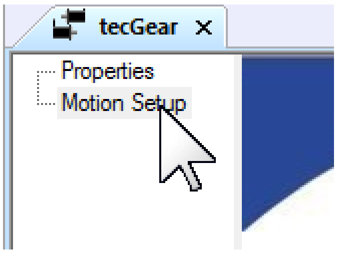
Name		Remote Path	⟳	Message
⊟ JC-365MC (JC-365MC - IP: 192.168.10.15)				
⊟ Config				
MC.ini	?	/Motion/MC.ini	⟳	
MCSetup.ini	?	/Motion/MCSetup.ini	⟳	
Publisher.pub	?	/SysConfig/JetSync/Publisher.pub	⟳	
Subscriber.sub	?	/SysConfig/JetSync/Subscriber.sub	⟳	
ModConfig.da	?	/SysConfig/ModConfig.da	⟳	
NetConsistency.xml	?	/SysConfig/NetConsistency.xml		
⊟ Parameter				
MCAxis_1.ini_ (xvMaster)	?	/Motion/MCAxis_1.ini		Local file doe
MCAxis_2.ini_ (xFollower)	?	/Motion/MCAxis_2.ini		Local file doe
MCTechno_1.ini_ (tecGear)	?	/Motion/MCTechno_1.ini		Local file doe

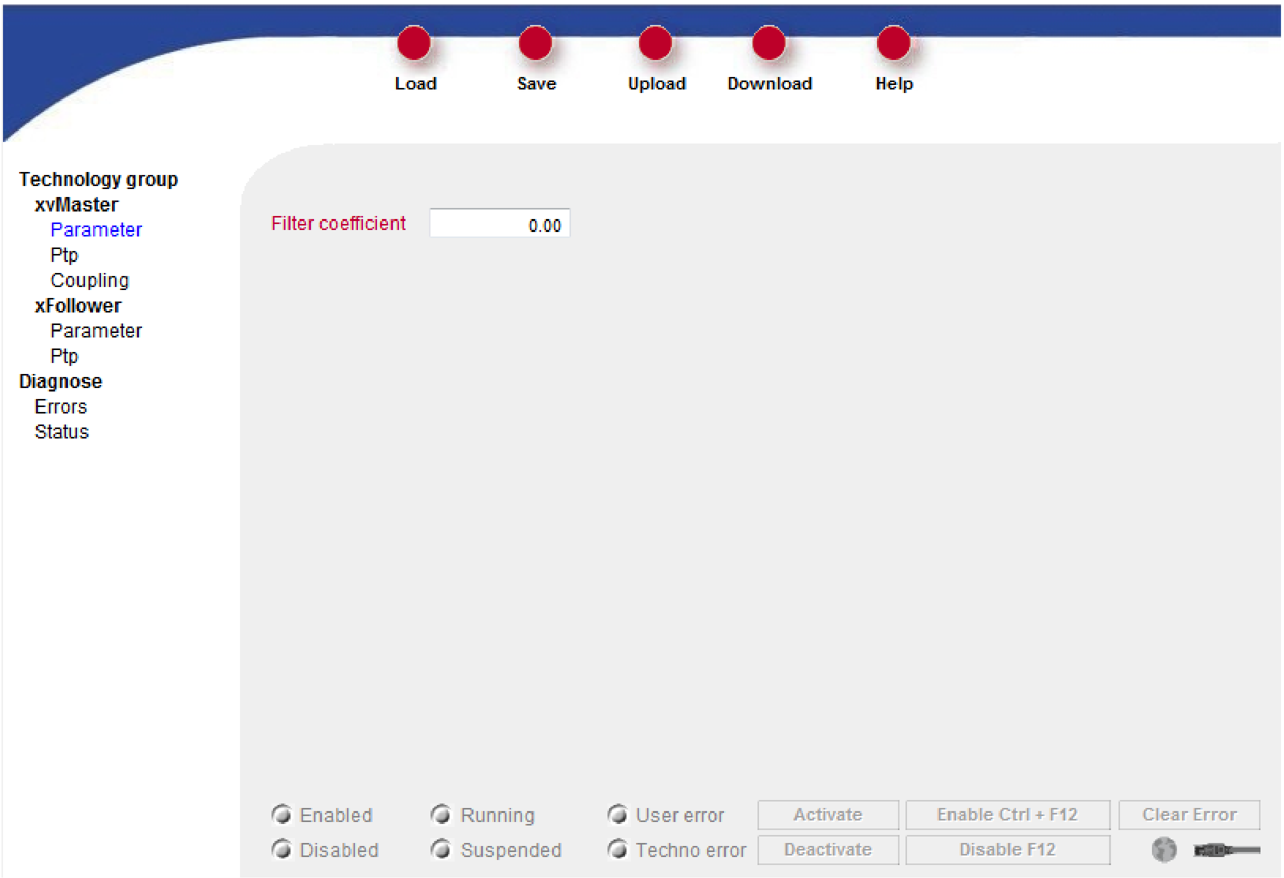
In the "*Compare and Download*" dialog, transfer the hardware configuration to the configured controller by clicking on the "*Download*" button.
If a reboot of the controller is necessary, a corresponding dialog is displayed where you can restart the controller.

