SIEMENS

SIMOTION

Motion Control Technology Objects Synchronous Operation, Cam

Function Manual

Preface

Fundamental safety instructions 1

Part I - Synchronous Operation 2

Part II - Distributed Synchronous Operation 3

Part III - Synchronous operation across multiple cycles 4

Part IV - Cam 5

Valid as of Version 5.5

04/2023
A5E33429453B

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ **DANGER**
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ **WARNING**
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ **CAUTION**
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ **WARNING**
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E33429453B
Ⓟ 02/2023 Subject to change

Copyright © Siemens AG 2023.
All rights reserved

Preface

Preface

This **document** is part of the **System and Function Descriptions documentation package**.

Scope

This manual is valid for the following versions:

- SIMOTION SCOUT V5.5
- SIMOTION Kernel V5.5, V5.4, V5.3, V5.2, V5.1, V4.5, V4.4, V4.3, V4.2, V4.1, V4.0
- SIMOTION technology packages Cam, Cam_ext/Path (as of Kernel V3.2) and TControl in the version for the respective kernel (including technology packages Gear, Position and Basic MC up to Kernel V3.0)

Chapters in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- Synchronous Operation (Page 13) (Part I)
  Function of the synchronous operation, i.e. the grouping of a master object and a following axis
- Distributed Synchronous Operation (Page 149) (Part II)
  Function of the distributed synchronous operation, i.e. synchronous operation across different controllers
- Synchronous Operation IPO - IPO_2 (Page 217) (Part III)
  Function of synchronous operation with master object and following axis in different interpolator cycle clocks (IPO or IPO_2)
- Cam (Page 223) (Part IV)
  Function of the Cam technology object
- Index
  Index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions

- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Table of contents

Preface **3**

1 Fundamental safety instructions **11**

1.1 General safety instructions 11
1.1.1 Malfunctions of the machine as a result of incorrect or changed parameter settings 11
1.2 Security information 11
1.3 Note regarding the general data protection regulation 12
1.4 Danger to life due to software manipulation when using removable storage media 12

2 Part I - Synchronous Operation **13**

2.1 Overview of synchronous operation 13
2.2 Fundamentals of Synchronous Operation 20
2.2.1 Gearing 20
2.2.2 Velocity gearing 25
2.2.3 Camming 26
2.2.4 Setpoint/actual value coupling 35
2.2.4.1 Actual value coupling with extrapolation 36
2.2.4.2 Actual value coupling with tolerance window 39
2.2.5 Synchronization 40
2.2.5.1 Synchronization criterion 42
2.2.5.2 Synchronization direction 46
2.2.5.3 Position of synchronization range relative to synchronization position 48
2.2.5.4 Synchronization via a specifiable master value distance 49
2.2.5.5 Synchronization profile 50
2.2.5.6 Settings for evaluation of the master value behavior during synchronization 55
2.2.5.7 Monitoring the synchronization 59
2.2.5.8 Display of the synchronous position 61
2.2.5.9 "Synchronous" status during synchronization 63
2.2.6 Desynchronization 64
2.2.6.1 Desynchronization - Overview 64
2.2.6.2 Desynchronization criterion/desynchronization position 64
2.2.6.3 Desynchronization over a specifiable master value distance 65
2.2.6.4 Desynchronization profile via specifiable dynamic response parameters 65
2.2.6.5 Position of synchronization range relative to desynchronization position 65
2.2.6.6 Replacement of an active synchronous operation 65
2.2.7 Dynamic response effect on slave values 66
2.2.8 Switching of the master value source 68
2.2.8.1 Switching over the master value source – Overview 68
2.2.8.2 Master value switchover without dynamic response 69
2.2.8.3 Master value switchover with dynamic response 69
2.2.8.4 Master value switchover with next synchronization (V4.1 and higher) 70
2.2.9 Superimposed synchronous operation 71
2.2.10 Synchronous operation monitoring 75
2.2.11 Simulation mode 78

2.2.12 Configuring units ... 79
2.2.13 Examples of synchronization operations as a function of the output position on the slave value side ... 81
2.2.13.1 Synchronization via a specifiable master value distance ... 81
2.2.13.2 Synchronization profile based on specifiable dynamic response parameters ... 83
2.2.14 Examples ... 87
2.2.14.1 Examples of typical synchronization operations ... 87
2.2.14.2 Example of offset and scale on the synchronous object ... 98
2.2.14.3 Example of applying offset as superimposition ... 101
2.2.15 Special actions ... 104
2.2.15.1 Redefining the axis position during active synchronous operation ... 104
2.2.15.2 Retaining a synchronous connection for _disableAxis ... 105
2.2.15.3 Substitution of velocity gearing with absolute synchronous operation ... 106
2.2.15.4 Motion with absolute synchronous operation ... 107
2.2.15.5 Canceling active and pending synchronous operations ... 107
2.2.15.6 Adapt the synchronization velocity to the master value velocity ... 107
2.2.15.7 Substitution of synchronous operation through _move, _pos, _stop with reset of an existing acceleration to zero ... 109

2.3 Synchronous Operation Configuration ... 110
2.3.1 Creating an axis with synchronous operation ... 111
2.3.2 Assigning master values and cams ... 112
2.3.3 Assigning parameters/defaults for synchronous operation ... 114
2.3.3.1 Gearing ... 116
2.3.3.2 Velocity gearing ... 117
2.3.3.3 Camming ... 118
2.3.3.4 Gearing synchronization ... 120
2.3.3.5 Synchronizing the gear ... 121
2.3.3.6 Position reference during synchronization ... 122
2.3.3.7 Desynchronizing the gear ... 123
2.3.3.8 Position reference during desynchronization ... 123
2.3.3.9 Camming synchronization ... 124
2.3.3.10 Cam synchronization ... 125
2.3.3.11 Cam desynchronization ... 126
2.3.3.12 Dynamic response ... 127
2.3.3.13 Master dynamic response ... 129
2.3.4 Set synchronization ... 130
2.3.5 Configuring synchronous operation monitoring ... 132

2.4 Synchronous Operation Programming/References ... 133
2.4.1 Overview of commands ... 133
2.4.1.1 Commands for reading out function values ... 135
2.4.1.2 Commands for command tracking ... 136
2.4.1.3 Commands for resetting states and errors ... 138
2.4.2 Command processing ... 138
2.4.2.1 Command execution ... 138
2.4.2.2 Command transition conditions ... 141
2.4.3 Error handling ... 143
2.4.3.1 Local alarm reaction on the synchronous object ... 143
2.4.3.2 Local alarm reaction of the following axis and its mutual effects with the synchronous object ... 143
2.4.3.3 Error handling in the user program ... 145
2.4.4 Menus ... 146

2.4.4.1 Synchronous Operation - Menu ... 146
2.4.4.2 Synchronous Operation - Context Menu ... 147

3 Part II - Distributed Synchronous Operation ... 149

3.1 Overview of distributed synchronous operation ... 149

3.2 Fundamentals of Distributed Synchronous Operation ... 150
3.2.1 Boundary Conditions ... 150
3.2.1.1 Rules for the communication/topology for distributed operation using PROFIBUS ... 150
3.2.1.2 Rules for the communication/topology for the distribution using PROFINET IO with IRT (V4.0 or later) ... 156
3.2.2 Compensations for distributed synchronous operation ... 157
3.2.2.1 Compensation on master value side by means of setpoint output delay ... 160
3.2.2.2 Compensation of the slave value side by means of master value extrapolation ... 162
3.2.2.3 Permissible combinations for cycle clock offset compensation in distributed synchronous operation ... 163
3.2.2.4 Cycle clock offset calculation using a command ... 164
3.2.3 Operating axes with distributed synchronous operation ... 164
3.2.3.1 Sign-of-life monitoring ... 164
3.2.3.2 Operating states ... 166

3.3 Distributed Synchronous Operation Configuration ... 168
3.3.1 Creating SIMOTION devices with SCOUT ... 169
3.3.2 Creating connections with HW Config ... 169
3.3.3 Creating synchronous operation connections with SCOUT ... 171
3.3.4 Generating a synchronous operation configuration ... 174
3.3.5 Possible error ... 174

3.4 Programming distributed synchronous operation ... 175
3.4.1 Synchronizing the interfaces ... 175
3.4.2 Synchronous commands ... 176

3.5 Configuring distributed synchronous operation across projects ... 177
3.5.1 Overview ... 177
3.5.2 PROFIBUS communication configuration ... 179
3.5.2.1 Communication via PROFIBUS DP ... 179
3.5.2.2 Creating and configuring a "synchronized axis" project ... 180
3.5.2.3 Creating and configuring a "master object" project ... 187
3.5.3 Communication configuration via PROFINET IO ... 190
3.5.3.1 Communication via PROFINET IO ... 190
3.5.3.2 Creating and configuring "Synchronous axis" project on SIMOTION D455-2 ... 191
3.5.3.3 Creating and configuring "Master Object" project on SIMOTION D455-2 ... 199
3.5.4 Proxy objects ... 205
3.5.4.1 Proxy object types ... 205
3.5.4.2 Creating proxy objects ... 205
3.5.4.3 Configuring proxy objects with SIMOTION scripting ... 209
3.5.5 Interconnection possibilities ... 211
3.5.6 Synchronizing the interface ... 213
3.5.7 Switching over to an external master value source ... 213

3.6 Configuration of distributed synchronous operation with different RT versions ... 213
3.6.1 Overview of distributed synchronous operation with different RT versions ... 213
3.6.2 Configuration of distributed synchronous operation with different RT versions across projects ... 214

3.6.3 Configuration of distributed synchronous operation with different RT versions in one project ... 214

4 Part III - Synchronous operation across multiple cycles ... 217

4.1 Overview of multiple cycle synchronous operation ... 217

4.2 Boundary conditions ... 219

4.3 Running synchronous operation across multiple cycles ... 220

4.4 Creating synchronous operation across multiple cycles in SCOUT ... 222

5 Part IV - Cam ... 223

5.1 Overview of cam ... 223

5.2 Fundamentals of Cam ... 224
5.2.1 Definition ... 224
5.2.2 Normalization ... 226
5.2.3 Scaling and offset ... 226
5.2.4 Interpolation ... 228
5.2.5 Inversion ... 233
5.2.6 Motion laws in accordance with VDI ... 234
5.2.6.1 Motion tasks ... 235
5.2.6.2 Defining a cam for a motion task using segments ... 237

5.3 Cam Configuration ... 237
5.3.1 Creating a cam ... 238
5.3.2 Exporting/importing cams ... 238
5.3.3 Defining and loading cams ... 239

5.4 Cam Programming/References ... 240
5.4.1 Commands for definition ... 240
5.4.2 Commands for reading out function values ... 242
5.4.3 Commands for resetting the cam and errors ... 243
5.4.4 Commands for command tracking ... 244
5.4.5 Programming and sequence model ... 245
5.4.6 Local alarm response ... 245

5.5 Graphic output ... 246
5.5.1 Reading out cams with CamEdit ... 246

Index ... 247

Fundamental safety instructions 1

1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

1.1.1 Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/cert).

1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Part I - Synchronous Operation

2

2.1 Overview of synchronous operation

This part describes the function of the **synchronous operation** technology. It introduces you to the setting and configuration functions, as well as providing information about the supplementary conditions and the operating characteristics of synchronous objects.

What is synchronous operation used for?

Synchronous operation functions are taking on an increasingly significant role in automation engineering. The progress in open-loop and closed-loop control engineering and the availability of increasingly more powerful systems mean that solely mechanical solutions are more and more frequently being replaced with "electronic" variants.

The synchronous operation functions of the SIMOTION technology provide the option of replacing rigid mechanical connections with "control engineering", thus producing more flexible, maintenance-friendly solutions.

What is synchronous operation in SIMOTION?

The synchronous operation functionality of axes is provided by the synchronous object. A leading object (master) generates a master value, which is processed by the synchronous object according to specific criteria (gear ratio, scaling, offset, cam) and assigned to the following axis (slave) as a reference variable.

Mechanical model

The mechanical model for a synchronous operation relationship is, for example, a gear with a drive wheel and an output wheel. The model for camming could be a cam gear with a mechanical cam and sampling mechanism. A coupling used for enabling and disabling the following motion on-the-fly is also used as a model.

Synchronous operation functions

The following synchronous operation functions can be implemented:

- With gearing (Page 20), a linear transmission function between a master value and a following axis can be achieved using control engineering, thus producing the same result as could be achieved mechanically using a gear. A gear ratio can be specified for use in linear mapping of the master axis position onto the following axis position.

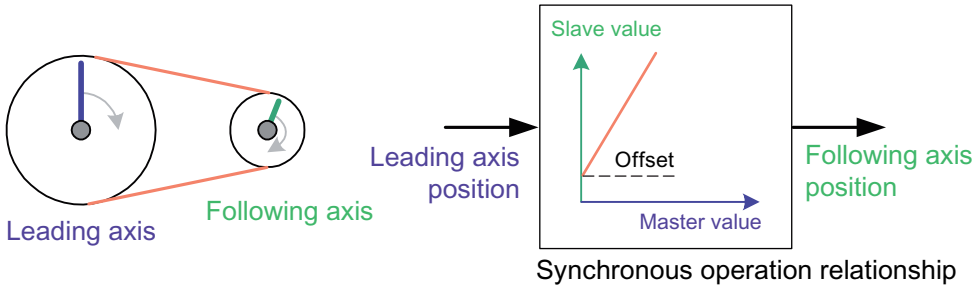


Figure 2-1 Gearing synchronous operation function (mechanical example)

- With velocity gearing (Page 25), a constant velocity coupling is implemented (as of V3.1).
- With camming (Page 26), a non-linear transmission function between a master value and following axis can be achieved. The slave value is generated from the master value using the transmission function defined in the cam. The cam is defined using interpolation points or mathematical functions and is interpolated between the specifications.

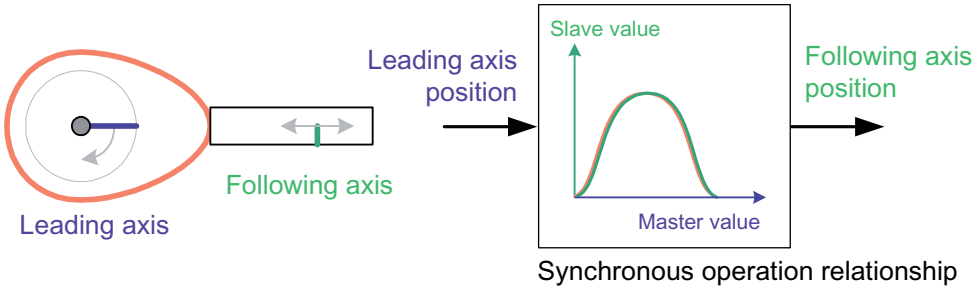


Figure 2-2 Camming synchronous operation function (mechanical example)

A synchronous operation sequence

Synchronous operation of a following axis to a master value using the SIMOTION synchronous operation functions is divided into three phases:

- Synchronization
- Synchronized traversing
- Desynchronization

Within these phases, there are several options for influencing the synchronous operation functions.

Synchronization/Desynchronization

The synchronous operation to the master value during synchronization or desynchronization can be defined differently depending on the application.

It is determined on the basis of:

- The synchronization criterion/synchronization position
- Synchronization direction
- The position of the synchronization range relative to the synchronization position
- Synchronization profile

See the section titled Synchronization (Page 40).

Objects

A *synchronous operation relationship* exists between the following objects:

- At least one *master object* (master)
  The master object is a technology object that provides motion information with a position (the motion slave value). This can be, for example, a positioning axis or an external encoder.
- At least one *synchronized axis*, comprising:
  - A *following axis* (slave)
  - One or two *synchronous objects*
  - Possibly one or more *cams*

A synchronous object is automatically created as a separate object in SIMOTION SCOUT when an axis with synchronous operation technology is created.

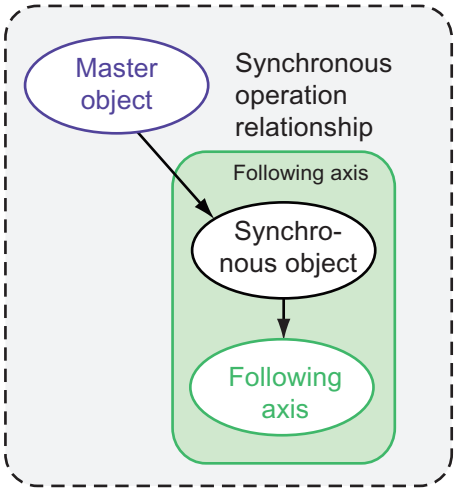


Figure 2-3 Objects in gearing

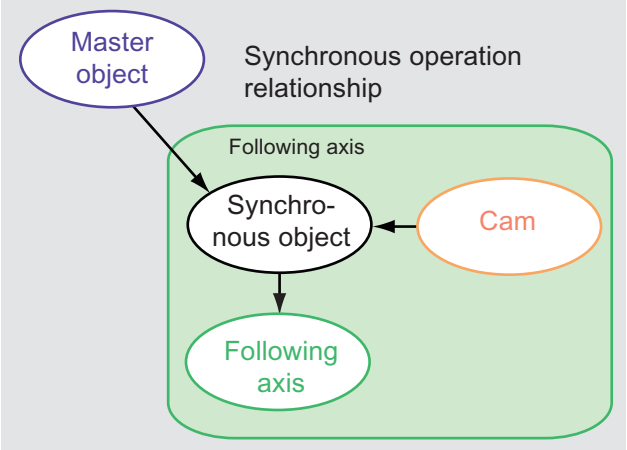


Figure 2-4 Objects in camming

Master values

The master value can be specified by the following technology objects:

- Axis
- External encoder

With restrictions (not for distributed synchronous operation or synchronous operation IPO-IPO_2), the following technology objects can also specify the master value:

- Fixed gear
- Addition object
- Formula object

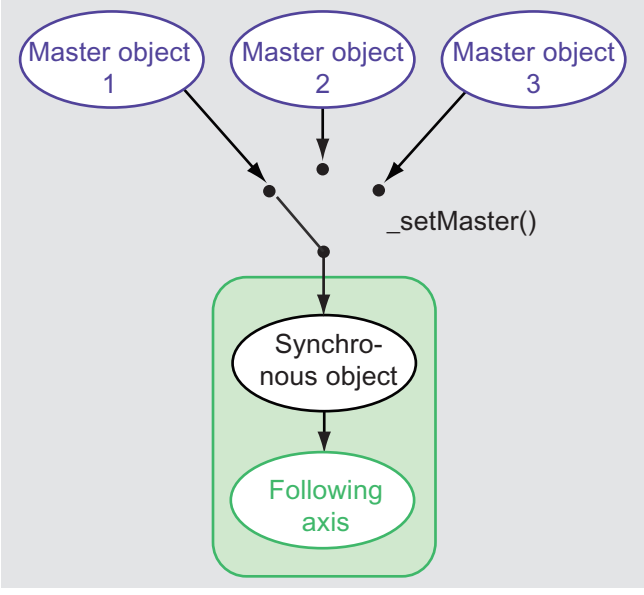


Figure 2-5 Example: Synchronous object with several master values

A following axis can be interconnected with more than one master value by means of the synchronous object. However, only one of these master values can be activated at any given

time. The process of switching over to a different master value is described in the section titled Switching over the master value source (Page 68).

When axes serve as the master value source, the setpoint coupling or the actual value coupling with extrapolation can be selected. When external encoders serve as the master value source, actual value coupling/actual value coupling with extrapolation (V3.0 and higher) can be selected. See the section titled Setpoint/actual value coupling (Page 35).

Note

A drive axis cannot be used as the master value source for a gearing.

Processing cycle clock of the synchronous object

The processing cycle clock of the synchronous object and the processing cycle clock of the synchronized axis must be identical.

Note

If you change the processing cycle clock of the synchronized axis using the configuration screen form, the processing cycle clock of the synchronous object is changed automatically. If you change the processing cycle clock of the synchronized axis or of the synchronous object using the expert list, the processing cycle clock of the synchronous object or synchronized axis is *not* changed.

Recursive synchronous operation interconnection

A recursive synchronous operation interconnection is present when, in a single synchronous operation relationship, a synchronized axis is interconnected directly or indirectly again as a master value via other technology objects. A synchronized axis cannot act as a following axis to a master value and as a master value for the same axis simultaneously. Recursive synchronous operation interconnections can result if, for example, a synchronous operation relationship is to be switched over in the event of an error. Also refer to the section titled Error handling in the user program (Page 145).

Units

The master and slave values are coupled without physical conversion *in the relevant parameterized units*. If, for example, the master axis is a linear axis and the following axis is a rotary axis, a length unit corresponds to an angular unit (for a 1:1 conversion ratio).