3 Applying the Technology Group in Motion Setup

3.1 Operating the technology group

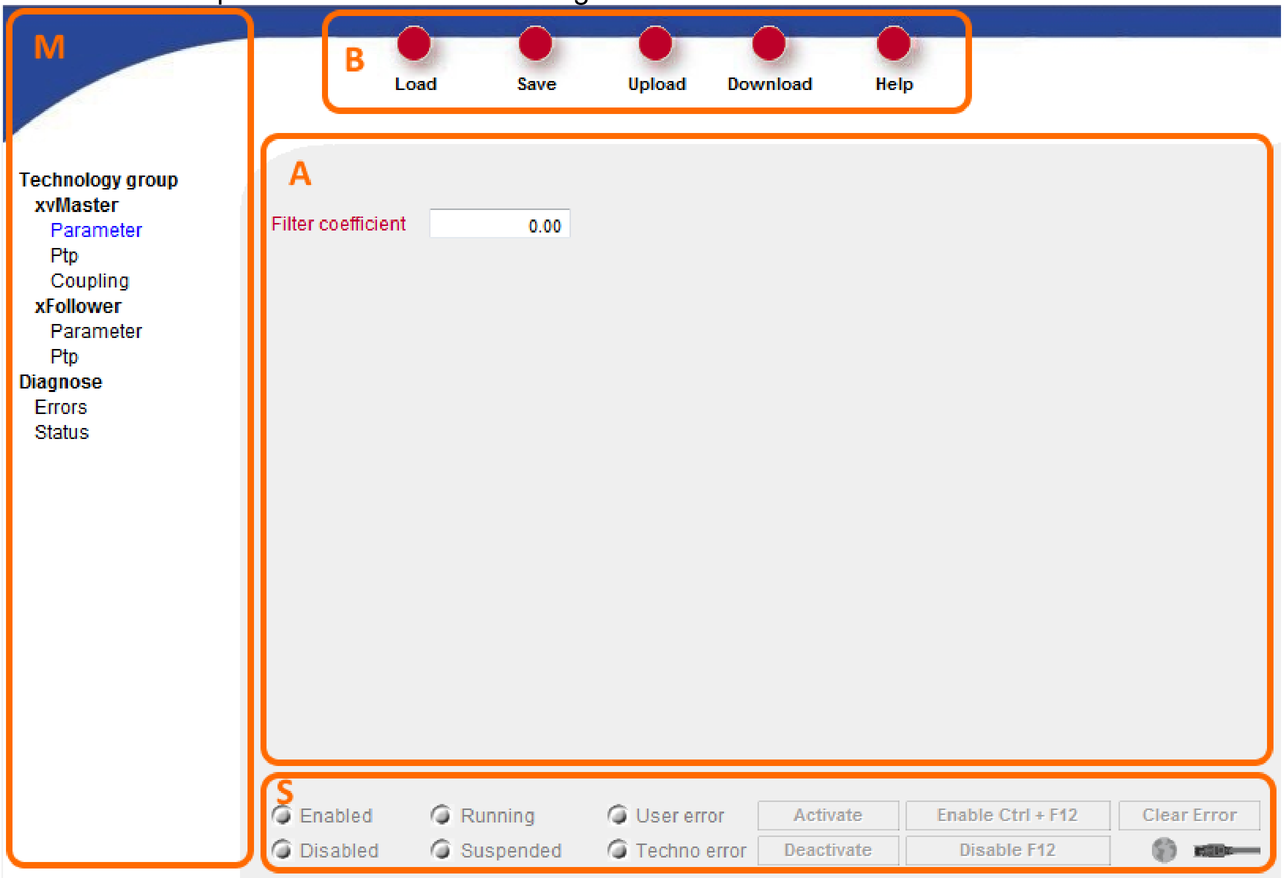
Open the Motion Setup dialog by double-clicking on the corresponding technology group. Then switch to "Motion Setup" in the configuration menu.





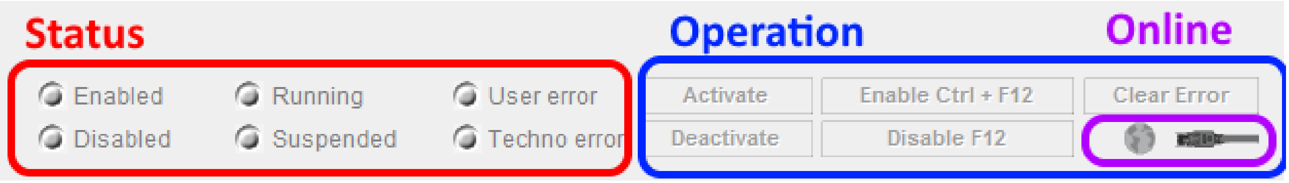
Filter coefficient 0.00

The Motion Setup is divided into the following areas:



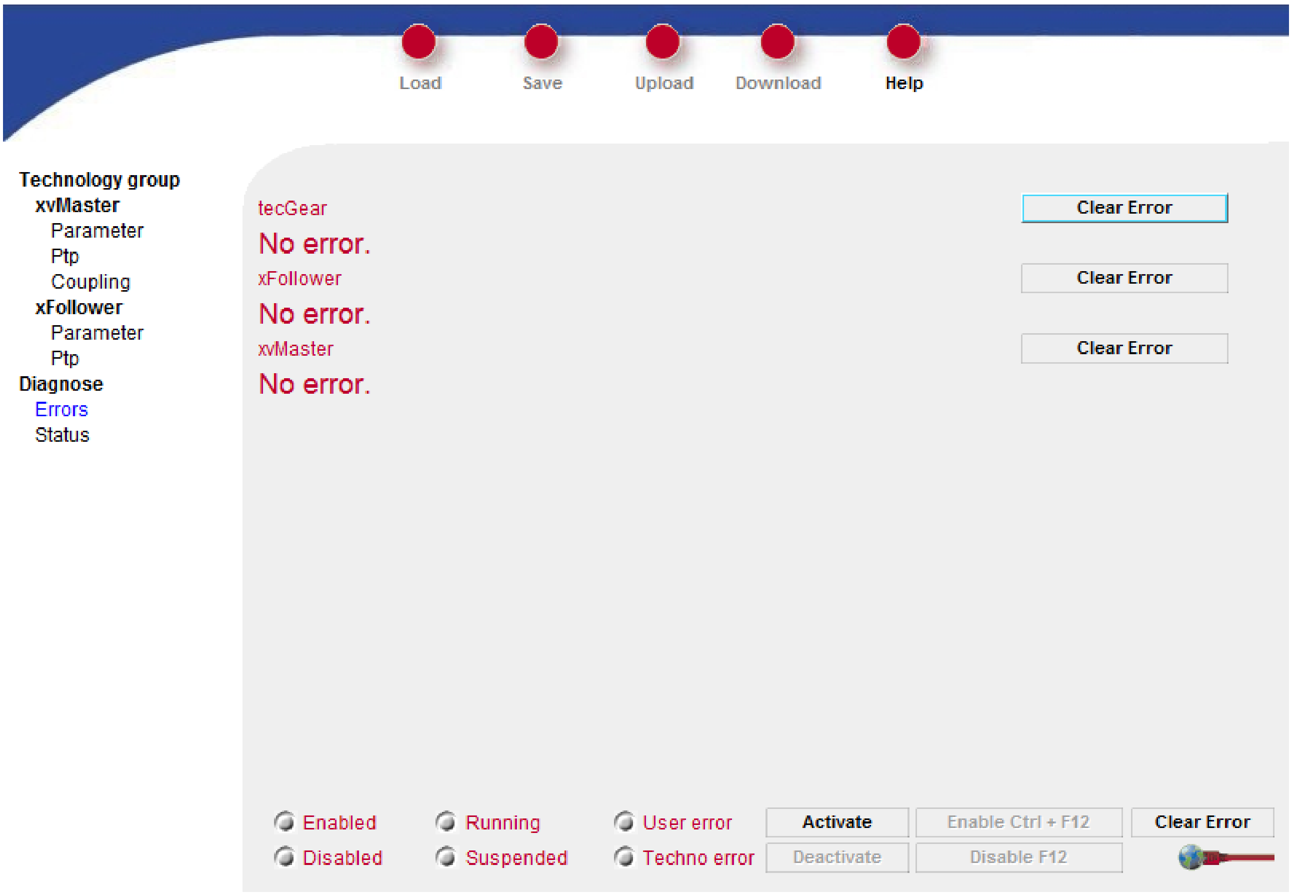
Filter coefficient 0.00

M: Menu - Selection of the individual commissioning pages
B: Control bar
A: Indication area
S: Status



- The elements allow the higher-level operation of the technology group
    - Activate/deactivate the technology group
    - Enable/disable (only if the group is active and error-free)
    - Clear error: Clears errors of the group and the axes involved in the active group
- Status indication
    - Enabled
    - Disabled
    - Running
    - Suspended
    - If the group is inactive, the statuses are grayed out
- Errors
    - User error
    - Techno error = Technology group errors
- Online
    - The online symbol indicates whether there is a connection with the controller.
    - Offline:
    - Online:

3.1.1 Error page - Overview



tecGear

No error.

xFollower

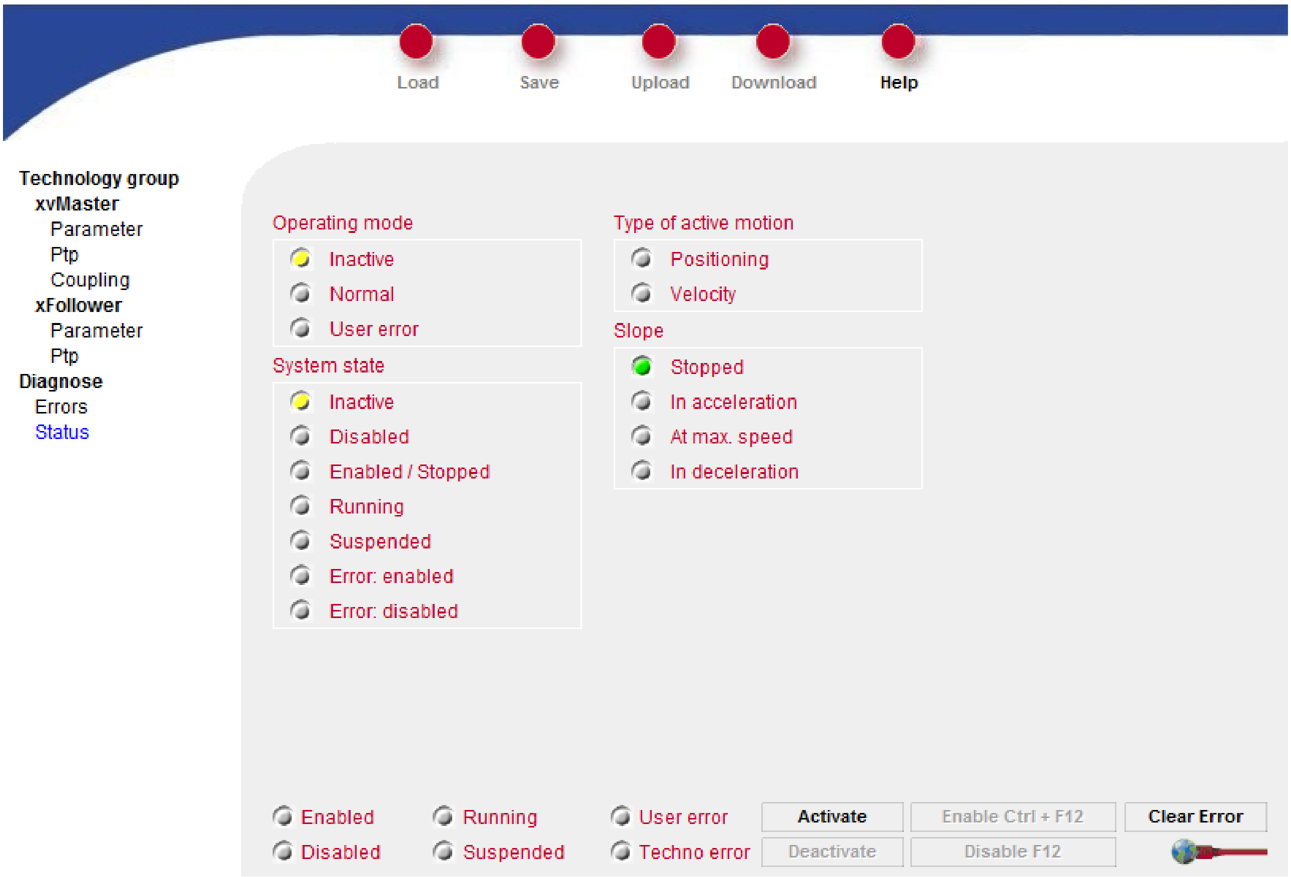
No error.