Modulo behavior

Different modulo ranges on the master value object and the following axis are taken into account on the synchronous object.

Camming with several cams

Several cams can be used in one camming operation. You can switch over to another cam dynamically using the **_enableCamming()** command in the user program.

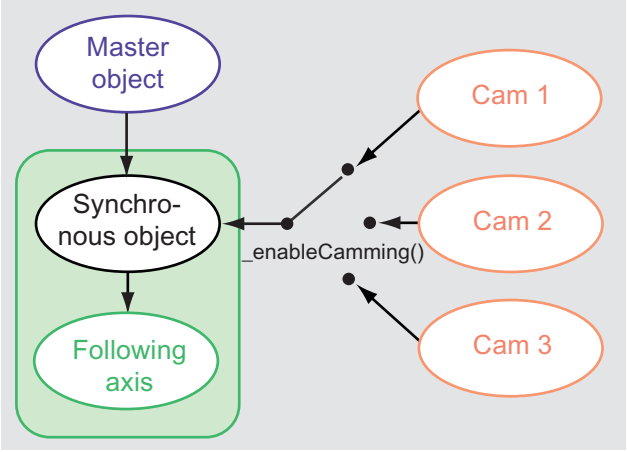


Figure 2-6 Example: Camming with several cams

Rules for interconnection

To recap, the following rules apply to synchronous operation:

- The synchronous object and following axis must be on the same runtime system (SIMOTION device).
- The master object and the following axis can be in different SIMOTION devices.
  In this case, we talk about distributed synchronous operation (Page 149).
- The master object and synchronized axis may operate in different IPO cycles
  (see Overview of synchronous operation IPO - IPO_2 (Page 217)).
- The synchronous object and the following axis are permanently assigned to each other during configuration.
- Up to two synchronous objects can be interconnected to one following axis.
- The master value object can be interconnected to several followings.
- The synchronous object can be interconnected to several master values and cams.
- A cam can be interconnected with several synchronous objects.

Superimposed synchronous operation

In superimposed synchronous operation, two synchronous objects can be connected to one following axis. The two synchronous operations superimpose one another (V3.0 and higher).

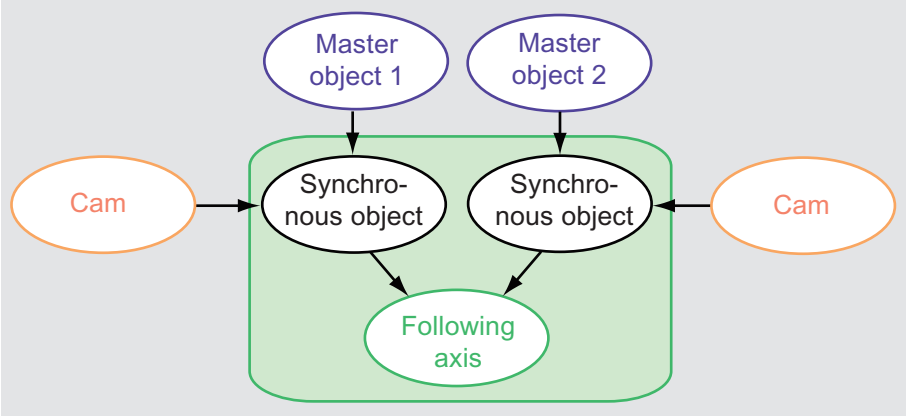


Figure 2-7 Example of superimposed synchronous operation

For additional information, see the section titled Superimposed synchronous operation (Page 71).

See also

Synchronous Operation Configuration (Page 110)

Synchronous Operation Programming/References (Page 133)

2.2 Fundamentals of Synchronous Operation

2.2.1 Gearing

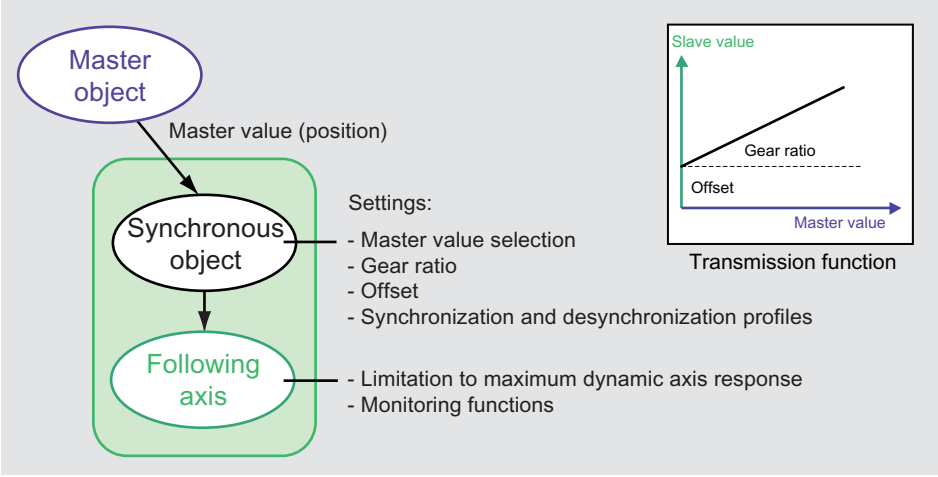


Figure 2-8 Gearing

Gearing is characterized by a linear transmission function between the master value source and the following axis/axes.

Slave value = Gear ratio x Master value + Offset

This gear ratio can be specified as the ratio of two decimal numbers (numerator/ denominator) or as a rational number. A zero point offset can be included in the calculation. Absolute or relative gearing can be set using the gearingType parameter of the **_enableGearing()** command.

Absolute gearing

With absolute gearing (**gearingType=ABSOLUTE**), the synchronous operation occurs absolutely relative to the zero point of the master value and slave value, taking into account the gear ratio.

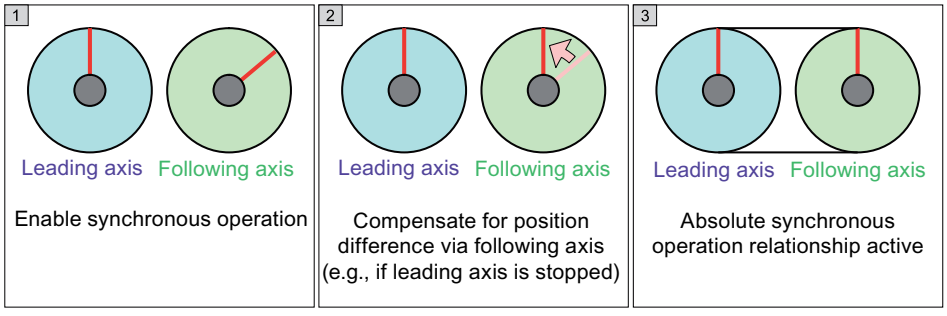


Figure 2-9 Sequence of the absolute gearing synchronization (simplified example)

A specified offset of the slave value is included. This offset is equal to zero, except when the synchronization criterion **ON_MASTER_AND_SLAVE_POSITION** or

IMMEDIATELY_AND_SLAVE_POSITION is set in the **syncPositionSlave** parameter, in which case an offset is specified.

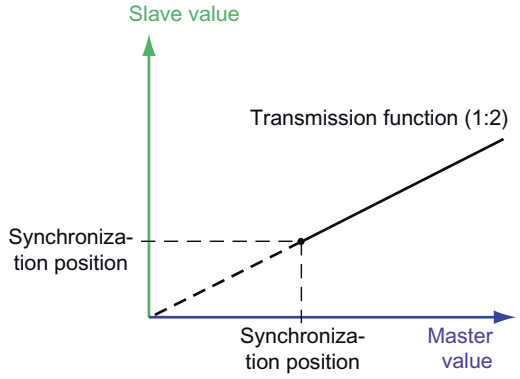


Figure 2-10 Absolute gearing without specification of the slave value position

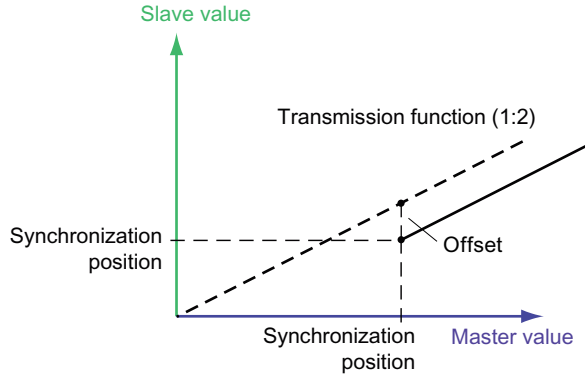


Figure 2-11 Absolute gearing with specification of the slave value position

Position differences on the slave value side are compensated for during synchronization. Modulo settings are taken into account.

Relative gearing

With relative gearing (**gearingType:=RELATIVE**), the synchronous operation occurs relative to the synchronization position on the master value and slave value sides.

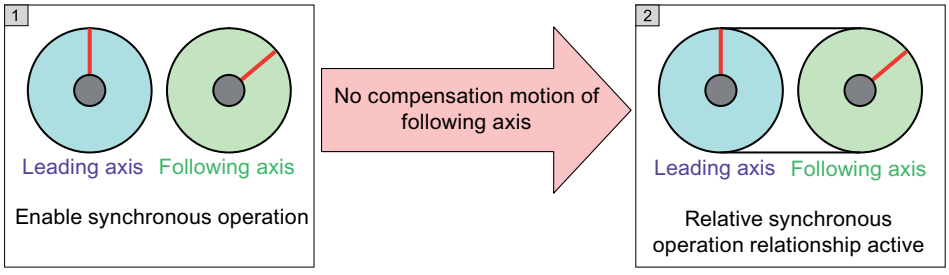


Figure 2-12 Sequence of the relative gearing synchronization (simplified example)

The offset is determined implicitly in the transmission function:

- If programmed without a specified offset: The offset is derived from the current position of the following axis at the start of synchronization and from an offset derived implicitly during synchronization if the axis is driven to a velocity (and acceleration) derived from the gear ratio.
- If programmed with a specified offset: Using the synchronization criterion setting **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION**, the offset is determined from the current following axis position at the start of synchronization and the offset programmed in **syncPositionSlave**.

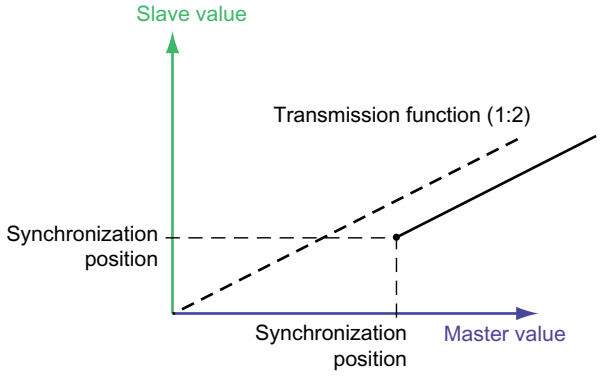


Figure 2-13 Relative gearing without offset

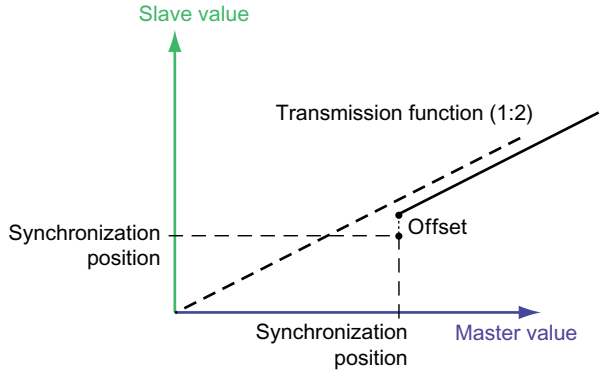


Figure 2-14 Relative gearing with offset

From the point at which the status becomes "synchronous", relative gearing is also in positional synchronism. In other words, the offset remains constant in the transmission function from this point onwards.

Gear ratio

The gear ratio is used to define the transmission function of the gearing between the master value and slave value. The gear ratio corresponds to the slope of the transmission function. It can entered as either a fraction or a floating-point number using the **gearingMode** parameter of the **_enableGearing()** command.

- As a fraction (**gearingMode:=GEARING_WITH_FRACTION**)
  The gear ratio is specified as a fraction (slave value difference/master value difference) using the following function parameters:
  - **gearingRatioType**: Type of gear ratio specification (directly or via replacement values)
  - **gearingNumerator**: Value for direct specification of the gear ratio numerator
  - **gearingDenominator**: Value for direct specification of the gear ratio denominator
- As a floating-point number (**gearingMode=GEARING_WITH_RATIO**)
  The gear ratio is specified as a floating-point number using the following function parameters:
  - **gearingRatioType**: Type of gear ratio specification (directly or via replacement values)
  - **gearingRatio**: Value for direct specification of the gear ratio as a floating-point number
    Disadvantage: Gear ratios such as 1/3 ≈ 0.333 are subject to rounding errors.

The long-term effect is to be taken into account with modulo axes. If master and slave axes are configured as modulo axes, to ensure the long-term stability of the gear, the gear ratio is preferably to be entered as a nominator/denominator ratio. If this is not possible, a LREAL value with corresponding decimal places should be used.

Direction of gearing

The gear ratio can be set in the same direction or in the opposite direction (corresponding to a negative gear ratio) using the **direction** parameter of the **_enableGearing** command.

- For **POSITIVE**, traversal is made in the same direction as the master values, this means that the axes run in the same direction.
- For **NEGATIVE**, traversal is made in the opposite to master values, this means that the axes run in the opposite direction.
- With **CURRENT**, the direction of the current slave values is retained; along with the direction of the master value; this results in coupling in the same or opposite direction, which is then maintained for the entire command execution time (that is, if the master value direction changes, then the slave direction changes as well).
- **REVERSE** means movement in the inverse direction of the slave values.

If the slave values are at a standstill at the point at which the command is activated, the following conversion is performed: **CURRENT** becomes **POSITIVE** and **REVERSE** becomes **NEGATIVE**.

Change in the offset

The **activationMode** parameter of the **_setGearingOffset()** command specifies when the offset takes effect (as of V3.1).

The changeover applies as follows:

- *For the next synchronous operation and all subsequent synchronous operations* if **DEFAULT_VALUE** is set
- *For the current synchronous operation only* if **ACTUAL_VALUE** is set
- *For the current synchronous operation and all subsequent synchronous operations* if **ACTUAL_AND_DEFAULT_VALUE** is set

Note the following:

- If the synchronization operation of the **_enableGearing()** command is not yet active, the current offset is carried out without compensation, that is, it is figured in directly.
- If the **_setGearingOffset()** command is programmed to current values during synchronization, the offset does not take effect until after synchronization. A compensating movement takes place.

Apply offset as superimposition

The **dynamicReference** parameter of the **_setGearingOffset** command can be used to specify whether the dynamic parameters refer to the total motion or the motion difference (V3.2 and higher).

- **TOTAL_MOVE**: Dynamic response parameters refer to the total motion. (Default setting)
  The transition is determined entirely on the basis of the offset values and the dynamic response parameters.
- **OFFSET_MOVE**: Dynamic response parameters refer to the motion difference.
  The transition is determined on the basis of the current synchronous operation definition as superimposed motion with the specified dynamic values.

Note

With a constant master value velocity, the dynamic transitions have a similar form and differ as a result of the dynamic response parameters that act differently.

See also

Example of applying offset as superimposition (Page 101)

2.2.2 Velocity gearing

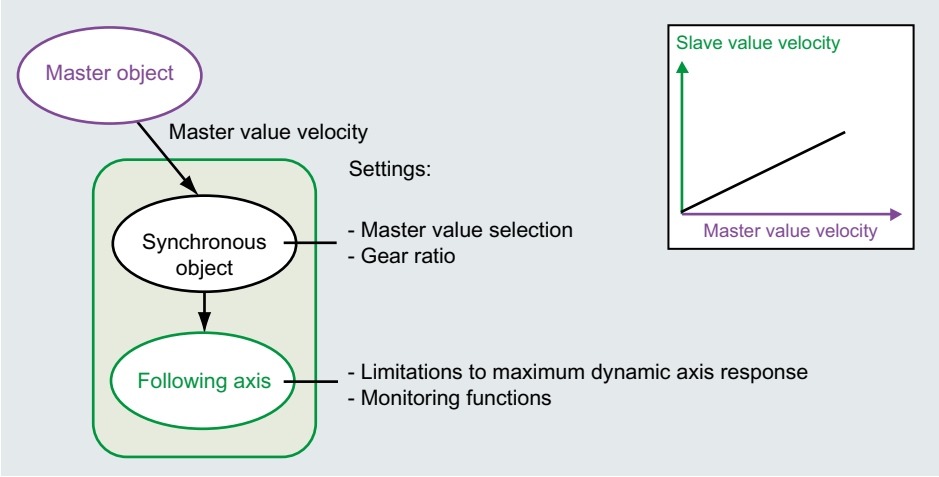


Figure 2-15 Velocity gearing

In contrast to gearing or camming, which relate to the position of an axis, **synchronous velocity operation** (V3.1 and higher) relates to the *velocity of an axis*. A velocity setpoint is calculated for the following axis. After activation, the axis travels immediately at the specified acceleration to the synchronous operation velocity. A linear transmission function is implemented.

The gear ratio can be specified as a positive floating-point number.

The gear ratio can be set in the same direction or in the opposite direction (corresponding to a negative gear ratio) using the **direction** parameter of the **_enableVelocityGearing()** command.

- **POSITIVE** means that the axes are running in the same direction.
- **NEGATIVE** means that the axes are running in opposite directions.

See also

Substitution of velocity gearing with absolute synchronous operation (Page 106)

2.2.3 Camming

Description

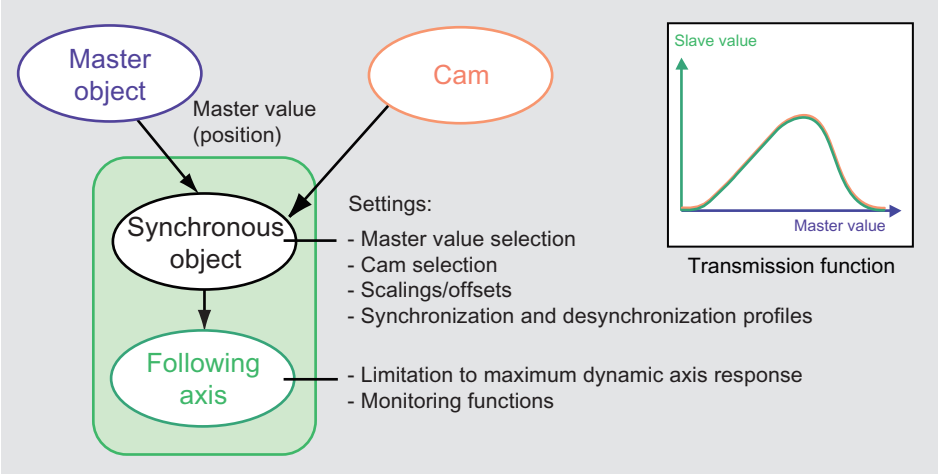


Figure 2-16 Camming

With **camming**, a non-linear transmission function between the master value position and following axis position is implemented using a **cam** .

See Cam, Definition (Page 224)

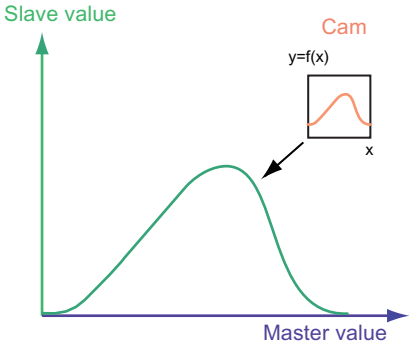
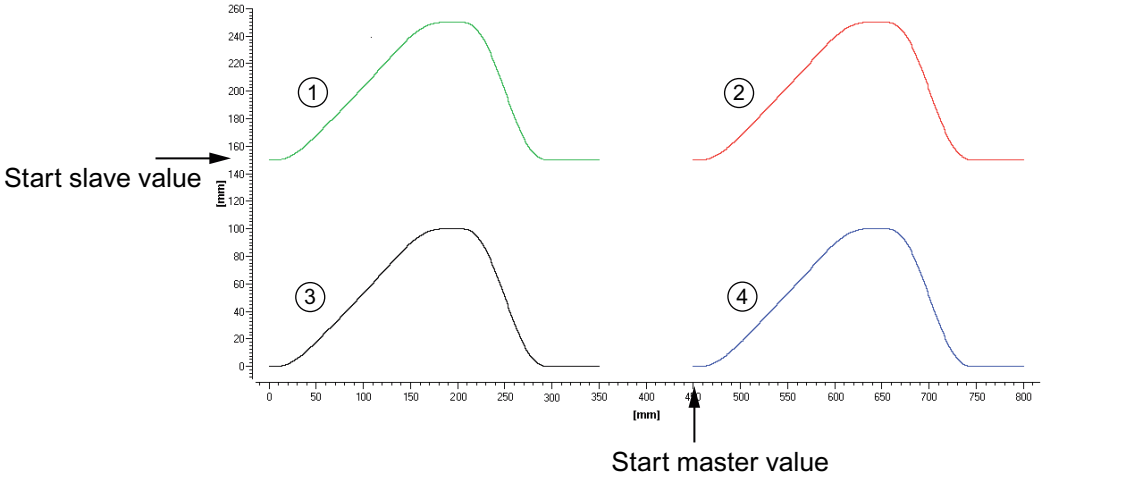


Figure 2-17 Example of transmission function for camming

The cam can be applied as an absolute cam or as a relative cam in both the definition range (master values) and the value range (slave values). The setting is made for the master values in the masterMode parameter and for the slave values in the slaveMode parameter of the _enableCamming() command.