xvMaster

No error.

Here you can see existing errors of the group and its member axes. Apart from the option of clearing all errors with the "Clear Error" button in the control bar, errors can be handled individually here.

3.1.2 Status page - Overview



The technology group displays the status for operating mode, operating state, movement type and ramp status (= Slope).
Movement type and ramp status refer to the movement of the master axis.

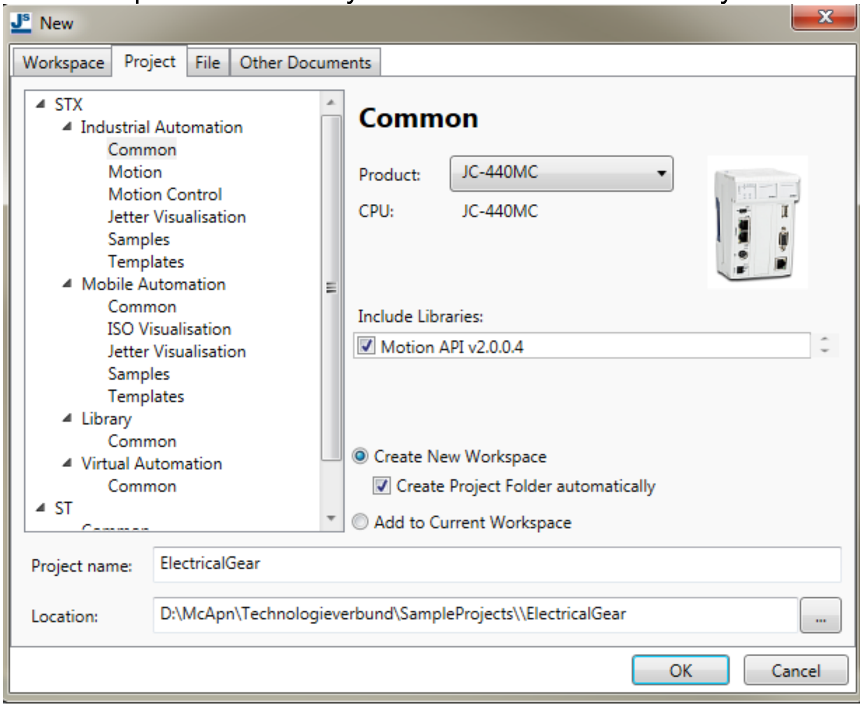
3.1.3 Other windows

Further windows for setting up and testing the technology group are explained in the thematically related Application Notes.

4 Applying the Technology Group in the Application Program

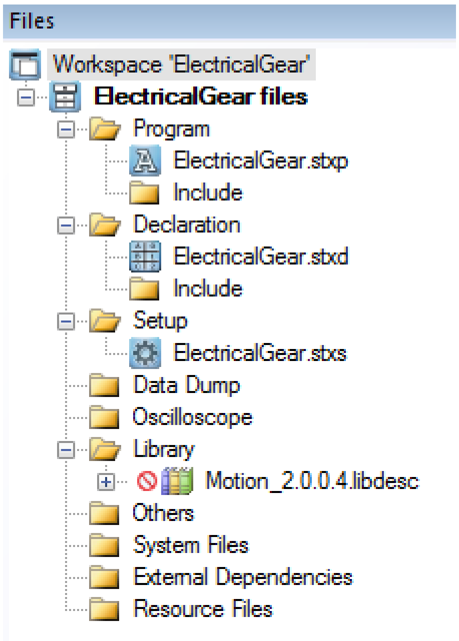
4.1 Integrating the Motion API library into a project

The first option is to directly select the Motion API library when creating a new project.

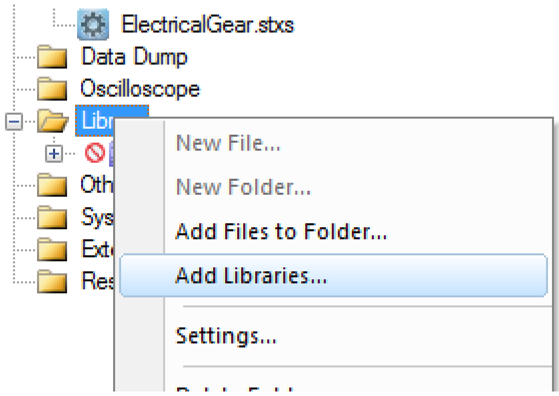


Product: JC-440MC
CPU: JC-440MC
Include Libraries:
☑ Motion API v2.0.0.4
● Create New Workspace
☑ Create Project Folder automatically
○ Add to Current Workspace
Project name: ElectricalGear
Location: D:\McApn\Technologieverbund\SampleProjects\\ElectricalGear

The library is then directly integrated into the project tree and included in the STX project.



By creating the hardware configuration and using the Motion API, a *"HardwareConfig.stxp*" is automatically created which is also available to the current project through the *"Include*" settings in the project settings.
In this program file, the motion objects are instantiated and the lists of the MC Manager are created.



In this view, libraries can also be changed and included via the shortcut menu of the “Library” folder.
Only one Motion API library can be included at a time. If you want to switch libraries, the existing link must first be deleted and then the desired library added. Deleting the library in this view does not delete the files or libraries in the file system!

This file is created during compilation. In order for the motion objects in the program to be used with the help of IntelliSense in the program, the program must be compiled once.
For a new project without existing code, the following short routine should be created to compile without errors:

```
Task tMain autorun
End_Task;
```

The motion objects are then entered in the IntelliSense which will support you during program creation.

4.2 Rebooting the program

When starting a program, we distinguish between two scenarios:

- Booting the controller
- Restarting the application by download or directly by JetSym

4.2.1 Rebooting the controller

When rebooting the controller, all components of the operating system of the controller are restarted. This also includes the Motion Control kernel. Based on the MC.ini and MCSetup.ini configuration files, the Motion Control kernel creates its motion objects. It also connects to and synchronizes the connected servo amplifiers. The configuration files associated with the Motion Control objects are then loaded so that control parameters, transmission ratios, etc. are available when the application is started.
Normally, all Motion Control objects are initialized, synchronized and error-free after completion of the boot phase. Axis groups are inactive while real, virtual, and simulation axes are in the operating state "locked". This is the default initial state from which the program proceeds.

Special cases:

- Boot error of the Motion Control kernel
  - Hardware configuration in MCSetup.ini does not match with actual hardware, e.g. if the number of specified axes does not match with connected drives or if the IP configuration of the servo amplifier is incorrect.
  - Synchronization failed e.g. due to wrong operating system of the interface cards in the servo amplifiers.
- Servo amplifier errors
  - Common errors:

- Encoder error (JM-200): If an encoder system other than resolver or HIPERFACE is used, this is not automatically detected by the JM-200 and it initially indicates an encoder error. This error can be cleared by "Clear Error", since the correct encoder should already be entered in the MCAxisXX.ini.
- Synchronization error F43 during Ethernet synchronization: Mostly generated by rebooting the controller only without simultaneously restarting the JM-200. When the controller is rebooted, the JM-200 detects a loss of synchronization and signals error F43. This error can be cleared by "Clear Error".

When starting the program, the above exceptions should be taken into account in order to reach the standardized initial situation on the one hand and to inform the user in case of boot errors on the other hand so that he can take appropriate actions.

4.2.2 Restarting the application

It is to be noted, that when the application is restarted, the controller will load the last available motion control objects as per the configuration files. Or be it through a previously run application program or through the use of Motion Control objects by Motion Setup.

This should be taken into account in standard applications. Since the actual application should start from a default initial state, the Motion Control objects should be transferred to this state.
This includes clearing errors, locking drives and deactivating groups.

Resetting the Motion Control objects

A procedure for resetting the Motion Control objects works both after booting the controller and when restarting the program.
The structure of the motion control results in a general sequence in which this can happen.

- Resetting the groups:
  - Disable
  - Clear error
  - Deactivate
- Resetting the axes
  - Disable
  - Clear error

It is recommended to restart the program or reboot the controller when the machine is stopped.
In the following suggestions, the motion is locked using an emergency stop ramp which results in a hard stop. If running axes are to be stopped in a targeted manner, the procedure must be adapted accordingly. In the simplest case, a targeted "MoveHalt" is sufficient before locking, but this approach quickly achieves a certain complexity due to the various requirements of the applications which cannot be generally described here.

It is also often desired that parameters that have been changed during runtime or by Motion Setup be reset again. To do this, load the parameters after clearing any errors.

4.2.2.1 Restarting the application - Suggestion 1:

```
//Resetting all technology groups
if not tecGear.State.IsInactive then
  if not tecGear.State.IsDisabled or not tecGear.State.IsDisabledError then
    tecGear.State.Transitions.Disable(MCStateTransitionDisableModes.VelocityControlSlope);
    when tecGear.State.IsDisabled or tecGear.State.IsDisabledError continue;
  end_if;

  tecGear.Diagnostics.ClearErrors();
  when tecGear.State.IsDisabled continue;

  tecGear.Deactivate();
  when tecGear.State.IsInactive continue;
end_if;

//Resetting all axes
if not xvMaster.State.IsDisabled or not xvMaster.State.IsDisabledError then
   xvMaster.Power.Quickstop();
when xvMaster.State.IsDisabled or tecGear.State.IsDisabledError continue;
end_if;

xvMaster.Diagnostics.ClearErrors();
when xvMaster.State.IsDisabled continue;

//Optional
xvMaster.Settings.LoadFromFile();
when xvMaster.State.IsDisabled continue;

if not xFollower.State.IsDisabled or not xFollower.State.IsDisabledError then
   xFollower.Power.Quickstop();
when xFollower.State.IsDisabled or xFollower.State.IsDisabledError continue;
end_if;

xFollower.Diagnostics.ClearErrors();
when xFollower.State.IsDisabled continue;

//Optional
xFollower.Settings.LoadFromFile();
when xFollower.State.IsDisabled continue;
```

In this example, the reset routines are executed separately for each Motion Control object created.