The figure below compares the relationship between the master value and slave value with the following supplementary conditions:

- Same cam: here, with definition range {0.0, 300.0} and value range {0.0, 100.0}
- Same initial value of following axis: here, 150 mm
- Same initial value of master value: here, 450 mm (master value is modulo value 0 to 1000 mm)



1: Absolute camming on the master value side and relative camming on the slave value side
2: Relative camming on the master value side and relative camming on the slave value side
3: Absolute camming on the master value side and absolute camming on the slave value side
4: Relative camming on the master value side and absolute camming on the slave value side

Figure 2-18 Possible combinations of absolute/relative camming on the master/slave value side

Absolute camming on the slave value side

Absolute camming on the slave value side is set in **slave Mode:=ABSOLUTE**. With absolute camming on the slave value side, the slave values are taken directly from the value range of the cam. The offset on the slave value side is equal to zero, except when the synchronization criterion **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION** is set in the syncPositionSlave parameter, in which case an offset is specified. (Case 3 and 4 in Figure **Possible combinations of absolute/relative camming on the master/slave value side**).

Relative camming on the slave value side

Relative camming on the slave value side is set in **slaveMode:=RELATIVE**. With relative camming in slave mode, the initial value of the cam is offset to the slave value position at the start of synchronization (case 1 and 2 in Figure **Possible combinations of absolute/relative camming on the master/slave value side**).

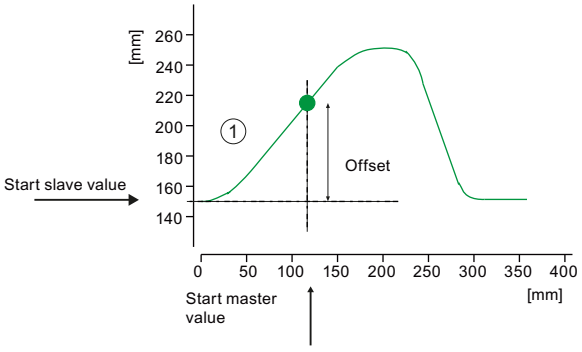
The offset on the slave value side is determined as follows:

- If programming is performed without a specified offset, the offset is determined from the offset of the cam initial value to the slave value position at the start of synchronization
- If programming is performed with a specified offset in the synchronization criterion setting **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION**, the offset is determined from the offset of the cam initial value to the slave value position at the start of synchronization plus the offset programmed in the **syncPositionSlave** parameter.

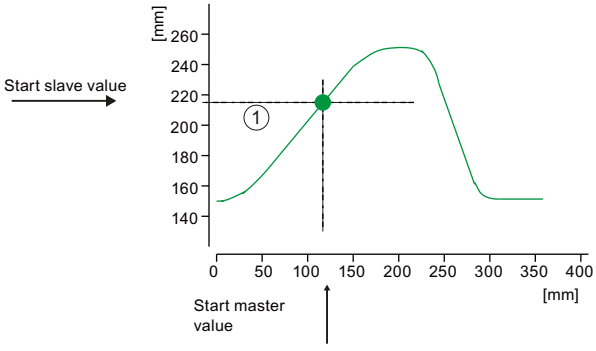
From the point at which the status becomes "synchronous", the offset on the slave value side remains constant in the transmission function.

In V4.2 and higher, there is new configuration data element **syncingMotion.camReferenceBySlaveModeRelative** with the following settings:

- Effectiveness in the case of relative camming on the slave value side and
  - on the absolute master value side with and without master value offset or
  - relative master value side with master value offset
- **BEGIN_OF_CAM** is based on the current following axis position to the initial cam value. During the synchronization, the following axis also covers the path differential between the start of the cam and the cam start value if this is not offset by a following value offset.



- **POSITION_AT_START_OF_CAMMING** is based on the current following axis position to the starting point of the synchronization within the cam. The difference between the slave-side starting point and the initial cam value does not have to be specified in the slave-value offset.



- In projects upgraded from < V4.2 to >= V4.2, the configuration data is set to **COMPATIBILITY_MODE**. This corresponds to **SYNCPOSITION_AT_START_OF_CAMMING** for synchronization at a standstill and **BEGIN_OF_CAM** with moving following axis or moving master value.
- When creating a new project as of V4.2, **syncingMotion.camReferenceBySlaveModeRelative:=BEGIN_OF_CAM** is set.

Absolute camming on the master value side

Absolute camming on the master value side is set in **masterMode:=ABSOLUTE**.
With **ABSOLUTE,** the master values are used as absolute values in the definition range of the cam.

Relative camming on the master value side

Relative camming on the master value side is set in **masterMode:=RELATIVE**. With relative camming on the master value side, the synchronization position on the master value side is assigned to the position within the cam definition range specified in the **camStartPositionMaster** parameter. The offset on the master value side is determined from the difference between the synchronization position on the master value side and the value specified in the **camStartPositionMaster** parameter.

If the position specified in **camStartPositionMaster** is not within the definition range of the cam, alarm "40017 Cam starting point is outside the definition range" is generated. From the point at which the status becomes "synchronous", the offset on the master value side remains constant in the transmission function.

Loading a new camming

A new cam is loaded by a recalling an **_enableCamming** command. Using the **synchronizingMode** parameter, the synchronization criterion is specified (e.g. at the end of the current camming with **AT_THE_END_OF_CAM_CYCLE**).

If the transition from the old cam to the new cam is already smooth, the synchronization parameters should be selected in such a way that either the synchronization length is very small for master value reference or the dynamics are very high for the forward time reference or, alternatively, use the synchronization profile with backward time reference.

Non-cyclic/cyclic cam application

The **cammingMode** parameter of the **_enableCamming()** command can be used to set the cam for either a non-cyclic application or a cyclic application.

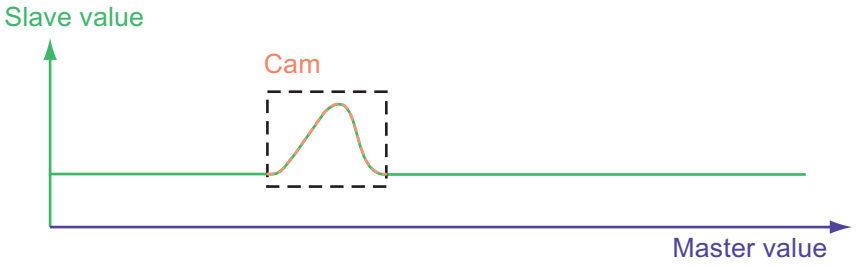


Figure 2-19 Non-cyclic cam application

- Non-cyclic (**NOCYCLIC**) means that the cam is applied exactly once in the defined master value range. When the end point or starting point of the cam is reached, the cam terminates itself.
  If the master value range is run through again either in the same direction or, after reversing, in the opposite direction, the cam is not applied again.

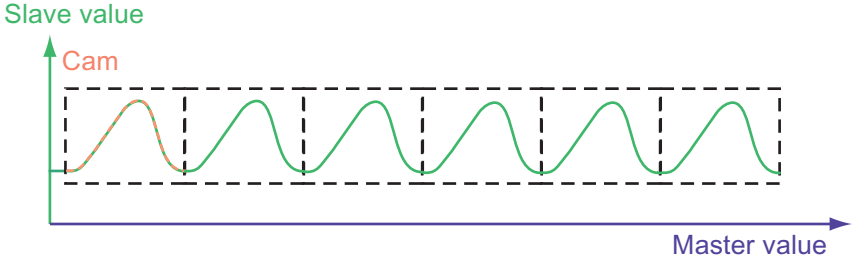


Figure 2-20 Cyclic cam application

- With the cyclic (**CYCLIC**) application of a cam, the definition range of the cam is mapped cyclically onto the master values.
  If the master values reverse, the cam is also continued cyclically beyond the original starting point.

Cyclic application of a cam with absolute synchronous operation on the slave value side

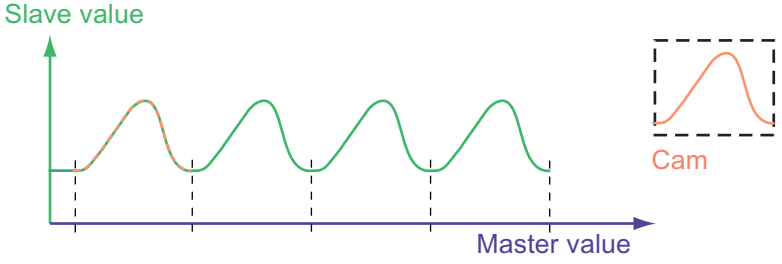


Figure 2-21 Cyclic absolute cam application with equal initial and end values on the slave value side

- If the function values of the cam are equal at the start and end of the definition range of the cam, the motion can be continued smoothly.
  This produces a periodic motion.

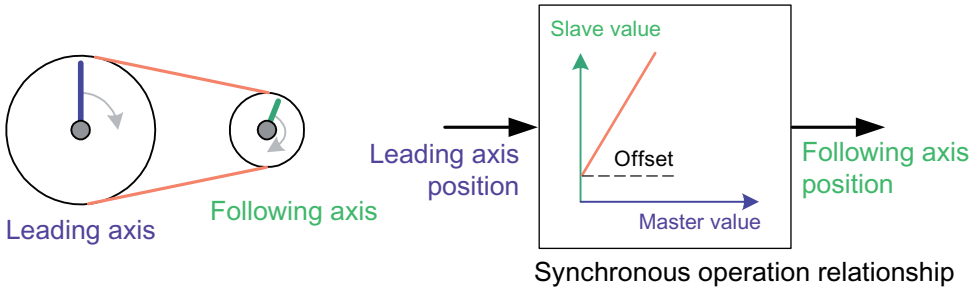


Figure 2-22 Cyclic absolute cam application with unequal start and end values on the slave value side

- If the function values of the cam are not equal at the start and end of its definition range, this results in a discontinuity in the position.
  This is limited on the following axis to the maximum dynamic values.

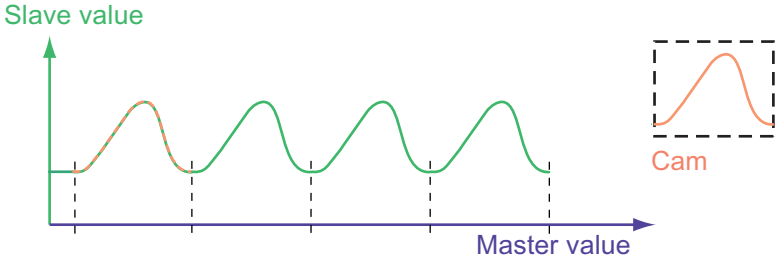


Figure 2-23 Example of cyclic cam application with identical start and end values

- If the function value of the cam is *not* identical, or equal in terms of a modulo relationship, at the start and end of its definition range, the new starting point of the cam is the end point of the executed cam.

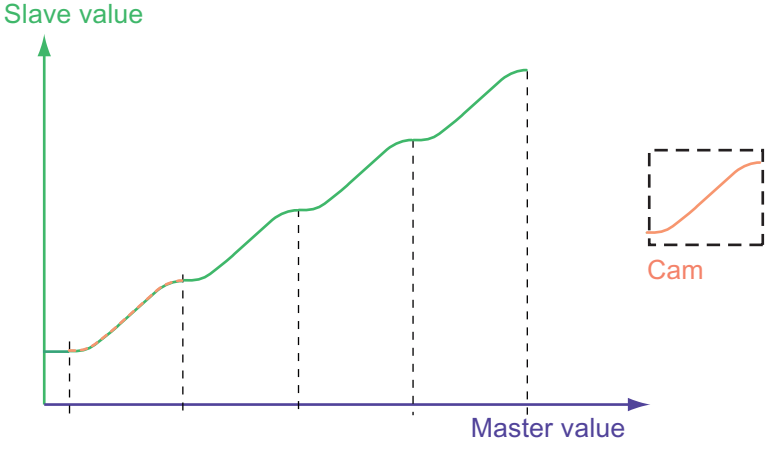


Figure 2-24 Example of cyclic cam application with different start and end values - relative

Cam direction

The **direction** parameter of the **_enableCamming** command can be used to set the cam in a positive or negative direction.

- **POSITIVE** means in the same direction. Increasing master values correspond to increasing values in the definition range of the cam, and vice versa.

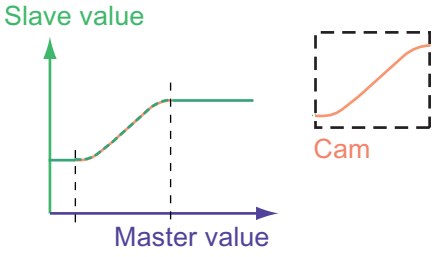


Figure 2-25 Positive cam application (POSITIVE)

- **NEGATIVE** means in the opposite direction. Decreasing master values correspond to increasing values in the cam definition range, and vice versa.
  The cam is mirrored at the center of its definition range.

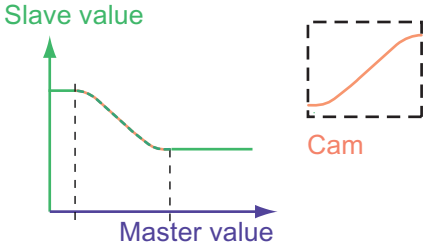


Figure 2-26 Negative cam application (NEGATIVE)

Example application:
The aim is to use the same cam for deceleration as for acceleration, but in the opposite direction.

Correction of camming motions

Synchronous motions can be corrected by changing the scaling and offset of the master value and the slave value.

Other options include:

- Offset and scaling on the cam itself
- Superimposed motions on the following axis
- On-the-fly setting of the reference point on the leading value source and the following axis

Scaling and offset

The *scaling* and *offset* can be specified on the *synchronous object* for camming on both the master value side and slave value side.

The slave value is determined from the master value using the following equation:

$$\text{Slave value} = F_{CAM}\left(\frac{\text{Master value} - \text{Offset}_{\text{master value}}}{\text{Scaling}_{\text{master value}}}\right) \bullet \text{Scaling}_{\text{slave value}} + \text{Offset}_{\text{slave value}}$$

Figure 2-27 Equation for scale and offset on the camming

See also Example of offset and scaling on the synchronous object (Page 98)

Scaling/offset on the cam

In addition to the option of the scaling/offset on the synchronous object, a scaling/offset is also possible on the cam. This enables a cam to be custom-adjusted in its definition range and value range.

See Scaling and offset (Page 226)

Changing the scaling and offset

The **_setCammingScale()** and **_setCammingOffset()** commands can be used to switch the scaling and offset within active, cyclic camming. The **activationMode** parameter determines when these take effect:

- For the next camming operation and all subsequent operations if **DEFAULT_VALUE** is set.
- For the current camming operation, only if **ACTUAL_VALUE** is set.
- For the current camming operation and all subsequent operations if **ACTUAL_AND_DEFAULT_VALUE** is set

Note the following:

- If synchronization of the **_enableCamming()** command is not yet active, the current scaling/offset is carried out without compensation; that is, it is included directly.
- If the **_setCammingScale()**/**_setCammingOffset()** command is programmed with new values during synchronization (using the **ACTUAL_VALUE** setting), the scaling/offset only becomes active after synchronization. A compensating movement takes place.

Effectiveness of scaling and offset

The **scaleSpecification**/**offsetSpecification** parameter of the **_setCammingScale()** or **_setCammingOffset()** command is used to program the effectiveness of a new scaling or offset procedure.

- With immediate effect (**IMMEDIATELY**)
- At the start of a new cycle for a cyclic cam application (**NEXT_CAM_CYCLE**)

Comment: If a **_setCammingScale()**/**_setCammingOffset()** command is canceled during the compensating movement due to another **_setCammingScale()**/**_setCammingOffset()** command with **NEXT_CAM_CYCLE**, compensation is canceled and a jump in the setpoints can occur. The new command is enabled at the beginning of the new cam cycle.

Examples

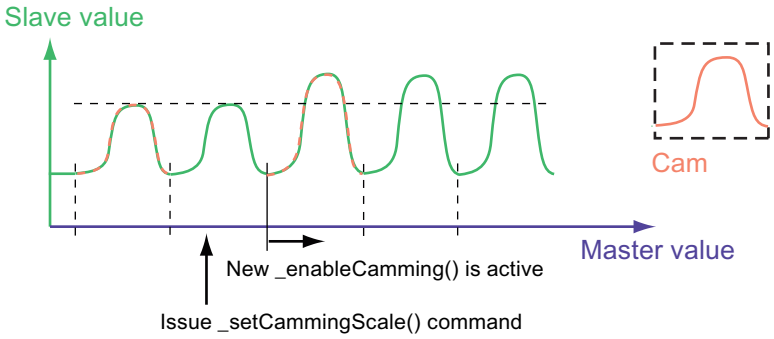


Figure 2-28 Example of switchover from scaling during cyclic synchronous operation; setting: activationMode:=DEFAULT_VALUE; effective: scaleSpecification:=NEXT_CAM_CYCLE

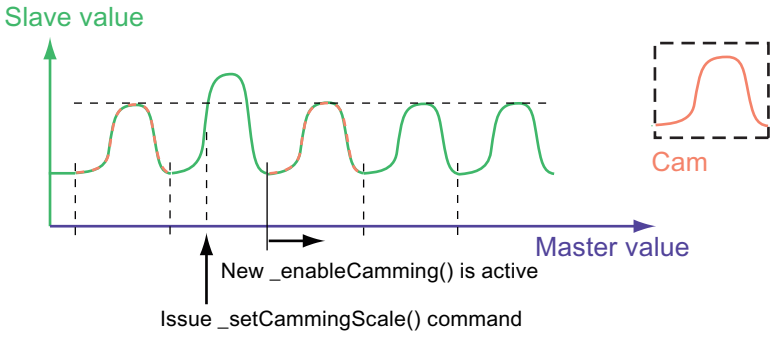


Figure 2-29 Example of switchover from scaling during cyclic synchronous operation; setting: activationMode:=ACTUAL_VALUE; effective: scaleSpecification:=IMMEDIATELY

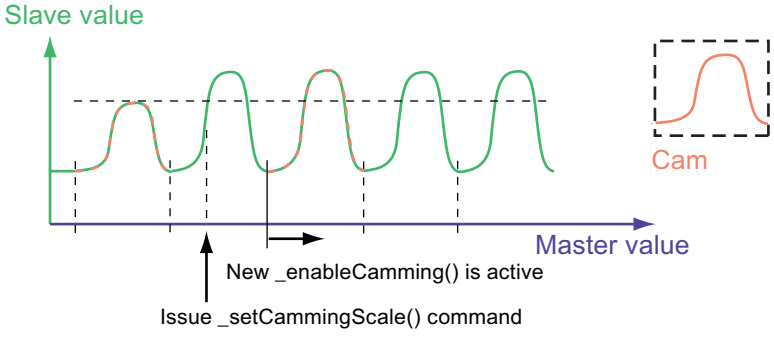


Figure 2-30 Example of switchover of scaling and offset during cyclic synchronous operation, ACTUAL_AND_DEFAULT_VALUE setting with effectiveness IMMEDIATELY

Applying scaling/offset as superimposition

The **dynamicReference** parameter of the **_setCammingScale()** or **_setCammingOffset()** command can be used to specify whether the dynamic parameters refer to the total motion or the motion difference (V3.2 and higher).

See **Apply offset as superimposition** at Gearing (Page 20).

See also

Display of the synchronous position (Page 61)

2.2.4 Setpoint/actual value coupling

Overview

When an axis is used as a master value object, the following can be configured for the synchronous operation:

- **Setpoint coupling**: The setpoint of the axis is used as the master value for the following axis. This is advantageous if the control specifies the setpoints for both the master axis and following axis, and the axes are to behave synchronously in relation to one another. In general, setpoint coupling is recommended for purposes of signal quality.
- **Actual value coupling with extrapolation** (V3.0 and higher): The actual value of an axis is used as the master value for the following axis.
  The actual value can be extrapolated in order to compensate for delays caused by actual value acquisition, actual/master value processing in the control, and the dynamic follow-up response of the following axis. Since the actual values are equal to the setpoints for the virtual axis, an extrapolated setpoint can be set.

When an external encoder is used as a master value object, the following can be configured for the synchronous operation:

- **Actual value coupling**: The actual value of an external encoder is used as the master value for the following axis.
- **Actual value coupling with extrapolation** (V3.0 and higher): The actual value can be extrapolated in order to compensate for delay times caused by actual value acquisition, actual/master value processing in the control, and the dynamic follow-up response of the following axis.

A tolerance window with respect to the actual value behavior can be specified for the actual value coupling.

Note

If the actual values/setpoints are required to be equal during a synchronous operation, the same Ti (actual value acquisition)/To (setpoint acceptance) times must be adopted for all drive units used (e.g. SINAMICS_Integrated, CU320).

See also

Actual value coupling with tolerance window (Page 39)

2.2.4.1 Actual value coupling with extrapolation

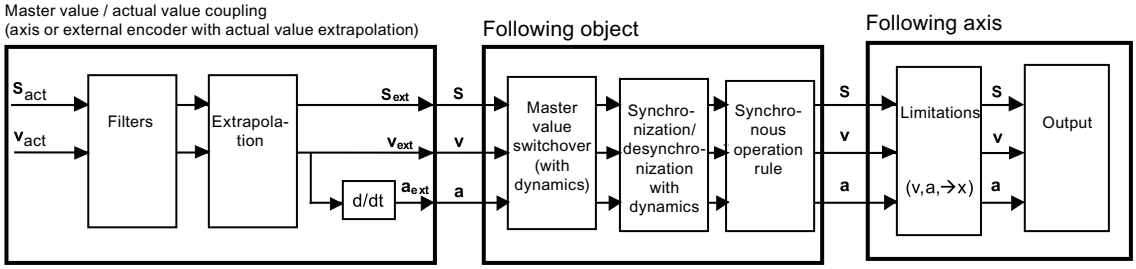


Figure 2-31 Principle of actual value coupling (overview)

For a synchronized group with actual value coupling (e.g. master value is the actual encoder value of an axis or an external encoder), the associated principle means delay times result because of bus communication, system cycle clocks and clock-pulse scaling, fine interpolation, position setpoint filters, and controller settings. These times can be compensated for using an extrapolation.

Extrapolation means you know the history and are looking into the future (extrapolation time).

The extrapolation time should be as short as possible for dynamic master value changes. An IPO : Servo pulse duty factor of 1:1 is good.

If an actual encoder value is assumed as the master value, it is useful to extrapolate the measured actual value for the synchronous operation in order to compensate for dead times that result within the system when acquiring actual values, e.g. due to bus communication and system processing times.

The extrapolation is set on the leading axis or on the external encoder.

Noisy sensor signals result in large velocity jumps which affect the extrapolation. These can be reduced or compensated by using suitable filter settings.

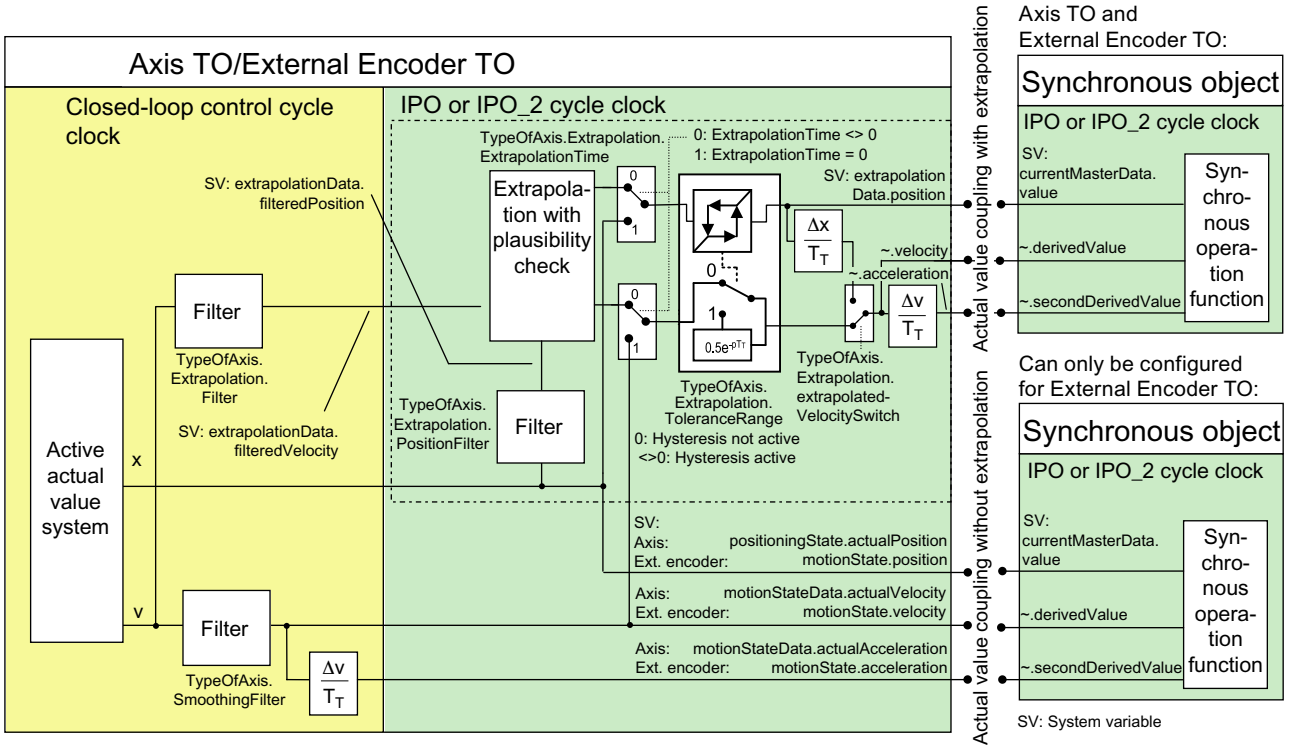


Figure 2-32 Actual value coupling with extrapolation for the Axis TO or External Encoder TO

You can find an overview of the actual value coupling with and without extrapolation in the axis configuration in the SCOUT signal flow dialog (project navigator, select Axis TO > **Signal flow > Extrapolation**).

Filtering of actual position

The actual position value for the synchronous operation can be filtered separately for the extrapolation using a PT2 filter (as of V4.1). The filter for the actual position value of the axis is set using the **typeOfAxis.extrapolation.positionFilter.T1** and **typeOfAxis.extrapolation.positionFilter.T2** configuration data. The filter acts on the actual position for the extrapolation. The velocity for the extrapolation is taken over from the actual values of the axis or External Encoder before application of the smoothing filter (**typeOfAxis.smoothingFilter**).

Filtering of actual velocity

The position is extrapolated based on the filtered or averaged velocity value.

We recommend setting the velocity filter (Extrapolation.Filter) first and then also using the position filter if the result is not satisfactory.

The position filter times are an additional factor to be taken into account in the extrapolation time.

The velocity filter (Extrapolation.Filter) does not affect the extrapolation time, but does have influence during dynamic changes to master values (due to delayed values). See also the *TO axis, Electric/hydraulic, External encoder* Manual.

- **TypeofAxis.Extrapolation.filter.timeConstant:** Time used for averaging or time constant for filtering

Extrapolation of actual velocity and actual position

These delay times can be compensated using an extrapolation.

- **TypeofAxis.Extrapolation.extrapolationTime:** Time definition for extrapolation

Extrapolation is not performed if 0.0 is specified. The extrapolated values (position and velocity) can be monitored (**extrapolationData** system variable). The extrapolation compensates for the local delays that result from use of the actual value instead of the setpoint.

Note

Extreme care must be taken when changing the extrapolation time to the runtime; otherwise knocking could result in the machine.

Note

SIMOTION contains utilities & applications which are included in the scope of delivery of SIMOTION SCOUT, a tool to assist in calculating the extrapolation times.

Switch for the velocity master value during master value extrapolation

The **TypeofAxis.Extrapolation.extrapolatedVelocitySwitch** configuration data element is used to select whether the velocity master value is generated from the extrapolated or filtered position master value using differentiation or if the extrapolated velocity master value is used.

Display

The extrapolated and filtered values are indicated in the following system variables:

- **sensorData[n].position**
- **sensorData[n].velocity**
- **sensorData[n].acceleration**

The system variables for **sensorData** are calculated in the position control cycle clock.

The actual axis value that is active for closed-loop control, the IPO cycle, and master value coupling

is displayed in the following variables:

- **positioningState.actualPosition**
- **motionStateData.actualVelocity**
- **motionStateData.actualAcceleration**

The system variables for **positioningState** and **motionState** are calculated in the IPO cycle.

These actual values are the reference for the output cam calculation in the IPO cycle, for the

actual value connection with the external encoder without extrapolation, for the actual value reference in the IPO cycle,

e.g. for profiles relating to actual position.

Reduction in reaction times/dead times

The **Execution.executionLevel:=SERVO** setting on the master value object, e.g. External Encoder technology object, can be configured in the Synchronous Object technology object and Following Axis technology object to enable execution of the IPO system component of the master value, synchronous operation, and axis in the Servo following actual value acquisition.

For further information, refer to the **Motion Control Technology Objects Axis Electric/ Hydraulic, External Encoder** Function Manual, "Motion execution/interpolator".

Transferring the actual velocity from the drive (V4.2 and higher)

With the
setting
typeOfAxis.numberOfEncoders.encoder_n.encoderValueType:=POSITION_AND_PROFIDRI VE_NIST_B, you have the option of converting the speed of rotation transferred in **PROFIdrive NIST_B** to a velocity and applying this value as the actual velocity of the encoder/sensor. In this case, the actual position of the sensor does not need to be differentiated to derive the actual velocity.

With the setting **typeofAxis.numberOfEncoders.encoder_n.encoderValueType:= POSITION_AND_DIRECT_NIST**, a speed of rotation transferred in the I/O area
and normalized as NIST_B is taken as the actual value and converted to an
actual velocity. In this case, 4000H corresponds to 100%. The address is set
in **typeofAxis.numberOfEncoders.encoder_n.sensorNist.logAddress**, and the reference value is set in **typeofAxis.numberOfEncoders.encoder_n.sensorNist.referenceValue**. With encoders with nact evaluation, the speed determined by the encoder and the resulting velocity can be accepted by the encoder. In this case, the actual position of the sensor does not need to be differentiated to derive the actual velocity. Two methods of transmission are available:

- Transmission in the PROFIdrive message frame
- Transmission in the I/O area

See also *section Actual value acquisition / Actual value system in the Manual TO Axis electric/ hydraulic, External Encoder*.

2.2.4.2 Actual value coupling with tolerance window

If the master value is superimposed with high-frequency noise signals that cannot be followed by the synchronous operation, this can cause the dynamic response boundaries to be exceeded or the master value to briefly change directions during synchronization.

In the **typeOfAxis.extrapolation.toleranceRange** configuration data element *on the master axis* or external encoder, a tolerance window can be set around the actual position

(V3.1 and higher), to prevent, for example, the dynamic limits from being exceeded on the following axis in the case of a master value with high-frequency disturbances, or direction changes during synchronization.

2.2.5 Synchronization

In order for the following axis to follow the master value according to the transmission function, the following axis must first be synchronized to the master value.

The type of synchronization is determined from several assignable parameters/settings:

- The **synchronization criterion/synchronization position**, which corresponds to the setting specified in the **synchronizingMode** parameter; the synchronization position on the master value side and/or the synchronization position on the slave value side are directly specified here or are derived from the synchronization criterion and, if necessary, the transmission function
- The **synchronization direction**, the motion direction of the slave values during synchronization; can be set in the **synchronizingDirection** parameter
- **Position of synchronization range relative to synchronization position:** Leading, trailing, or symmetrical synchronization; can be set in the **syncPositionReference** parameter
- the **reference of the synchronization profile**; can be set in the **syncProfileReference**
  - **Synchronization over a specifiable master value distance**
    The synchronization length over the master value is specified in the synchronization command.
  - **Synchronization profile via specifiable dynamic response parameters (time reference)**
    The dynamic response parameters are specified in the synchronization command.

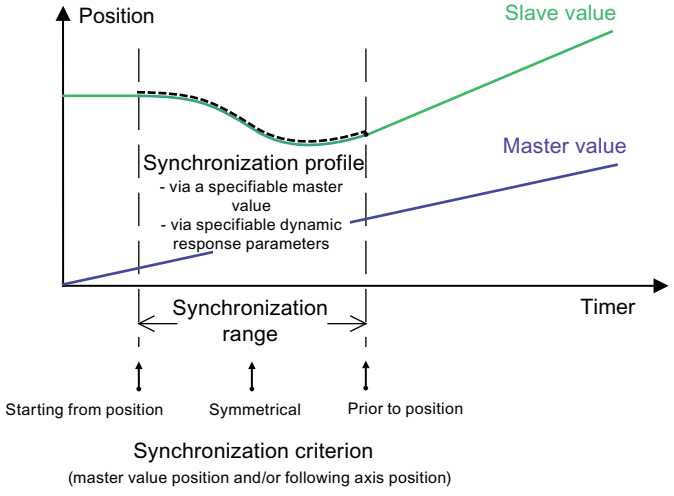


Figure 2-33 Parameters for synchronization

Properties	Synchronization via a specifiable master value distance	Synchronization profile based on specifiable dynamic response parameters, leading synchronization	Synchronization profile based on specifiable dynamic response parameters, trailing synchronization
Dynamic response properties			
• Constant velocity synchronization profile	Yes	Yes	Yes
• Constant acceleration synchronization profile	No	With SMOOTH velocity profile setting	With SMOOTH velocity profile setting
Adherence to dynamic response parameters (without limiting functions on the following axis side)	No User can influence the dynamic response via the synchronization length	With master value and constant velocity, otherwise master value dynamic response is superimposed	Yes
Dynamic response can be adapted to the master value dynamic response	Indirectly	With dynamicAdaption setting	With dynamicAdaption setting
Applicability to stationary master value			
• If following axis is at a standstill	Conditional Following axis must already be at the synchronous position, for example, with relative gearing	Conditional Following axis must already be at the synchronous position, for example, with relative gearing	Yes
• With moved following axis	No	No	Yes
Applicability to master value with constant velocity			
• If following axis is at a standstill	Yes	Yes	Yes
• With moved following axis	Yes	Yes	Yes
Applicability to master value with non-constant velocity			
• Master value with constant acceleration / deceleration	Yes Superimposition of master value dynamic response	Yes Superimposition of master value dynamic response	Conditional With extended look-ahead or dynamic response of synchronization >> master value dynamic response
• Modified master value dynamic response or faulty/ noisy master value signal	Yes Superimposition of master value dynamic response	Yes Superimposition of master value dynamic response	No Exception: Dynamic response of synchronization >> master value dynamic response
Synchronization properties			

Properties	Synchronization via a specifiable master value distance	Synchronization profile based on specifiable dynamic response parameters, leading synchronization	Synchronization profile based on specifiable dynamic response parameters, trailing synchronization
• Synchronism reached after starting the synchronization	Yes Exception: master value changes motion direction	Yes Exception: master value changes motion direction	Conditional No, if master value dynamic response > resulting dynamic response of synchronization or varying master value dynamic response; see above
• Specification of synchronous position after starting the synchronization	Supported	Supported	No

Properties of different synchronization options

2.2.5.1 Synchronization criterion

Synchronization criterion/synchronization position

The **synchronization criterion** can be set for synchronization using the synchronizingMode parameter of the **_enableGearing()**/**_enableCamming()** or **_disableGearing()**/**_disableCamming()** command as outlined below. Synchronization can be performed over several modulo ranges of the master value or slave value.

Synchronization on current master value position without specification of an offset on the slave value side

The current master value position is the synchronization criterion and the synchronization position on the master value side. The synchronization criterion is set with **synchronizingMode:=IMMEDIATELY**. The **syncPositionMaster** parameter is not active. An offset on the slave value side is not specified, and the **syncPositionSlave** parameter is not active. With relative camming on the master value side, the **camStartPosition** parameter is active. Synchronization starts immediately. Synchronization occurs subsequently.
The **syncPositionReference** parameter is not active.

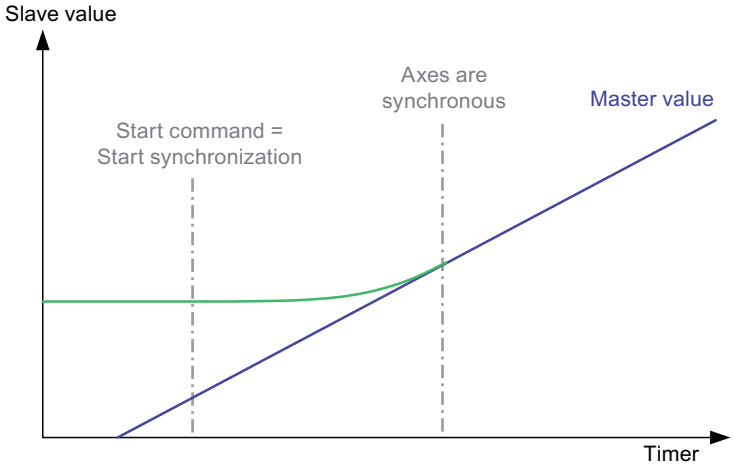


Figure 2-34 Example of synchronization - immediately active, trailing synchronization, absolute without offset, ratio 1:1

Synchronization on current master value position with specification of an offset on the slave value side

The current master value position is the synchronization criterion, and an offset on the slave value side is specified. The synchronization criterion is set with **synchronizingMode:=IMMEDIATELY_AND_SLAVE_POSITION**. The synchronization position on the master value side is the current master value position. The **syncPositionMaster** parameter is not active. The offset on the slave value side is specified in the **syncPositionSlave** parameter. With relative camming on the master value side, the **camStartPosition** parameter is active. Synchronization starts immediately. Synchronization occurs subsequently.
The **syncPositionReference** parameter is not active.

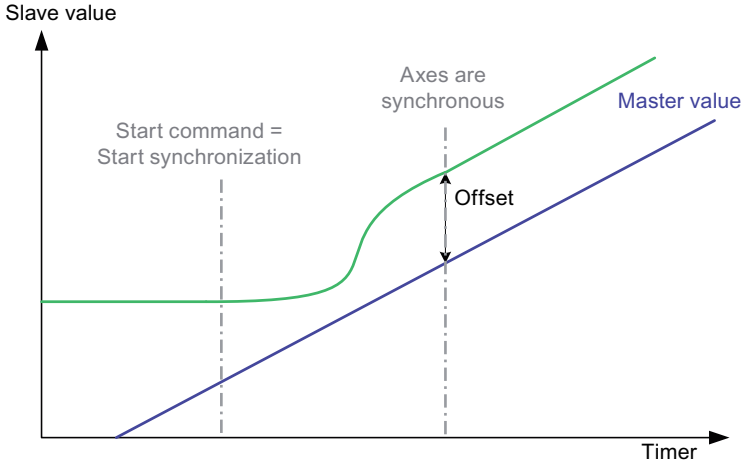


Figure 2-35 Example of synchronization - immediately active, trailing synchronization, absolute and offset on following axis position, ratio 1:1

Synchronization on specified master value position without specification of an offset on the slave value side

The specified master value position is the synchronization criterion. The synchronization criterion is set with **synchronizingMode:=ON_MASTER_POSITION**. The synchronization position on the master value side is set in the **syncPositionMaster** parameter. An offset on the slave value side is not specified, and the **syncPositionSlave** parameter is not active. With relative camming on the master value side, the **camStartPosition** parameter is active.
The **syncPositionReference** parameter specifies whether to activate leading, symmetrical (only for synchronization via a specifiable master value distance), or trailing synchronization.

Regarding the start of synchronization, see Position of synchronization range relative to synchronization position (Page 48).