See also sample project:

- TechnologyStartUp1

4.2.2.2 Restarting the application - Suggestion 2:

```
function TechnoResetAll()
var
  nListIndex:  int;
end_var;

  for nListIndex := 0 to mcMgr.TechnoCount - 1 by 1 do
    if not mcMgr.TechnoList[nListIndex].State.IsInactive then
      if not mcMgr.TechnoList[nListIndex].State.IsDisabled or
         not mcMgr.TechnoList[nListIndex].State.IsDisabledError
      then
        mcMgr.TechnoList[nListIndex].State.Transitions.Disable(
        MCStateTransitionDisableModes.VelocityControlSlope);
        when mcMgr.TechnoList[nListIndex].State.IsDisabled or
             mcMgr.TechnoList[nListIndex].State.IsDisabledError continue;
      end_if;
      mcMgr.TechnoList[nListIndex].Diagnostics.ClearErrors();
      when mcMgr.TechnoList[nListIndex].State.IsDisabled continue;
      mcMgr.TechnoList[nListIndex].Deactivate();
      when mcMgr.TechnoList[nListIndex].State.IsInactive continue;
    end_if;
  end_for;
end_function;


function AxisResetAll()
var
  nListIndex:  int;
end_var;

  for nListIndex := 0 to mcMgr.AxisCount - 1 by 1 do
    mcMgr.AxisList[nListIndex].State.Transitions.Disable(
      MCStateTransitionDisableModes.VelocityControlSlope);
    when mcMgr.AxisList[nListIndex].State.IsDisabled or
         mcMgr.AxisList[nListIndex].State.IsDisabledError continue;
    mcMgr.AxisList[nListIndex].Diagnostics.ClearErrors();
    when mcMgr.AxisList[nListIndex].State.IsDisabled continue;
//Optional
    mcMgr.AxisList[nListIndex].Settings.LoadFromFile();
    when mcMgr.AxisList[nListIndex].State.IsDisabled continue;

  end_for;
end_function;
```

In this example, the reset routines are executed separately for each Motion Control object created. You can include these functions as standard functions in other projects.

See also sample project:

- TechnologyStartUp2

4.3 Activating the technology group

In the default initial state, groups are inactive. The operating state is therefore "Inactive" which can be queried using the "tecGroup.State.IsInactive" command.

The technology group is activated using the "tecGroup.Activate" command. Depending on the previous state of the member axes, successful activation can be queried before activation with "Disabled" or "Enabled" as a corresponding change of state.

The group is assigned the operating state "Enabled" or "Disabled" according to the states of the member axes before activation. If not all stateful member axes have the same status, an error is displayed when activating.

Stateful axes are those that can assume a state other than "Inactive". These are virtual, real or simulation axes.

External axes or shadow axes always have the "Inactive" state and are not taken into account.

Note:

A group can only be activated if the stateful member axes are in the "Enabled" or "Disabled" state. If, for example, at least one stateful member axis is in the "Running" state, activation is not possible.

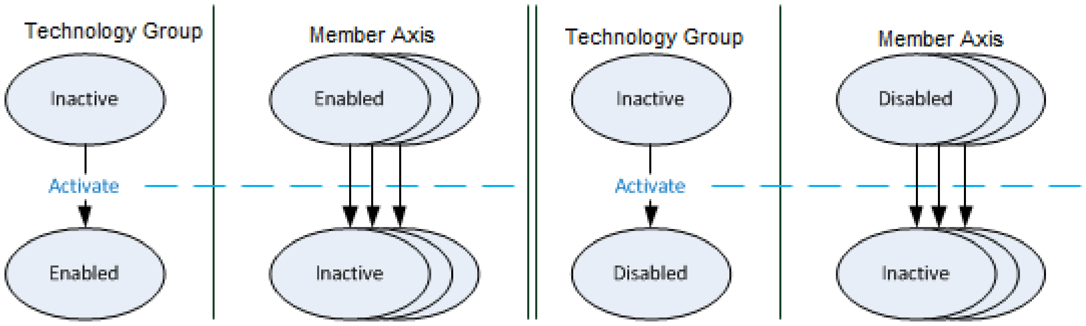
Example:

```
xvMaster.Power.Enable();
xFollower.Power.Enable();
when xvMaster.State.IsEnabled continue;
when xFollower.State.IsEnabled continue;

// Activating the group
tecGroup.Activate();
// Waiting for activation to succeed
when tecGroup.State.IsEnabled continue;
```

For simplicity, a successful activation is assumed in the process. The handling of errors can be performed in a separate task.
A typical error is the activation of the group with different operating states of the stateful member axes.

After the group has been activated, the operating state of the stateful member axes changes to "Inactive".



State transitions when the technology group is activated

4.4 Deactivating the technology group

The technology group is deactivated using the "tecGroup.Deactivate" command. The successful deactivation can be checked by querying for "Inactive". Alternatively, it is sufficient to query the operating status of the stateful member axes for "Enabled" or "Disabled".
The stateful member axes change from "Inactive" to "Enabled" if the group is in the "Enabled" state before deactivation.
The stateful member axes change from "Inactive" to "Disabled" if the group is in the "Disabled" state before deactivation.

Note: A group can only be deactivated if it is in the "Enabled" or "Disabled" state.