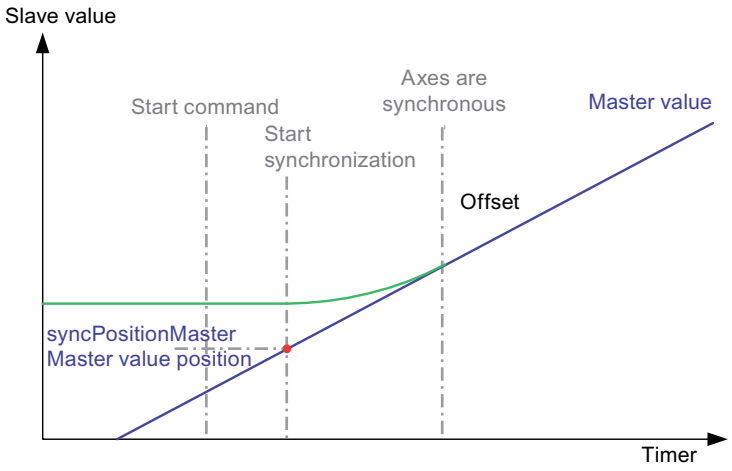


Figure 2-36 Example of synchronization - specification of master value synchronization position, trailing synchronization, absolute, ratio 1:1

Synchronization on specified master value position with specification of an offset on the slave value side

The specified master value position is the synchronization criterion. The synchronization criterion is set with **synchronizingMode:=ON_MASTER_AND_SLAVE_POSITION**. The synchronization position on the master value side is set in the **syncPositionMaster** parameter. With relative camming on the master value side, the **camStartPosition** parameter is active. The offset on the slave value side is specified in the **syncPositionSlave** parameter.
The **syncPositionReference** parameter specifies whether to activate leading, symmetrical (only for synchronization via a specifiable master value distance), or trailing synchronization.

Regarding the start of synchronization, see Position of synchronization range relative to synchronization position (Page 48).

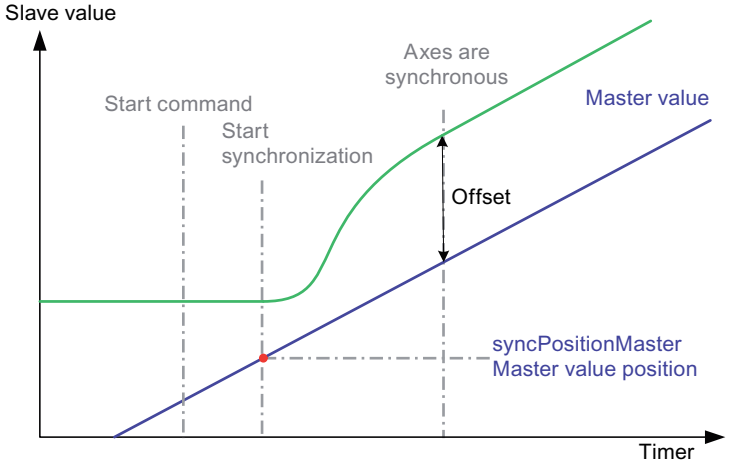


Figure 2-37 Example of synchronization - specification of master value synchronization position and following axis offset, trailing synchronization, absolute, ratio 1:1

Synchronization on the specified following axis position

The specified following axis position is the synchronization criterion. The synchronization criterion is set with **synchronizingMode:=ON_SLAVE_POSITION**. The synchronization position on the slave value side is specified in the **syncPositionSlave** parameter. An offset on the slave value side cannot be specified. The synchronization position on the master value side is determined from the application of the inverse transmission function to the synchronization position on the slave value side. With relative camming on the master value side, the **camStartPosition** parameter is active. The **syncPositionMaster** parameter is not active. Synchronization starts if the synchronization position specified in the **syncPositionSlave** parameter is reached on the following axis as a result of a motion initiated elsewhere.

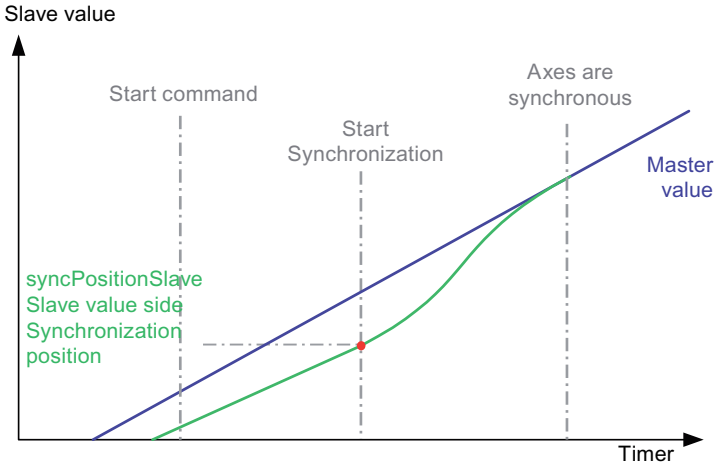


Figure 2-38 Example of synchronization - specification of following axis synchronization position, trailing synchronization, absolute, ratio 1:1

Synchronization at the end of the current camming cycle

The master value position at the end of the current camming cycle is the synchronization criterion. This setting can only be assigned in conjunction with relative camming on the master value side and already active camming. The synchronization criterion is set with **synchronizingMode:=AT_THE_END_OF_CAM_CYCLE**. The synchronization position on the master value side is the master value position at the end of the current camming cycle. The **syncPositionMaster** parameter is not active. With relative camming on the master value side, the **camStartPosition** parameter is active. An offset on the slave value side cannot be specified, and the **syncPositionSlave** parameter is not active. The **syncPositionReference** parameter specifies whether to activate leading, symmetrical (only for synchronization via a specifiable master value distance), or trailing synchronization.

2.2.5.2 Synchronization direction

The **synchronizingDirection** parameter of the synchronous operation commands can be used to specify the direction of the motion for synchronization. If a specific synchronization direction is specified, the synchronization motion is in this direction only.

The synchronization direction of the following axis in the synchronization phase can be specified with the **synchronizingDirection** parameter in the **_enableGearing(), _disableGearing(), _enableCamming()**, and **_disableCamming()** commands (V3.1 and

higher). This function is relevant, for example, for axes, for which synchronization is possible in both directions.

For information on axes with backstop, refer to the **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Manipulated variable limitation (backstop)"

Synchronization with direction of specification can be set as follows:

- **Maintain present system behavior (SYSTEM_DEFINED):**
  This corresponds to the **shortest path** setting, however the direction of motion is maintained for the axis motion.
- **Maintain direction of the following axis (SAME_DIRECTION):**
  The current direction of following axis motion is maintained during the synchronization phase.
  - The direction of following axis motion is maintained during synchronization when the master value is at a standstill.
  - The synchronization occurs in the positive direction during synchronization at both the master value standstill and the following axis standstill.
- **POSITIVE_DIRECTION** setting: A **positive** synchronization direction is defined.
- **NEGATIVE_DIRECTION** setting: A **negative** synchronization direction is defined.
- **SHORTEST_WAY** setting: Synchronize to the **shortest way**, regardless of which direction of motion ensues in the synchronization phase.
  With this setting, however, a direction reversal could occur during synchronization.

2.2.5.3 Position of synchronization range relative to synchronization position

The position of the synchronization range relative to the synchronization position can be set with the **syncPositionReference** parameter of the **_enableGearing()** or **_enableCamming()** command:

- Synchronize before the specified synchronization position: **Leading synchronization (syncPositionReference:=BE_SYNCHRONOUS_AT_POSITION)**
  - The end point of the synchronization is specified via the synchronization criterion.
  - The starting point of synchronization is determined for synchronization via a specifiable master value distance, on the basis of the specified synchronization length, and is calculated with reference to the dynamic response parameters of the system in accordance with the specified dynamic values and the master value behavior.
- Synchronize starting from the specified synchronization position: **Trailing synchronization (syncPositionReference:=SYNCHRONIZE_WHEN_POSITION_REACHED)**
  - The starting point of synchronization is specified directly or implicitly (by means of the following axis position).
  - The end point of the synchronization is calculated for synchronization via a specifiable master value distance from the specified synchronization length and is calculated with reference to the dynamic response parameters of the system according to the dynamic response parameters and the master value behavior.
- Symmetrically relative to the specified synchronization position **(syncPositionReference:=SYNCHRONIZE_SYMMETRIC)**
  - With synchronization via a specifiable master value distance, the starting point and end point of synchronization are determined on the basis of the master value positions, in accordance with the specified synchronization length.
  - Synchronization with the SYNCHRONIZE_SYMMETRIC specification is not possible with a synchronization profile using specifiable dynamic response parameters (RELATE_SYNC_PROFILE_TO_TIME). This command is rejected with TO alarm "30001: illegal parameter".

Synchronization starts under the following conditions:

- With trailing synchronization: When the synchronization position on the master value side or slave value side is reached.
- With symmetrical synchronization: When the master value has reached the synchronization position, reduced by half the synchronization length in the master value direction of motion Master value>= synchronization position (synchronization length in direction of motion of master value/2).
- With leading synchronization: When the synchronization position on the master value side is reached, reduced by the synchronization length in the direction of motion.

Note the following:

- With trailing synchronization and absolute synchronous operation, the following axis distance to be traveled according to the transmission function based on the progress of the master value must be applied too. For this reason, leading synchronization should be used whenever possible.
- With synchronization criterion **synchronizingMode:=IMMEDIATELY** or **IMMEDIATELY_AND_SLAVE_POSITION**, trailing synchronization is implicitly present.

2.2.5.4 Synchronization via a specifiable master value distance

With synchronization via a specifiable master value distance, a synchronization profile is calculated from a specifiable path length of the master value and is applied relative to the master value.

The setting is made with the **syncProfileReference:= RELATE_SYNC_PROFILE_TO_LEADING_VALUE parameter.**

The path length of the master value is specified in the parameter **syncLength.** As a result, synchronism is always achieved in the setpoint specification. The dynamic response during synchronization is dependent on the calculated profile via the master value and on the change in the master value. The dynamic response values specified in the command are not active.

Synchronization length

The synchronization operation takes place as long as the master value is within this defined length. No dynamic response values are taken into account. The profile is calculated as a function of the master value velocity. (See **Synchronization profile type**)

The **synchronization range** is specified using the synchronization length of the master value defined in the **syncLength** parameter of the **_enableGearing()** or **_enableCamming()** commands.

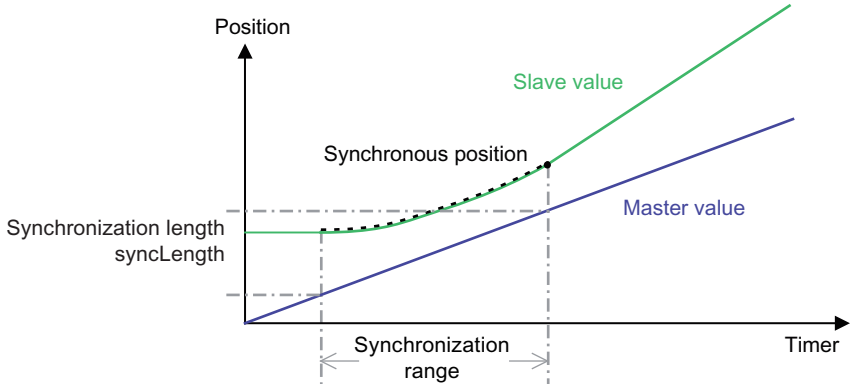


Figure 2-39 Synchronization length for synchronization via a specifiable master value distance

The "synchronous" status is set immediately if the master value source and following axis are at a standstill and the synchronization criterion has already been fulfilled. In this case, the message "50006 Activation/deactivation of synchronous operation executed directly" is output.

Synchronization profile type

The synchronization profile type for synchronization with master value reference is set using the **syncingMotion.velocityMode** configuration data element:

- In the **CONTINUOUS** setting (default), the synchronization profile is calculated with a constant position and constant velocity, but not with constant acceleration.
  The slave value is synchronized using a polynomial profile and the master value. Therefore, the resulting velocity and acceleration of the following axis for synchronization are dependent on the synchronization length and the dynamic response of the master value during synchronization.
- In the **NON_CONTINUOUS** setting, the synchronization profile is calculated using the specified master value length with a constant position only in the slave value behavior. The slave value is synchronized using a linear profile and the master value.

2.2.5.5 Synchronization profile

Synchronization profile based on specifiable dynamic response parameters (time reference)

When this synchronization profile is used, a synchronization profile is calculated according to the specified dynamic response parameters and the master value dynamic response that exists at the start of the profile. The profile is applied *according to the master value* for leading synchronization and *according to time* for trailing synchronization.

The setting is made with the **syncProfileReference:= RELATE_SYNC_PROFILE_TO_TIME parameter.**

The dynamic response for the synchronization is specified in the dynamic response parameters for the synchronous operation commands. A velocity profile with constant velocity (**TRAPEZOIDAL**) or constant acceleration (**SMOOTH**) can be specified using the **velocityProfile** parameter. For synchronization with a synchronization profile based on dynamic response parameters and with constant acceleration, any reversing that occurs during synchronization is at zero acceleration at the reversing point.

The synchronization profile based on dynamic response parameters can be applied:

- For leading synchronization
- For trailing synchronization

Symmetrical synchronization is not possible.

See also Position of synchronization range relative to synchronization position (Page 48), as well as section Adapt the synchronization velocity to the master value velocity (Page 107).

Application

A synchronization profile based on dynamic response parameters is especially suited for:

- Time-optimized synchronization, according to dynamic response specifications

Adaptation to the dynamic response of the master value (leading and trailing synchronization)

If the current dynamic response variables of the master value are larger than the dynamic response parameters of the synchronization command, the parameters can be automatically adapted to the dynamic response parameters (as of V3.1).

Parameters for adapting the synchronization dynamic response to the target dynamic response can be assigned on the synchronous object under **Settings (syncingMotion.synchronizing-Adaption)**.

If dynamic response adaptation is disabled, the synchronization dynamic response is no longer adapted to the required target dynamic response. This can result in the situation where, during trailing synchronization, the synchronized axis can no longer synchronize with the leading axis. During leading synchronization, the synchronization motion may not be started under certain circumstances.

Overdrive factor

The permissible magnification of the specified dynamic response values for making up a constant path difference is specified using the magnification factor **(syncingMotion.overdriveFactor)** under **Settings**. The magnification factor relates to the dynamic response of the master value. At 100% magnification, the dynamic response of the synchronization is adapted to the current dynamic response of the master value, taking into account the transmission. When **overDriveFactor** > 100% is set and has taken effect, the "synchronous" status can also be established if the master value is in the acceleration or deceleration phase during synchronization.

If an overshoot occurs, alarm 40012 "Dynamic limitations (type: ...) are violated" is output at the synchronous object.

Application

If a low synchronization velocity is selected on the command, the adaptation is set, and a corresponding overshoot factor is selected, the synchronization velocity of the following axis is adapted to the master value velocity.

Direction-dependent dynamic response

Direction-dependent or direction-independent effectiveness of the programmed dynamic response values can be set with **syncingMotion.directionDynamic**.

See **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Dynamic limits"

Leading synchronization with synchronization profile based on dynamic response parameters

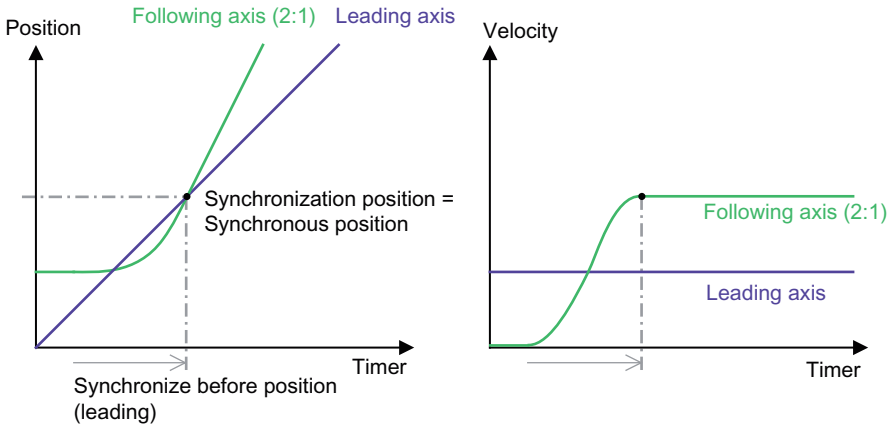


Figure 2-40 Example of leading synchronization (gearing with 2:1 gear ratio, synchronize following axis from standstill)

Only with a synchronization profile based on specifiable dynamic response parameters is a synchronization profile calculated for leading synchronization, taking into account the current master value velocity, the current position, and the dynamic response of the following axis, as well as the dynamic values for synchronization specified in the command. The synchronization profile is then traversed relative to the master value. In addition, if the master value dynamic response changes, the synchronization profile is not recalculated. For this reason, a change in the master value velocity can be seen superimposed in the synchronization operation.

In addition, with the **Extended Look ahead** setting, the acceleration of the master value in the synchronization profile is not taken into account.

Start of synchronization

The synchronization operation starts:

- At the time calculated by the system from which the specified dynamic response parameters can be optimally synchronized with respect to time at a constant master value velocity
- Immediately, if it is not possible to calculate an optimum synchronization time and the synchronization position can be reached (with static master value, for example)

Application

Leading synchronization is appropriate in the following cases:

- If there is to be synchronism at the specified synchronous position and the synchronous position can be easily specified from the application, taking into account the required synchronization operation.
- If dynamic response changes for the master value can be expected during synchronization and are taken into account in the synchronization profile but are not to be reinforced through extrapolation.

Comments

- The programmed velocity profile (SMOOTH, TRAPEZOID) is applied.
- The **syncingMotion.smoothAbsoluteSynchronization** configuration data element is not relevant for leading synchronization.
- Synchronization with extended look-ahead is not active with leading synchronization.
- With leading synchronization, the current master value velocity, and the resulting synchronization profile, if there is insufficient time for synchronization before the master value reaches the synchronization position, synchronization does not occur. The status can be read out via system variables.
  Exception: Modulo master value; in this case, the next possible synchronization position is awaited.

Trailing synchronization with synchronization profile based on dynamic response parameters

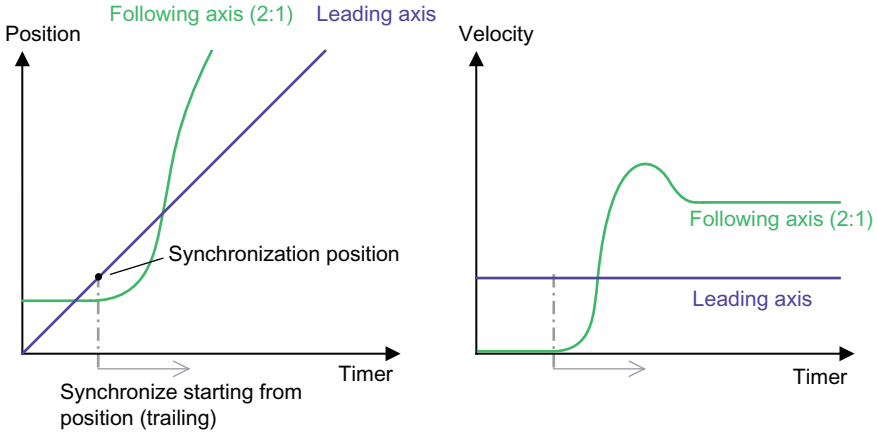


Figure 2-41 Example of trailing synchronization (gearing with 2:1 gear ratio, synchronize following axis from standstill)

With trailing synchronization, the synchronization operation starts when the synchronization criterion is reached. Taking into account the current master value velocity and the specified dynamic response values, the system calculates and executes a time-related synchronization profile such that synchronization is achieved as fast as possible. When calculating the synchronization profile, the current master value acceleration is only taken into account in cases where extended look-ahead is specified.

Changes of the master value for the synchronization are directly taken into account.

Application

Trailing synchronization is appropriate in the following cases:

- When the current master value position must be used directly as the synchronization position.
- When no dynamic response changes of the master value during synchronization are expected.
- When other reasons dictate that synchronization can take place only after the synchronization position.
  Depending on the position of the slave value at the synchronous position, large dynamic movements of the slave value may occur since, in order to comply with the dynamic limits and taking into account the master value dynamic response, the following must occur:
  - Synchronism must be achieved
  - If there are dynamic response changes to the master value, the (anticipated) position changes to the master value must be made up for.

Comments

Trailing synchronization is not suitable for non-constant velocity and acceleration of the master value (i.e. if the acceleration/deceleration changes continually).

Smooth synchronization

Absolute trailing synchronization with consideration of jerk

The **syncingMotion.smoothAbsoluteSynchronization** configuration data element can be used to specify whether a smooth velocity profile is supported during the succeeding synchronization (as of V3.2).

Synchronization is not calculated subject to changes in the master value velocity.

- Provision for jerk during absolute synchronization can be made by setting **syncingMotion.smoothAbsoluteSynchronization:= YES**.
- A setting of **NO** means that jerk will not be taken into account during absolute synchronization, even with velocity profile **SMOOTH**.

Trailing synchronization with extended look ahead

The synchronization with extended look-ahead allows a constant acceleration/deceleration of the master value to be used during the synchronization.

- With synchronization with **standard look ahead**, the position and the velocity are included in the synchronization calculation.
- For synchronization with **extended look-ahead**, the current values of the master axis for position, velocity and acceleration/deceleration are used to calculate the synchronization distance of the following axis (as of V3.2).
  Any changes to the acceleration of the master axis during the synchronization action are ignored and can cause a longer synchronization distance than for synchronization with standard look-ahead.
- During the braking phase of the master synchronization, it can occur that the synchronization process is terminated only after the braking phase of the master. In this case, synchronization without **extended look-ahead**, i.e. in **Standard** mode, can be executed. It can nevertheless occur that the following axis becomes synchronous only after the braking phase.

The function can be activated via the **synchronizingWithLookAhead** parameter of the **_enableGearing()** command.

Extended look-ahead can be preset on the synchronous object via system variable **.gearingSettings.synchronizingWithLookAhead** (as of V4.0).

See also

Position of synchronization range relative to synchronization position (Page 48)

2.2.5.6 Settings for evaluation of the master value behavior during synchronization

Toleration of a master value reversal during synchronization:

During synchronization, when the direction of the master value is reversed, the synchronization process is canceled with error message 50007 output at the synchronous object.

If the direction reversal is based on tolerable and predominantly non-influenceable master value fluctuations during downtime, as they occur in the case of an actual value coupling (so-called actual value "noise") or primary extrapolation of the master value, cancelation of synchronization can be prevented by specifying a tolerance band for the master value reversal (master value hysteresis) (as of V4.0).

The maximum master value fluctuation to be tolerated by the system is to be parameterized in the configuration data **syncingMotion.masterReversionTolerance** in the master value positioning unit. An effective hysteresis at a value greater than 0 freezes the master value in the reversal point during direction reversal and thus simulates a still-standing master value for synchronization.

The master value tolerance continuously acts on the active or soon-to-be active master value (see master value switching), beginning with activation of the master value and the hysteresis.

Note

When a master value tolerance > 0 is specified, the system is permitted to immediately offset the following axis change corresponding to one of these master value variables, with maximum dynamics on the following axis, e.g. during use in connection with time-related synchronization. During this process, the current effective coupling factor must also be taken into account.

The variable of the master value tolerance should therefore be selected at as low a level as possible and should be immediately oriented toward the measured fluctuations of the actual value or the extrapolation error.

The effective hysteresis, and thus also an existing reversal point, can be reset by setting the master value tolerance to the value 0. The change, in the same way as setting the tolerance to a value greater than 0, immediately becomes active.

Effective direction of the hysteresis

The effective direction of the hysteresis is automatically determined by the system and is maintained by the permanent functionality that is also outside of synchronization.

As of V4.4, you can specify the effective direction of the master value via the configuration data item **syncingMotion.toleranceDirection**.

- **BOTH_DIRECTIONS** (default)
- **POSITIVE**. The effective direction of the tolerance band is positive. The tolerance band is effective in the positive direction for a master motion, i.e. the hysteresis function takes effect when the master moves from a positive to a negative direction, but not vice versa.
- **NEGATIVE**. The effective direction of the tolerance band is negative. The tolerance band is effective in the negative direction for a master motion, i.e. the hysteresis function takes effect when the master moves from a negative to a positive direction, but not vice versa.

No synchronized status while master value in the hysteresis

As of V4.4 there is a configuration data item **syncingMotion.enableSyncWithinTolerance** with which the reaching of the **synchronized** status is retained within the hysteresis of the master value, or the new response that the **synchronized** status is only reached by the master value when it has left the hysteresis range.

You can set that synchronous operation does not go into **synchronized** status and therefore refers to the current position of the master value using the synchronous operation rule as long as the master value is in the hysteresis as a result of a change of direction. In this way, the difference that results from the synchronous operation rule and the position of the master value at the start of the hysteresis and the current position, is no longer traversed with the maximum dynamic response of the following axis.

If a master value reversal tolerance greater than zero is set for the synchronization, with

- synchronization with master value reference: relative synchronization at rest
- synchronization trailing with time reference: synchronization ends within the tolerance

an unexpected traversing motion can occur when the synchronized status is reached, because the difference resulting from the synchronous operation slave value at the start of the hysteresis and from the synchronous operation slave value for the current master value position is traversed immediately with the maximum dynamic response of the following axis when the **synchronized** status is reached.

Example of switchover in the hysteresis:

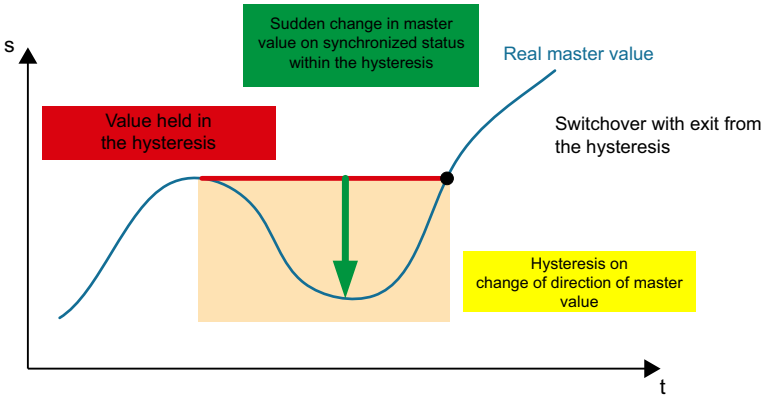


Figure 2-42 Switchover in the hysteresis

To prevent that, as of V4.4 there is a configuration data item **syncingMotion.enableSyncWithinTolerance**.

- **Synchronized status also in hysteresis**
  With the setting **syncingMotion.enableSyncWithinTolerance = YES** (system default setting, compatibility setting), the **synchronized** status can also be reached by the master value within the hysteresis.
- **No synchronized status in hysteresis**
  With the setting **syncingMotion.enableSyncWithinTolerance = NO** (system setting when creating a new synchronous axis), the **synchronized** status is not set for the master value in the hysteresis range.

Toleration of master value velocity changes during synchronization:

The tolerance of the velocity changes of the master value can be set in the configuration data **syncingMotion.maximumOfMasterChange** (default setting: 20%).
If, during synchronization with a synchronization profile based on the master value distance, there is a change to the master value velocity in excess of what has been parameterized in the configuration data element, an error message is output and the synchronization profile is recalculated.
If, during synchronization with a synchronization profile based on dynamic response parameters and with leading synchronization, there is a change to the master value velocity in excess of what has been set in the configuration data element, an error message is generated but the profile is not recalculated.
During synchronization with a synchronization profile based on dynamic response parameters and with trailing synchronization, the configuration data element is not active. A change to the master value velocity produces an immediate response.

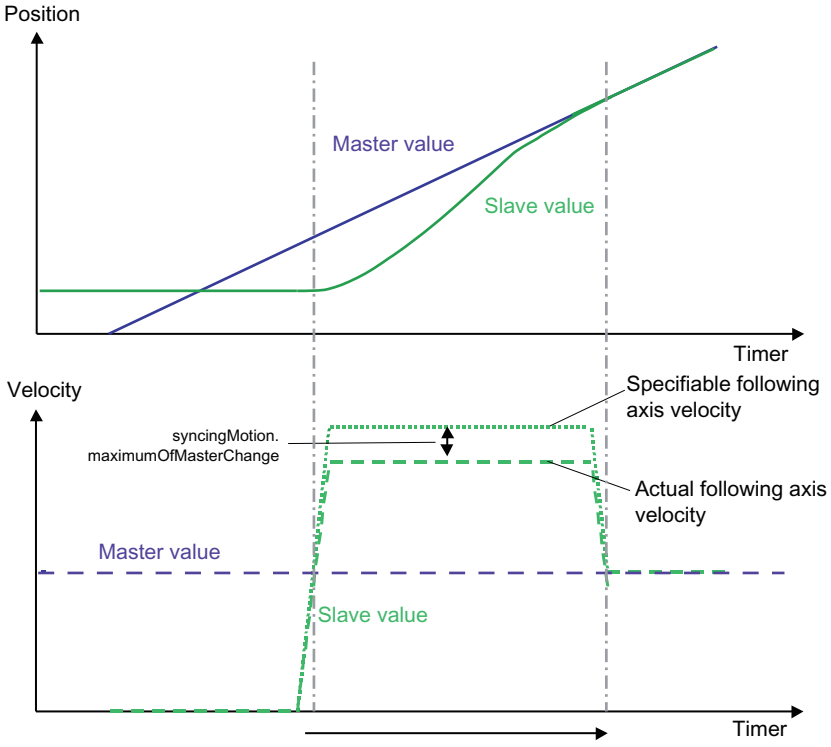


Figure 2-43 Example of syncingMotion.maximumOfMasterChange for leading synchronization with synchronization profile based on specifiable dynamic response parameters

The values of customizable dynamic parameters for synchronization profiles are initially reduced by the values defined in the **syncingMotion.maximumOfMasterChange** configuration data element. The following axis is then accelerated at reduced acceleration to the reduced velocity, in order to maintain reserves and to terminate synchronization safely at the specified synchronization position.

If the master value velocity changes, then the same changes are made to the dynamic response values of the synchronization procedure. Error message "50009 Changing the dynamic response of the master leads to a dynamic violation when synchronizing/ desynchronizing" is issued if the parameterized tolerance is exceeded.

If the direction of the master value reverses during synchronization, the synchronization procedure is aborted with the "50007 Error occurred while activating/deactivating the synchronous operation" error. This does not apply for immediate synchronization or synchronization starting at a defined reference point when **syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME** with **synchronizingMode:=IMMEDIATELY** or **syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME** with **synchronizingMode:=SYNCHRONIZE_WHEN_POSITION_REACHED**.

With trailing synchronization, the **syncingMotion.maximumOfMasterChange** configuration data element is not active, i.e. responses are issued continuously.

2.2.5.7 Monitoring the synchronization

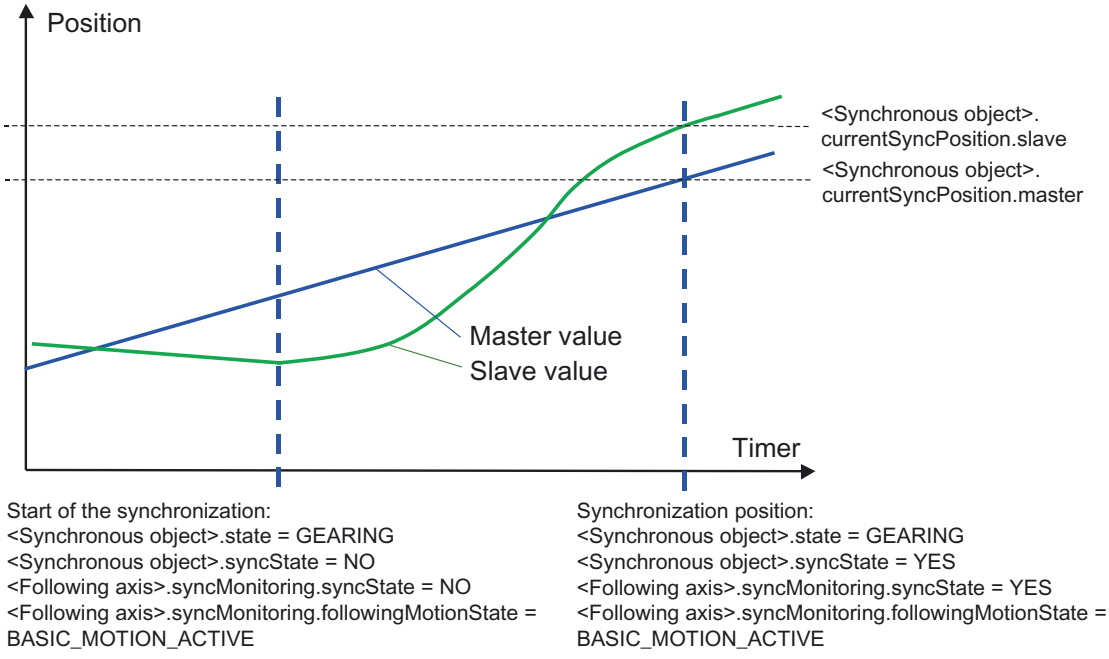


Figure 2-44 System variables for the synchronization

Synchronization state on synchronous object

- The **state** system variable on the synchronous object indicates whether a synchronous operation is active:
  - With state:=**CAMMING** a camming is active.
  - With state:=**GEARING** a gearing is active.
  - With state:=**VELOCITY_GEARING** a velocity gearing is active.
  - With state:=**INACTIVE**, no function is active on the synchronous object.
    When synchronization starts, the system variable is set to the appropriate value and reset again once synchronization is complete.
- The **syncState** system variable on the synchronous object indicates whether the slave value calculated on the synchronous object is synchronous with the master value.
  - If both the master value and slave value are synchronous, this variable will be set to the **YES** state.
    The master value pending on the synchronous object (**currentMasterData.value**) and the slave value output to the following axis (**currentSlaveData.value**) will then be synchronous.
  - The start of desynchronization or any other loss of synchronism causes the variable to be reset to the **NO** value.
    Any restrictions imposed on the slave value transferred by the following axis as a result of the limitation to the maximum dynamic values and the associated non-synchronization of the master and following axis are *not* reflected in the state of the **syncState** variable.
    For the dynamic response limits on the following axis, see Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder Function Manual, "Dynamic limits."
- The master value information on the synchronous operation object does not match the input values from the master value object 1:1, but is instead influenced by:
  - Extrapolation/interpolation for distributed synchronous operation
  - Extrapolation/interpolation through task-wide synchronous operation
  - Extrapolation/interpolation through setpoint output delays
  - Plausibility check at standstill of the master value

  Example:
  If the actual velocity adopts the value zero (because two position values are the same in succession), the filtered value is no longer taken for the actual velocity, but is instead set directly to zero. In the active synchronous operation object therefore 0 is taken for a cycle in currentMasterData.derivedValue. If the (position) setpoint change = 0, the dynamics for preventing reverse procedures are already adjusted to the position change with the setpoint calculation.
- The associated synchronization position of the master and the slave value, i.e. the position from where the master and the slave axis run synchronously are contained in the **currentSyncPosition** system variables on the synchronous object.
  See Display of the synchronization position (Page 61).

- The state of the synchronization can be queried using the **synchronizingState** system variable on the synchronous object (as of V3.2).
  - **WAITING_FOR_SYNC_POSITION**: Waiting for master value synchronization position
  - **WAITING_FOR_CHANGE_OF_MASTER_DIRECTION**: Waiting for master value direction reversal
  - **SYNCHRONIZING_NOT_POSSIBLE**: Synchronization is not possible
  - **SYNCHRONIZING**: Synchronization in progress
  - **INACTIVE**: Synchronization phase is not active
  - **WAITING_FOR_MERGE**: Synchronization command issued, but not yet active
- The execution state of the active command for activation/deactivation is described in the **enableCommand** and **disableCommand** system variables.
  - **INACTIVE** means that no command is configured.
  - **WAITING_FOR_START** means that the command is executed during slave value generation, and that it is waiting for the start criterion for synchronization to be reached.
  - **ACTIVE** means that synchronization is active and that the operation is synchronous.
  - If there are two commands during slave value generation, both system variables can assume a value not equal to **INACTIVE**. If both are enable commands, the state of the current command is displayed (the state of the next command is always **WAITING_FOR_START**).
- The relevant active command parameter and the parameter for synchronization are grouped together and can be read out in the **effectiveData** system variable structure.

Synchronization state on following axis

- The **syncMonitoring.syncState** variable on the following axis indicates the synchronous operation state on the setpoint side.
  During synchronization and desynchronization, syncState:=NO.
- The **syncMonitoring.followingMotionState** variable on the following axis indicates the state of the synchronous motions:
  - **INACTIVE**: Synchronous motion is not active
  - **BASIC_MOTION_ACTIVE**: Synchronous operation is active as main motion
  - **SUPERIMPOSED_MOTION_ACTIVE**: Synchronous operation is active as superimposed motion
  - **BASIC_AND_SUPERIMPOSED_MOTION_ACTIVE**: Synchronous operation is active as main and superimposed motion

2.2.5.8 Display of the synchronous position

The **currentSyncPosition** system variables on the synchronous object indicate the last calculated synchronous position of a synchronous operation.

- **currentSyncPosition.master**: Synchronous position of the master value
- **currentSyncPosition.slave**: Synchronous position of the following axis

These values are valid only when "syncState = YES."

Start position of the cam on the axis

The master value and slave value at the cam start of the current camming operation are shown in system variables (V4.0 and higher). The values can also be shown when the starting point of the synchronous operation lies within the cam.

- **currentSyncPosition.camMasterMatchPosition**:
  Master value at the cam start
- **currentSyncPosition.camSlaveMatchPosition**:
  Slave value at the cam start
- **currentSyncPosition.distanceCamMasterMatchPostion**:
  Current relative position in the cam (distance to the cam start)
  Application: Calculation of the corresponding axis positions, also for camming, e.g. a desynchronization position. The axis position must be specified absolute in relation to the axis, even for relative synchronous operation.
  These system variables, can be used, for example, to determine the exact position of the cam for the axis, even for relative camming, and to specify the desynchronization position relative to the axis.

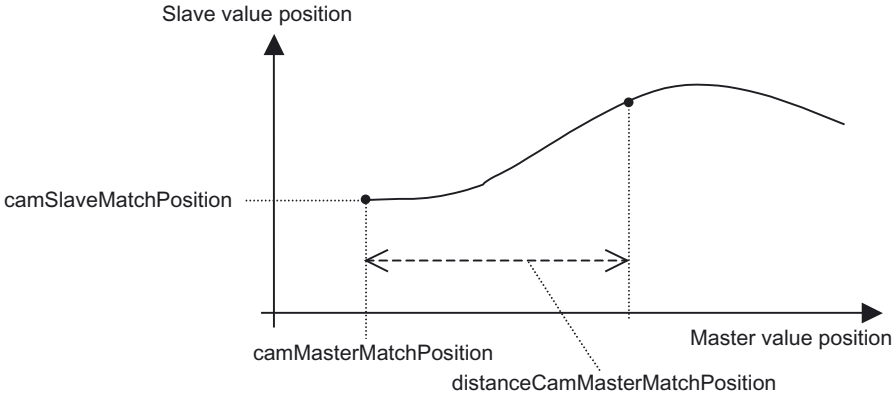


Figure 2-45 Display of the master value and slave value positions in the currentSyncPosition system variable

Position reproduction for modulo axes with gearing

The **currentSyncPosition.slavePositionAtMasterModuloStart** system variable can be used to read out the slave value position at the modulo starting point of the master value (V4.0 and higher).

- **currentSyncPosition.slavePositionAtMasterModuloStart**:
  Slave value position at the modulo starting point of the master value **currentSyncPosition.slave** will be indicated if no modulo is active at the master value.

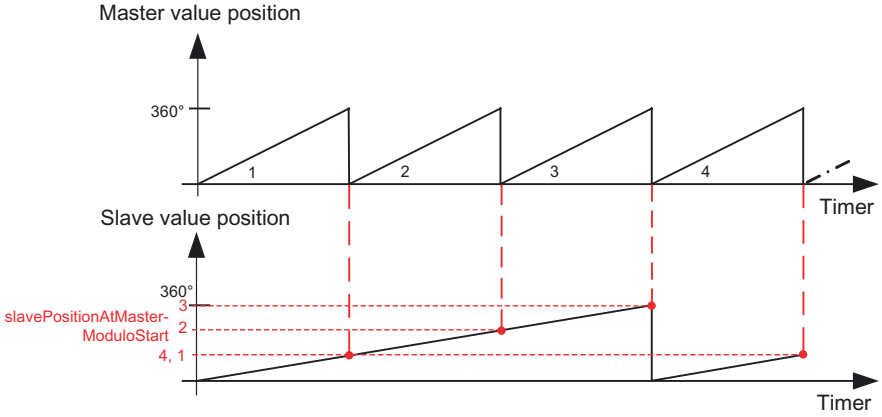


Figure 2-46 Position difference caused by a different modulo starting point

Application

For known gear ratio and known modulo lengths, the assignment of master value and slave value can be closed, even when the slave value modulo length does not correspond to the master value modulo length.

2.2.5.9 "Synchronous" status during synchronization

- With synchronization via a specifiable master value distance, the "synchronous" status is achieved at the end of the synchronization distance.
- The "synchronous" state of synchronization with a synchronization profile based on customizable dynamic response parameters and leading synchronization is achieved when the synchronization position is reached. (In this case, this is the same as the synchronous position.)
  At the synchronous point, synchronism is present in the position, velocity, and acceleration (only with **SMOOTH** velocity profile).
- With synchronization using a synchronization profile based on specifiable dynamic response parameters and with trailing synchronization, the "synchronous" status is achieved when synchronism exists in the position, velocity, and acceleration (only with profile **SMOOTH** and **syncingMotion.smoothAbsoluteSynchronization:=YES**) in accordance with the transmission function.