Example "Disabled":

```
when tecGroup.State.IsDisabled continue;
// Deactivating the group
tecGroup.Deactivate();
// Waiting for deactivation to succeed
when tecGroup.State.IsInactive continue;
```

Example "Enabled":

```
when tecGroup.State.IsEnabled continue;
// Deactivating the group
tecGroup.Deactivate();
// Waiting for deactivation to succeed
when xvMaster.State.IsEnabled continue;
when xFollower.State.IsEnabled continue;
```

After the group has been deactivated, its operating state changes to "Inactive".

State transitions when the technology group is deactivated:

See also sample project:

- TechnologyActivateDeactivate

4.5 Positioning

When positioning an axis, you must consider whether it is active as a single axis or as a member axis in the group.

For a single axis, the positioning functions are invoked directly on the axis object.

Example:

```
xvMaster.MovePtp.Start(AxisPositioningModes.AbsModuloAuto, 100.0, 200.0, 1000.0, 1000.0, 1.0);
xvMaster.MoveVelocity.Start(Directions.Positive, 100.0, 200.0, 300.0);
xvMaster.MoveHalt.Start(MCAxisHaltModes.Normal, 1000.0);
xvMaster.MoveHome.Start();
xvMaster.MoveHome.SetReference();
```

If the group is active, the positioning functions are invoked via the group:

Example:

```
tecGear.MovePtp.Start(xvMaster, MCTechnoPositioningModes.AbsModuloAuto, 100.0, 200.0, 1000.0,
1000.0, 1.0);
tecGear.MoveVelocity.Start(xvMaster, Directions.Positive, 100.0, 200.0, 300.0);
tecGear.MoveHalt.Start(xvMaster, MCTechnoHaltModes.Normal, 1000.0);
```

If positioning functions are sent to the single axis, although it is active in a group, the error "10 - Object inactive" is generated which indicates that an inactive object has been accessed here.

See also sample projects:

- TechnologyMoveInGroup
- TechnologyMoveNoGroup

4.5.1 MovePtp

Single axis:

There are two ways to query whether a positioning is finished:

- Operating state query: AxisName.State.IsEnabled
    - When positioning is executed, the operating state of the axis changes to "Running" which changes back to "Enabled" when it is finished.
- Ramp state query

- When positioning is executed, the ramp states change from "Accelerating", "ConstantSpeed", "Decelerating" to "Stopped" signaling the axis has stopped.
- Accelerating: AxisName.Mechanism.Slope.IsAccelerating
- Constant speed: AxisName.Mechanism.Slope. IsAtConstantSpeed
- Decelerating: AxisName.Mechanism.Slope. IsDecelerating
- Stopped: AxisName.Mechanism.Slope. IsStopped

These queries all refer to the setpoint of the axis.
Whether the actual position has also reached the target window should also be checked using the Axis.IsInTargetWindow command.
An axis that has been stopped prematurely, e.g. by a MoveHalt, changes to the "Enabled" operating state and the ramp status "Stopped", but may not have reached the target position!

Axes involved in a group:
When querying whether a positioning has been completed, only the ramp status and, if applicable, the target window of the respective axis should be taken into account. The operating state of the group is the sum of the operating states of the individual axes. As soon as a member of the group moves, the operating state changes to "Running" and only if all members are standing still, the state "Enabled" is reached. It may therefore be ambiguous, depending on the application.

Ramp status of the master axis:
The group itself also has a ramp status. This is the ramp status of the master axis! For this reason, the ramp status query must also take this into account when positioning the master axis. The ramp status of the master axis as a single axis always remains "Stopped" during the active grouping and can, therefore, not be used.

Example:

```
// Master axis
tecGear.MovePtp.Start(xvMaster, MCTechnoPositioningModes.AbsModuloAuto, 100.0, 200.0, 1000.0,
1000.0, 1.0);
when tecGear.Mechanism.Slope.IsStopped and xvMaster.IsInTargetWindow continue;

// Follower axis
tecGear.MovePtp.Start(xFollower, MCTechnoPositioningModes.AbsModuloAuto, 100.0, 200.0, 1000.0,
1000.0, 1.0);
when xFollower.Mechanism.Slope.IsStopped and xFollower.IsInTargetWindow continue;
```

4.5.2 MoveVelocity

Single axis:
To query whether the endless positioning is started, you can check for the operating state "Running". The axis changes to this state as soon as it starts to accelerate.

To query the phases of an endless positioning the ramp status is checked:

- AxisName.Mechanism.Slope.IsAccelerating
- AxisName.Mechanism.Slope.IsAtConstantSpeed

Example:

```
xvMaster.MoveVelocity.Start(Directions.Positive, 100.0, 200.0, 300.0);
when xvMaster.Mechanism.Slope.IsAtConstantSpeed continue;
```

Axes involved in a group:
As with MovePtp, the ramp status should be used here as with the single axis.

The operating state of the group is the sum of the operating states of the individual axes. As soon as a member of the group moves, the operating state changes to "Running" and only if all members are standing still, the state "Enabled" is reached. The state may therefore be ambiguous, depending on the application.

Ramp status of the master axis:
The group itself also has a ramp status. This is the ramp status of the master axis! For this reason, the ramp status query must also take this into account when positioning the master axis. The ramp status of the master axis as a single axis always remains "Stopped" during the active grouping and can, therefore, not be used.

Example:

```
// Master axis
tecGear.MoveVelocity.Start(xvMaster, Directions.Positive, 100.0, 200.0, 300.0);
when tecGear.Mechanism.Slope.IsAtConstantSpeed continue;

// Follower axis
tecGear.MoveVelocity.Start(xFollower, Directions.Positive, 100.0, 200.0, 300.0);
when xFollower.Mechanism.Slope.IsAtConstantSpeed continue;
```

4.5.3 MoveHalt

Single axis:
There are two ways to query whether stopping is complete:

- Operating state query: AxisName.State.IsEnabled
    - When positioning is executed, the operating state of the axis changes to "Running" which changes back to "Enabled" when it is finished.
- Ramp state query
    - In stopping, the ramp states change from "Accelerating", "ConstantSpeed", "Decelerating" to "Stopped" when the axis is stopped.
    - Accelerating: AxisName.Mechanism.Slope.IsAccelerating
    - Constant speed: AxisName.Mechanism.Slope. IsAtConstantSpeed
    - Decelerating: AxisName.Mechanism.Slope.IsDecelerating
    - Stopped: Axis.Mechanism.Slope.IsStopped

Example:

```
xvMaster.MoveHalt.Start(MCAxisHaltModes.Normal, 1000.0);
when xvMaster.Mechanism.Slope.IsStopped continue;
```

Axes involved in a group:
Only the ramp state should be taken into account when querying whether stopping is complete.

The operating state of the group is the sum of the operating states of the individual axes. As soon as a member of the group moves, the operating state changes to "Running" and only if all members are standing still, the state "Enabled" is reached. The state may therefore be ambiguous, depending on the application.

Ramp status of the master axis:
The group itself also has a ramp status. This is the ramp status of the master axis! For this reason, the ramp status query must also take this into account when positioning the master axis. The ramp status of the master axis as a single axis always remains "Stopped" during the active grouping and can, therefore, not be used.

Example:

```
// Master axis
tecGear.MoveHalt.Start(xvMaster, MCTechnoHaltModes.Normal, 1000.0);
when tecGear.Mechanism.Slope.IsStopped continue;

// Follower axis
tecGear.MoveHalt.Start(xFollower, MCTechnoHaltModes.Normal, 1000.0);
when xFollower.Mechanism.Slope.IsStopped continue;
```

4.5.4 MoveHome

Single axis:
MoveHome.Start
The reference run is started as usual with MoveHome.Start.

The operating state and the reference status should be used when querying whether the reference run has been completed:

- Operating state query: AxisName.State.IsEnabled
    - When positioning is executed, the operating state of the axis changes to "Running" which changes back to "Enabled" when it is finished.
- Query of the reference run status
    - Bei der Referenzfahrt wechseln die Status von "ReferenceRunStarted", "ReferenceFound", "SwitchFound" zu "ReferenceSet".
    - AxisName.MoveHome.IsReferenceRunStarted
    - AxisName.MoveHome.IsSwitchFound: Switch found
    - AxisName.MoveHome.IsReferenceFound: Reference found
    - AxisName.MoveHome.IsReferenceSet: End of reference run

Example:

```
xFollower.MoveHome.Start();
when xFollower.State.IsEnabled and xFollower.MoveHome.IsReferenceSet continue;
```

MoveHome SetReference()
If only the reference is set, it is sufficient to query the reference status "ReferenceSet".

Example:

```
xvMaster.MoveHome.SetReference();
when xvMaster.MoveHome.IsReferenceSet continue;
```

Note:
For a detailed description of the reference run, see the JM-200 Servo Amplifier documentation.

Axis involved in a group:
Reference run and setting the reference are not available in the group.
Therefore, all group members that need referencing should be handled individually before activating the group.

4.6 Diagnostics

Hint:
For more information on diagnostics of errors, warnings and messages also read the Application Note 054 "Motion Control Basics".

Single axis

The diagnostics of a single axis may be accessed via the axis object's identifier. Querying and clearing errors are done via the available functions and properties.

Example:

```
// Is there an error on the axis?
if xFollower.Diagnostics.IsErrorPending then
        // Clearing the axis error
        xFollower.Diagnostics.ClearErrors();
        // Waiting until error is cleared
        when xFollower.State.IsDisabled continue;
end_if;
```

Technology group

In a technology group, errors are queried and cleared in the same way as for single axes.
If a member axis of a group has an error, e.g. a tracking error, this results in an error state of the technology group, too. If a technology group error is cleared, all errors of the member axes are also cleared.

Example:

```
// Is there an error in the technology group?
if tecGear.Diagnostics.IsErrorPending then
// Clearing the error of the technology group
        tecGear.Diagnostics.ClearErrors();
        // Waiting until error is cleared
        when tecGear.State.IsDisabled continue;
end_if;
```

If the error state of an individual member axis is of interest, you can check the single axis object. Clearing an error will reset the error of the respective individual axis. However, the error of the technology group must still be explicitly cleared.

Example:

```
// Is there an error in the technology group?
if tecGear.Diagnostics.IsErrorPending then
        // Is there an error in the follower axis?
        if xFollower.Diagnostics.IsErrorPending then
                // Clearing the error in the follower axis
                xFollower.Diagnostics.ClearErrors();
                // Waiting until error is cleared
                when not xFollower.Diagnostics.IsErrorPending continue;
        end_if;

// Clearing the error of the technology group
        tecGear.Diagnostics.ClearErrors();
        // Waiting until error is cleared
        when tecGear.State.IsDisabled continue;
end_if;
```

Note:

The examples assume that the errors can be cleared. In practice, however, it is also possible that errors are still pending and clearing of the errors fails. If, for example, a real encoder of an axis is defective, it remains defective even after the error has been cleared. The above queries would now hang on the wait condition.

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg,
Germany
www.jetter.de

E-mail info@jetter.de
Phone +49 7141 2550-0

We automate your success.