With relative gearing without offset, the position is not evaluated during synchronization.

2.2.6 Desynchronization

2.2.6.1 Desynchronization - Overview

Desynchronization refers to the termination of the synchronous operation.

The desynchronization is defined by several parameters/settings:

- **Desynchronization criterion/desynchronization position**
- **Position of synchronization range relative to desynchronization position**
- the **desynchronization profile**
  - **Desynchronization over a specifiable master value distance**
    The desynchronization length is specified in the desynchronization command.
  - **Desynchronization profile via specifiable dynamic response parameters**
    The dynamic response parameters are specified in the desynchronization command.

2.2.6.2 Desynchronization criterion/desynchronization position

The desynchronization criterion/desynchronization position is specified in **syncOffMode**.

- Desynchronization at the current master value position; immediate desynchronization
  Desynchronization at the current master value position is set in **syncOffMode:=IMMEDIATELY**. Only trailing desynchronization can occur. The setting in the **syncOffPositionReference** parameter is not active. The **syncOffPositionMaster** parameter is not active. The **syncOffPositionSlave** parameter is not active.
- Desynchronization at a specified master value position
  Desynchronization at a specifiable master value position is set in **syncOffMode:=ON_MASTER_POSITION**. The **syncOffPositionReference** parameter can be used to set leading, symmetrical (only with desynchronization via master value distance), and trailing desynchronization. The desynchronization position on the master value side is set in the **syncOffPositionMaster** parameter. The **syncOffPositionSlave** parameter is not active.
- Desynchronization at a specified slave value position
  Desynchronization at a specifiable slave value position is set in **syncOffMode:=ON_SLAVE_POSITION**. The **syncOffPositionReference** parameter can be used to set leading, symmetrical (only with desynchronization via master value distance), and trailing desynchronization. The desynchronization position on the slave value side is specified in the **syncOffPositionSlave** parameter. The **syncOffPositionMaster** parameter is not active.
- Desynchronization at the end of cam cycle
  Desynchronization at the end of the cam cycle is set in **syncOffMode:=AT_THE_END_OF_CAM_CYCLE**. The **syncOffPositionMaster** parameter is not active. The **syncOffPositionSlave** parameter is not active.

2.2.6.3 Desynchronization over a specifiable master value distance

Desynchronization via a specifiable master value distance is set in the **syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE** parameter. The slave values travel to zero velocity while the master value travels through the desynchronization length. The desynchronization length is specified in the **syncOffLength** parameter.

See also Synchronization over a specifiable master value distance (Page 49).

2.2.6.4 Desynchronization profile via specifiable dynamic response parameters

Desynchronization based on specifiable dynamic response parameters is set in the **syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME** parameter. The slave values travel to zero velocity with the dynamic response values specified in the desynchronization command according to the specified desynchronization criterion.

See also Synchronization profile based on specifiable dynamic response parameters (time reference) (Page 52).

2.2.6.5 Position of synchronization range relative to desynchronization position

The position of the synchronization range relative to the desynchronization position can be specified more precisely with the **syncPositionReference** parameter of the **_disableGearing()** or **_disableCamming()** command:

- Desynchronization before the specified desynchronization position
  Desynchronization before the specified desynchronization position is set using the **syncOffPositionReference:=AXIS_STOPPED_AT_POSITION** parameter. The slave value travels to zero velocity to the specified desynchronization position.
- Desynchronization starting from the specified desynchronization position
  Desynchronization starting from the specified desynchronization position is set using the **syncOffPositionReference:=BEGIN_TO_STOP_WHEN_POSITION_REACHED** parameter. The slave value travels to zero velocity starting from the specified desynchronization position.
- Symmetrical desynchronization relative to the specified desynchronization position
  Symmetrical desynchronization relative to the specified desynchronization position is set with the **syncOffPositionreference:= STOP_SYMMETRIC_WITH_POSITION** parameter. The slave value travels at zero velocity symmetrically to the specified desynchronization position. The setting is not possible for the desynchronization profile based on dynamic response parameters.

2.2.6.6 Replacement of an active synchronous operation

If a synchronous operation is active, it can be replaced by another synchronous operation only when the synchronization criterion of the following synchronous operation can be maintained.

Example:

- For an active camming, the following axis travels only in the negative direction.
- A **_enableCamming()** command with **synchronizingDirection:=POSITIVE_DIRECTION** is issued at the end of the cam cycle (or the criterion, for example, the specified master distance ensures this).

In this case, the first active camming is not ended: the system waits until the following axis travels in the positive direction, which never occurs because of synchronous operation is active. To prevent the described behavior, change the synchronization criterion appropriately or end the active synchronous operation first (using a **_disable** command) and then activate the new synchronous operation.

Replacement of a synchronous operation with small synchronization length

If a synchronous operation group is ended using a **_disable** command with a very short desynchronization length, high dynamic response values result that must be replaced using discontinuous acceleration. If the **_disable** command is replaced before its ending by a new motion command with continuous velocity profile, the still-present high acceleration values are replaced prior to processing the added command. This results in a longer traversal distance of the axis that can also cause the reversing of the axis.

In cases in which an immediate replacement of the synchronous operation is necessary, for the described reason, no **_disable** command should be used, but rather an immediate switch made to the replacing motion command. In this case, the jerk for the following command may need to be increased.

If no **_disable** command is necessary, the subsequent move/pos command for a continuous velocity command should have high dynamic response values or traverse on a "trapezoidal profile."

Substituting motions

A motion that is in an acceleration or deceleration phase is replaced by a motion with jerk limitation (**smooth**). Correspondingly long times can result from the reduced jerk until the current acceleration is reduced.

For an actual value coupling, the result is essentially a noisy signal, depending on the encoder quality. In the case of substitution with jerk limiting (**smooth**), the slave movement is continued relative to this noise-prone signal (acceleration). The system interprets the signal as if the default value is in an acceleration or braking phase and reduces it via the jerk limitation.

With the optional **abortAcceleration** parameter, the functions **_stop** and **_stopEmergency** can be set so that any acceleration is reduced immediately and not via the jerk limitation.

2.2.7 Dynamic response effect on slave values

The dynamic response of the slave values is determined from:

- The dynamic response of the master value
- The dynamic response of any master value switchover occurring during the motion
- The dynamic response of the synchronization
- The dynamic response resulting from the transmission function

- If necessary, the dynamic response resulting from the application of offsets and scaling changes
- The limitation of the slave value dynamic response to the maximum values of the following axis

The dynamic response specifications on the synchronous object refer to the slave values calculated on the synchronous object during synchronization. The dynamic limits of the following axis are not taken into account by the synchronous object.

To avoid excessive dynamic response specifications on the slave values,

- The slave values calculated from the the master value based on the transmission function should not exceed the dynamic limits
- The dynamic response specifications for synchronization and master value switchover should not exceed the dynamic limits

The resulting dynamic values are limited to the maximum values on the following axis based on the axis configuration.

The individual dynamic response parameters that are active during synchronous operation are illustrated in the figure below.

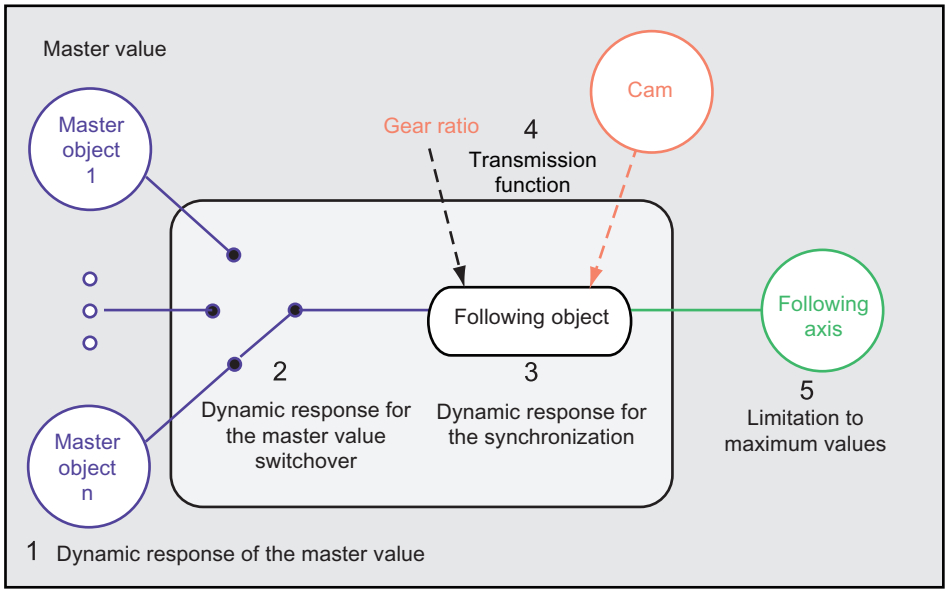


Figure 2-47 Dynamic response during synchronous operation

Legend:

1. The dynamic response of the master value is determined by the motion.
2. The dynamic response of the master value switchover can be specified using the **_setMaster()** command.

3. Synchronization/desynchronization and compensation:
   - *Without* dynamic specification during synchronization via a specifiable master value distance **(RELATE_SYNC_PROFILE_TO_LEADING_VALUE)**
   - Synchronization profile based on specifiable dynamic response parameters (time reference) **(RELATE_SYNC_PROFILE_TO_TIME)**
     The dynamic values set on the synchronous object are only valid for camming or gearing *during synchronization/desynchronization* and when applying corrections, but not in "synchronous" status. See Synchronization (Page 40), Desynchronization (Page 64).
4. The dynamic response specification for the following axis is determined by the synchronous object and the gear ratio. The synchronous object is *not* subject to any dynamic limitation in the 'synchronous' status.
5. The slave setpoints on the following axis are limited to the maximum axial dynamic response.
   Configuration data: **TypeOfAxis.MaxAcceleration/MaxVelocity/MaxJerk**
   System variables: **plusLimitsOfDynamics/minusLimitsOfDynamics**. The lowest limit is taken into account in each case.
   See the **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Dynamic limits."
   The maximum axial jerk is only used to monitor and, when appropriate, limit the synchronous operation setpoints if synchronous operation monitoring has been activated with provision made for jerk. The setting for synchronous operation monitoring has no effect on the synchronous operation setpoints generated. This even applies for the synchronization procedure, for example.
   The alarm "40202 Dynamic response of the synchronous operation setpoint cannot be achieved" is output if the slave setpoints calculated by the synchronous operation are higher than the active axial limits for velocity and acceleration. If the slave values are limited as a result of this or due to dynamic discontinuity of slave values for synchronous operation (caused by master value jumps, for example), a setpoint error is generated in the slave values. See Synchronous operation monitoring (Page 75).
   The maximum jerk on the axis can be exceeded during synchronization and desynchronization if the jerk setting in the synchronization parameters on the synchronous object is greater than the maximum jerk on the axis. To prevent this, an alarm response can be configured, for example.
6. If desynchronization occurs within one system clock cycle and the target dynamics are therefore zero in respect of velocity and acceleration when the next clock cycle starts, no alarm will be output.

2.2.8 Switching of the master value source

2.2.8.1 Switching over the master value source – Overview

If more than one master value is assigned to a synchronized axis, the master value source can be selected and switched over on the synchronous object using the **_setMaster** command.

If a synchronous object is assigned to multiple master values, a random master value source is selected internally following system startup. The correct master value source must be specified in the user program. The master value source can be switched "on-the-fly". When it

is enabled, the master values are referenced to the units system of the current master value source. A relative or absolute coupling influences the transition process.

The master value transition can be set with and without dynamic response using the **transientBehavior** parameter of the **_setMaster** command (V3.2 and higher):

- **DIRECT**: Without dynamic response (default)
- **WITH_DYNAMICS**: With dynamic response
- **WITH_NEXT_SYNCHRONIZING**: With next synchronization (as of V4.1)

See also

Master value switchover without dynamic response (Page 69)

Master value switchover with dynamic response (Page 69)

Master value switchover with next synchronization (V4.1 and higher) (Page 70)

2.2.8.2 Master value switchover without dynamic response

The transition behavior when the master value source is changed is different for absolute synchronous operation and relative synchronous operation.

- With relative synchronous operation, an additional slave value difference occurs only if the dynamic master values are different with regard to velocity and acceleration.
- With absolute synchronous operation, a non-continuous master value transition can occur. Discontinuities in the slave values are limited to the maximum dynamic axis parameters on the following axis. A compensation movement is generated in certain circumstances.

Different modulo settings of the master value sources are taken into account.

2.2.8.3 Master value switchover with dynamic response

The dynamic response parameters: The velocity profile, velocity, acceleration, and, if necessary, jerk can be specified directly in the **_setMaster()** command. These parameters refer to the dynamic response of the transition of the master value source. The **setMasterCommand** system variable indicates the status of the **_setMaster()** motion on the synchronous object.

Note

If the **_setMaster** command switches over the master value, the output of the synchronous object remains asynchronous to the new master value during the transition. The system variables for the synchronization remain unaffected. The transition behavior of the master value does not have any effect on the active gearing/camming.

Please note that a master value switchover does not constitute a new synchronization procedure, i.e. the syncState system variable (on the synchronous object) indicates YES.

To ensure setpoint synchronism, the **setMasterCommand** and **syncState** system variables must be monitored.

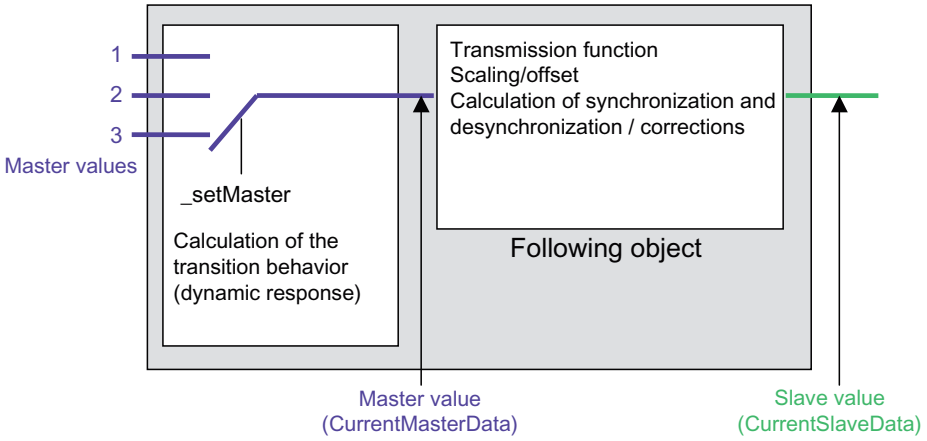


Figure 2-48 Transition behavior and master value for the master value switching with dynamic response

The transition behavior for the new master value is calculated separately from the synchronization and desynchronization, and until the end of the compensation is used as master value for the setpoint monitoring and the evaluation of the **syncState, synchronizingState** synchronization status.

The comparison of this value with the output value of the synchronous object sets the **syncState** and **synchronizingState** variables: **syncState=YES** and **synchronizingState=INACTIVE**. Despite the switching, the **differenceCommandValue** setpoint difference is zero.

2.2.8.4 Master value switchover with next synchronization (V4.1 and higher)

The master value switchover is active together with the next **_enableCamming()/ _enableGearing()** synchronization command, whereby all specifications refer to the new master value. The dynamic response values in the **setMaster()** command are not active because the dynamic response values of the synchronization command are active during synchronization.

System variable **stateSetMasterCommand** indicates the current status. Parameter **transientBehaviour** sets the master value switchover:

- Master value switchover: is not active: system variable **stateSetMasterCommand = INACTIVE ,** parameter **transientBehaviour = INACTIVE**
- Master value switchover is active, switchover occurs directly: system variable **stateSetMasterCommand = TRANSIENT_BAHAVIOR_DIRECT,** parameter **transientBehaviour = DIRECT**
- Master value switchover is active, switchover occurs with dynamic response values: system variable **stateSetMasterCommand = TRANSIENT_BAHAVIOR_WITH_DYNAMICS ,** parameter **transientBehaviour = WITH_DYNAMICS**
- Master value switchover is active; switchover occurs with next synchronization: system variable **stateSetMasterCommand = TRANSIENT_BAHAVIOR_WITH_NEXT_SYNC,** parameter **transientBehaviour = WITH_NEXT_SYNCRONIZING)**

2.2.9 Superimposed synchronous operation

Description

Two synchronous operations can be *superimposed* by creating another synchronous object on an axis (as of V3.0).

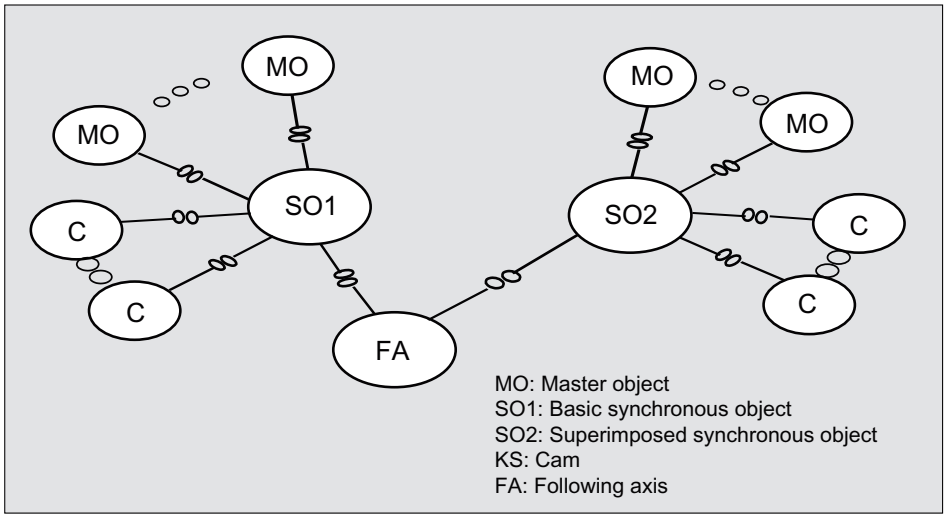


Figure 2-49 Schematic of a superimposed synchronous operation

To differentiate from the simple synchronous operation, the first synchronous operation is called a *basic synchronous operation* and the second is called a *superimposing synchronous operation*. The synchronous objects are called accordingly *basic synchronous object* and *superimposing synchronous object*.

The synchronous operation can be set on the synchronous object to act as the basic motion, main motion (has the same effect as non-superimposed synchronous operation), or superimposed (or secondary) motion (**syncingMotion**.**motionImpact:=STANDARD**/ **SUPERIMPOSED_MOTION** configuration data element).

A maximum of one basic synchronous object can be interconnected to an axis, plus one superimposed synchronous object on the same axis.

Creating an axis with superimposed synchronous operation

In the project navigator under <**Axis_n**>, it is possible to insert a maximum of one further synchronous object <**Axis_n_SYNCHRONOUS_OPERATION_1**>, which is then superimposed, i.e. configuration data **motionImpact** is initialized to **SUPERIMPOSED_MOTION**.

1. Select the axis in the project navigator.
2. Select **Expert > Superimposed synchronous object** from the shortcut menu.

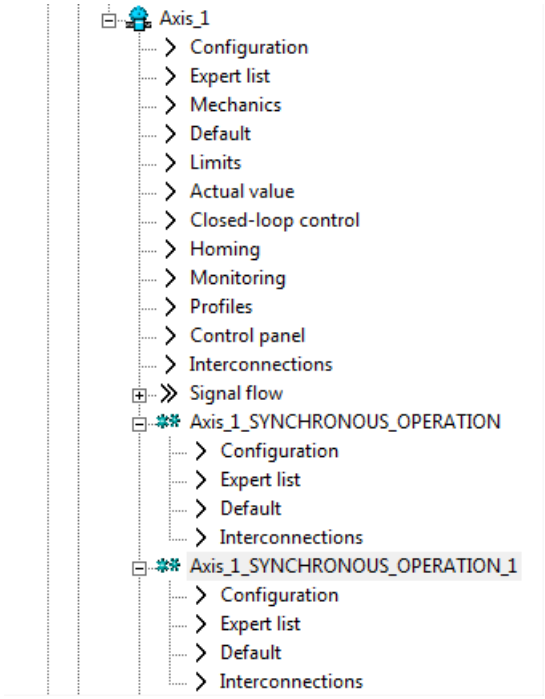


Figure 2-50 Representation of superimposed synchronous operation in the project navigator

The configuration and settings for superimposed synchronous operation are made in the same way as for the basic synchronous object.

Programming

All functions of the basic synchronous object (e.g. _enableGearing, _disableGearing, etc.) can be applied on the superimposed synchronous object. Cross-references between synchronous objects are not possible.

Absolute or relative superimposed synchronous operation

With superimposed synchronous operation, the properties for absolute or relative are the same as for basic synchronous operation, with the exception that the coordinates refer to the superimposed coordinate system on the following axis.

Coordinates

For the basic synchronous object, the synchronization parameters specified for the slave value position refer to:

- The total coordinate system with **mergeMode:**=**IMMEDIATELY** and **TypeOfAxis.decodingConfig**.**transferSuperimposedPosition** <> **TRANSFER_RESET**
- The basic coordinate system in all other cases

The superimposed synchronous object refers to the superimposed coordinates where slave value position specifications have been given.

$$\begin{pmatrix} s \\ v \\ a \end{pmatrix}_{\substack{\text{Cumulative} \\ \text{coordinates}}} = \begin{pmatrix} s \\ v \\ a \end{pmatrix}_{\substack{\text{Basic} \\ \text{coordinates}}} + \begin{pmatrix} s \\ v \\ a \end{pmatrix}_{\substack{\text{Superimposed} \\ \text{coordinates}}}$$

Figure 2-51 Coordinates for superimposed synchronous operation

Just as with a superimposed motion, each synchronous object has its own coordinate system. The "outputs" of the synchronous objects are added in the Slave Axis technology object. If, for example, the basic synchronous operation and superimposed synchronous operation have the same master value, and motion takes place in absolute gearing with a ratio of 1:1, the position value of the following axis will be twice that of the master axis after both synchronous objects have been synchronized.

Behavior of superimposed synchronous operation in relation to basic motion

Superimposed synchronous operation behaves like a superimposed positioning motion with regard to the basic motion on the axis (motion or synchronous operation). **TypeOfAxis.decodingConfig**.**transferSuperimposedPosition** configuration data on the synchronous axis set when superimposed motions will be applied to the basic motion, and when they will be substituted. This setting determines, for example, when **mergeMode**=**IMMEDIATELY** that the superimposed motion is to be substituted on the basic motion. These configuration data settings also apply to superimposed synchronous operations (see also here (Page 138)).

There can be only *one* superimposed motion on the axis at one time, for example, a **superimposed positioning motion** *or***superimposed synchronous operation**. A **superimposed synchronous operation** can be active without a basic motion being active at the same time.

See also superimposed motion for axis, **Motion Control Technology Objects Axis Electric/ Hydraulic, External Encoder** Function Manual.

Monitoring

The output values of a synchronous object (and thus also the motion component of the superimposed synchronous operation for the axis) can be read out in the **currentSlaveData** system variable on the synchronous object.

Table 2-1 Coordinates on the following axis for the superimposed synchronous operation

	System variable	Description
All coordinates:		
positioningState.	**commandPosition**	Target position (total)
motionStateData.	**commandVelocity**	Target velocity (total)
	commandAcceleration	Target acceleration (total)
Basic coordinates:		
basicMotion.	**position**	Position in the basic coordinate system
	velocity	Velocity in the basic coordinate system
	acceleration	Acceleration in the basic coordinate system
Superimposed coordinates:		
superimposedMotion.	**position**	Position in the superimposed coordinate system
	velocity	Velocity in the superimposed coordinate system
	acceleration	Acceleration in the superimposed coordinate system

The **syncMonitoring** system variable on the following axis also displays the state of the synchronous operation motion (as of V3.0):

- **followingMotionState =**
  - **INACTIVE**: Synchronous motion is not active
  - **BASIC_MOTION_ACTIVE**: Standard synchronous operation is active
  - **SUPERIMPOSED_MOTION_ACTIVE**: Superimposed synchronous operation is active
  - **BASIC_AND_SUPERIMPOSED_MOTION_ACTIVE**: Standard synchronous operation and superimposed synchronous operation are active

Compensations during distributed synchronous operation

Compensation during distributed synchronous operation is also useful/effective during superimposed synchronous operation.

See Compensations for distributed synchronous operation (Page 157).

Synchronous operation monitoring / state

- The following table describes the conditions that apply when the state is synchronous (**syncMonitoring.syncstate = YES**):

Basic synchronous operation	Superimposed synchronous operation
Not sychronous (**synchState = No**)	Synchronous (**synchState = Yes**)
Basic synchronous operation synchronous (**synchState = Yes**)	Not synchronous (**synchState = No**)

- The Not synchronous state (**syncMonitoring.syncstate = No**)only applies during the synchronization/desynchronization process (**synchronizingState <> Inactive**)

Example:
A synchronous operation is started. After synchronization, the synchronous operation is assigned the 'synchronized' state. Now another synchronous operation is started. The 'synchronized' state disappears for the duration of the synchronization and is not reset until the synchronization of the second synchronous operation is complete.

The variables and monitoring on the axis refer to the total synchronous operation. Error messages (synchronous operation errors on the synchronized axis) are issued on all interconnected synchronous objects.

2.2.10 Synchronous operation monitoring

The slave values calculated by the synchronous object (**currentSlaveData**) and any additional setpoints on the axis are limited on the following axis to the maximum dynamic response values. The deviation in the slave values resulting from this limitation is monitored. The current maximum limits for velocity and acceleration (and jerk) on the axis influence this monitoring process.

See the **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Dynamic limits".

The **syncMonitoring** system variables on the following axis indicate setpoint and actual value differences:

- **Setpoint monitoring**
  differenceCommandValue shows the difference between the setpoint generated at the synchronous object and the executable setpoint at the axis, with dynamic limits taken into account. **limitCommandValue** shows that the difference between the calculated slave value and executable setpoint is above the permissible tolerance.
- **Actual value monitoring**
  differenceActualValue contains the sum from **differenceCommandValue** (the difference between the setpoint generated at the synchronous object and the executable setpoint at the axis, with dynamic limits taken into account) and the position lag at the following axis. **limitCommandValue** shows that the difference between the calculated slave value and the actual value lies above the permissible tolerance.

Setpoint monitoring

On the axis, the slave values calculated by the synchronous operation are limited to the maximum dynamic response values of the axes. This can cause the setpoints on the axis to change.

Consequently, the setpoint error is the difference between the calculated setpoint of the synchronous object (**currentSlaveData**) and the associated setpoint at the following axis that results from observing the acting dynamic maximum limits of the following axis (**basicMotion.position**).

The setpoint error of one of the motions possibly superimposed to the synchronous operation is the difference between the calculated setpoint of the synchronous object (**currentSlaveData**) and the superimposed setpoint at the following axis that results from observing the acting dynamic maximum limits of the following axis (**superimposedMotion.position**).

The setpoint error from the main and superimposed motion is displayed in **syncMonitoring.differenceCommandValue** on the following axis.

The **syncMonitoring.syncState** synchronization state on the following axis is set according to the **syncState** on the synchronous object. Exception: Superimposed synchronous operation Superimposed synchronous operation (Page 71).

The following comparison must be performed to determine whether the following axis is synchronized on the setpoint value side:

(<following axis>.**syncMonitoring.syncState:=YES) AND**

(<following axis>.**syncMonitoring.differenceCommandValue = 0)**

Actual value monitoring

The **syncMonitoring.differenceActualValue** system variable allows the position lag present at the following axis, including any acting dynamic response limitation of the following axis (setpoint error in **syncMonitoring.differenceCommandValue**), to be queried. The position lag for DSC operation applies to the position lag present at the position controller in the drive.

For versions earlier than V4.2, the difference is formed between the setpoint calculated on the synchronous object and the actual value present in the interpolator.

Velocity gearing monitoring

The **syncMonitoring** system variable on the following axis also displays the status of the velocity gearing (as of V3.1):

- **differenceCommandVelocity**: velocity difference on the setpoint side between the velocity setpoint calculated on the synchronous object (**currentSlaveData**) and the executable velocity on the following axis (only applies to synchronous velocity operation).
- **differenceActualVelocity**: velocity difference on the actual value side between the velocity setpoint calculated on the synchronous object (**currentSlaveData**) and the executable velocity on the following axis (only applies to velocity gearing).

Configuration

The synchronous operation monitoring is set on the following axis under **Monitoring - Synchronous operation monitoring** (**GearingPosTolerance** configuration data element).

Limiting and monitoring the setpoint error:

- With setting **enableCommandValue := NO_ACTIVATE**:
  - the tolerance monitoring of the setpoints is not activated.
- With setting **enableCommandValue := WITHOUT_JERK**:
  - the tolerance monitoring of the setpoints is activated without inclusion of the jerk. Alarm 40201 is output if the setpoint tolerance is exceeded.
- With setting **enableCommandValue := WITH_JERK**:
  - the tolerance monitoring of the setpoints is activated with inclusion of the jerk. Alarm 40201 is output if the setpoint tolerance is exceeded. The jerk on the axis is also monitored.

Note

For distributed synchronous operation with extrapolation on the following axis, the setpoint monitoring with jerk setting is not appropriate.

The actual value deviation monitoring is set via **GearingPosTolerance.enableActualValue**. The synchronous operation monitoring setting only becomes active after synchronization is complete **(syncState = YES).**

For the dynamic response limits on the following axis, see the Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder Function Manual, "Dynamic limits".

Error handling

When the synchronous operation tolerance is exceeded, the following axis issues the technological alarm "40201 Synchronous operation tolerance exceeded on the following axis". An alarm message can also be sent to the master value source; this setting is made in the **TypeOfAxis.GearingPosTolerance.enableErrorReporting** configuration data element. Here, a distinction can be made between deviation tolerance violations of the calculated slave value setpoint and of the calculated slave value actual axis value. The leading axis then issues the error "40110 Error triggered on slave during synchronous operation (error number: ...)".

As of V4.2, an error message is also output on the master object when the following axis disconnects the synchronous coupling for any reason in the event of an error.

Note

If the **ALL_ERRORS_WITH_ABORT_SYNCHRONIZATION** parameter is set, error message 40110 is triggered on the relevant master object (e.g. also with following error on following axis).

If the synchronous motion is interrupted by a substitutional motion, no error message is generated.

Only following axis errors are reported to the master object. Errors occurring on the synchronous object on command output and/or synchronization are not taken into account.

The troubleshooting can be set using the following configuration data element:

- <Following axis>.TypeOfAxis.GearingPosTolerance.enableErrorReporting
  - **NO_REPORTING**- (default) no message, present
  - **COMMAND_VALUE_TOLERANCE**- setpoint monitoring, present
  - **ACTUAL_VALUE_TOLERANCE**- actual value monitoring, present
  - **ALL_ERRORS_WITH_ABORT_SYNCHRONIZATION**- all errors that occurred

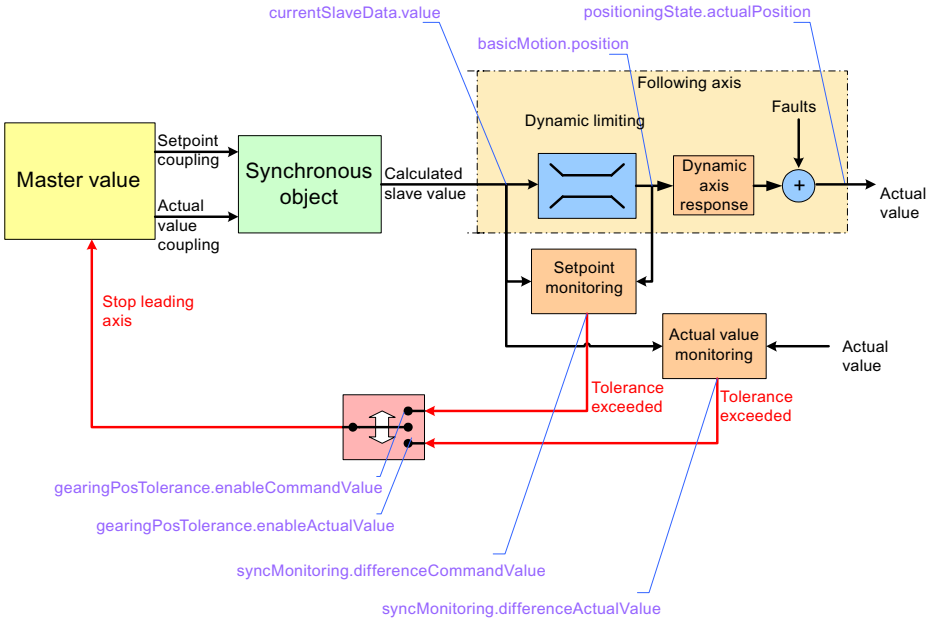


Figure 2-52 Interrelationship between synchronous operation monitoring functions

See Error handling in the user program (Page 145).

See also

Monitoring the synchronization (Page 59)

Commands for command tracking (Page 136)

2.2.11 Simulation mode

Synchronous operation can be switched to *simulation*, in which the values at the synchronous object are calculated but not output to the following axis. The synchronous operation simulation can be activated and deactivated at any time provided there are no faults present.
The **simulation** [ACTIVE/INACTIVE] system variable provides information about the simulation status of the axis.

Application: Retaining a synchronous operation connection with _disableAxis() (Page 105).

Commands for the simulation operation

- The **_enableFollowingObjectSimulation()** command sets a synchronous operation into simulation mode. The synchronous values are calculated, but are not output to the following axis. This can be done at any time. However, the axis states are taken into account in generating the slave values.
- The **_disableFollowingObjectSimulation()** command resets the synchronous operation relationship out of simulation mode. The synchronization values are output to the following axis again.
  If there is a difference between the setpoint calculated in synchronous operation and the setpoint present at the axis, or superimposed motion has occurred, only dynamic limitation to the maximum values of the following axis is performed.

Configuration data for the simulation operation

The **disableSynchronousOperation** configuration data can be used to set whether master values are to be forwarded to the slave axis.

- If **NO** (default), the synchronous operation is also canceled in simulation mode, provided that the enables on the following axis have been canceled.
- If **YES**, the synchronous operation is not canceled in simulation mode if the enables on the following axis have been canceled while synchronous operation is in simulation mode. Any synchronous operation commands that are undergoing execution are retained.

2.2.12 Configuring units

You can define the basic units for each technology object. The same physical variables can have different units in different technology objects. These are converted:

How to configure the units:

1. In the project navigator, open the context menu under the TO.
2. In the context menu, select Expert > Configure units. The Configure Units window appears in the working area.
3. Select the unit for the physical variables. These units are used for the technology object, for example, s for units of time.

or

1. In the project navigator, open the Configuration under the TO.
2. Select the "Units" tab.

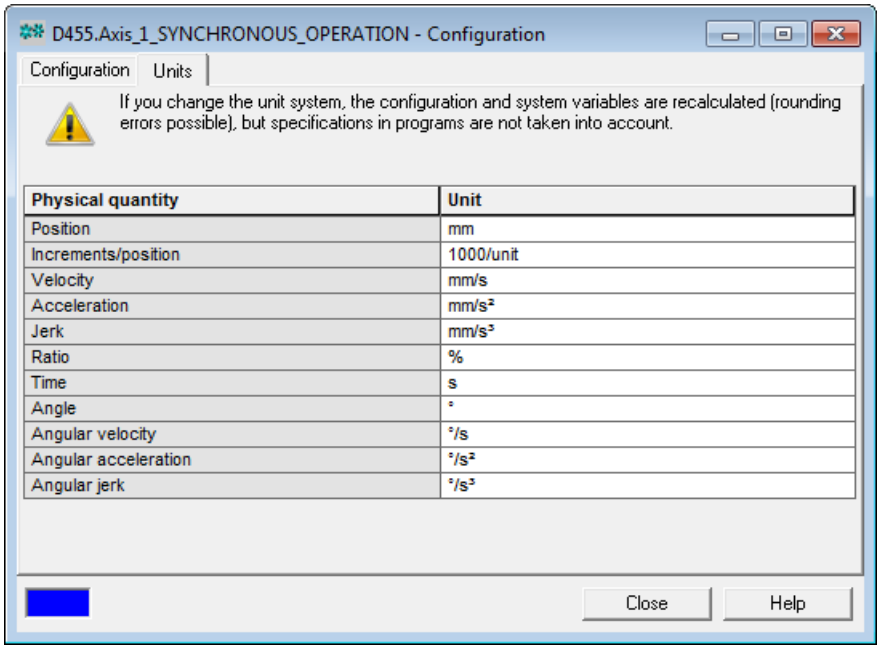


Figure 2-53 Synchronous operation units

You can set the following parameters:

Field/Button	Meaning/Instruction
Table with units	
Physical variable column	Shows the physical variable. The physical variables which are used by the TO are available for the configuration.
Unit column	Displays and configures the unit. Clicking on the cell opens up a combo box for selecting the unit.
Toolbar	
	• Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display
Close	Button for closing the dialog.
Help	Button for opening the online help for the dialog.

2.2.13 Examples of synchronization operations as a function of the output position on the slave value side

2.2.13.1 Synchronization via a specifiable master value distance

Together with the master value behavior, the starting position of the following axis relative to the synchronous position on the following axis side largely determines the characteristic of the synchronization profile.

Influence of starting position of the following axis

Example:

- Absolute 1:1 gear without offset
- Constant master value velocity
- Slave value at start of synchronization at standstill

- Synchronization over a master value distance
- Velocity profile type **CONTINUOUS** for synchronization

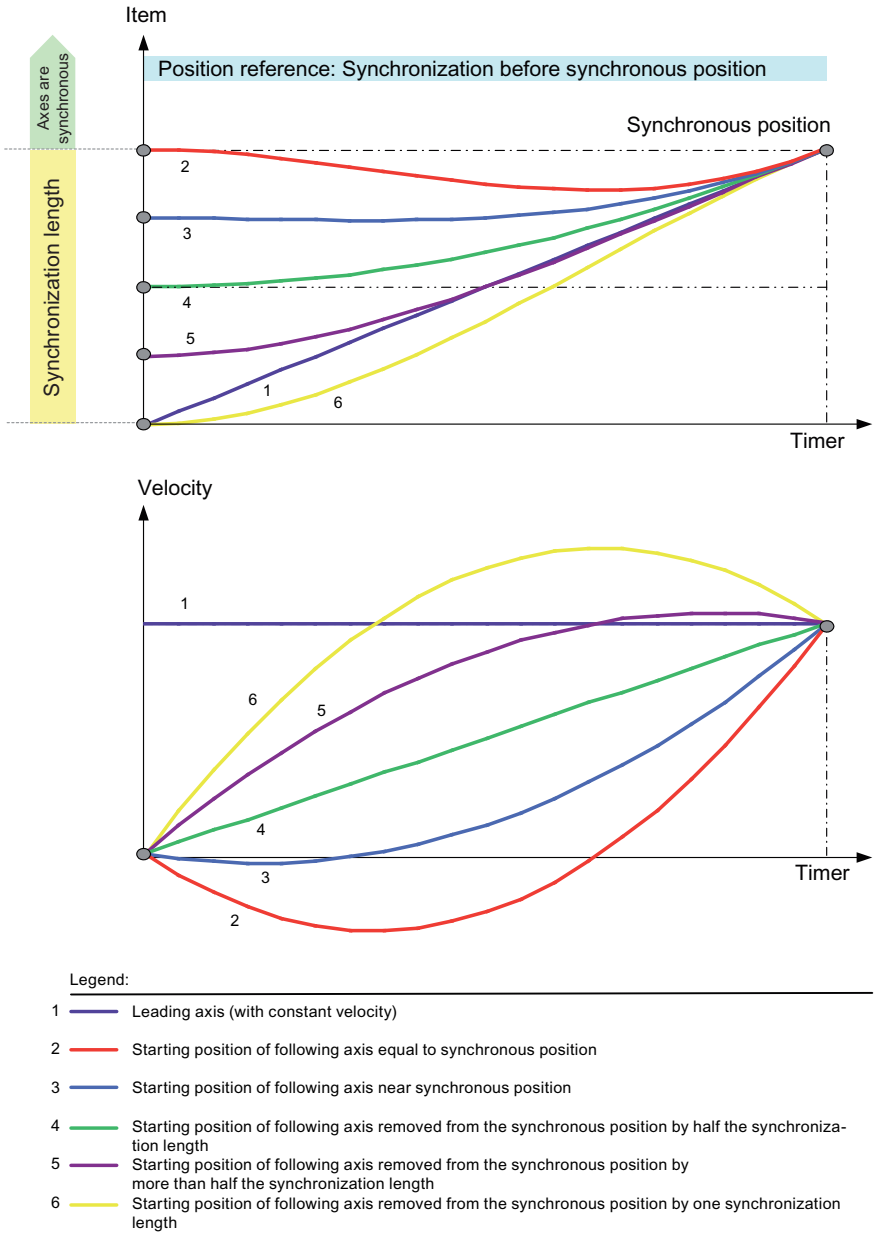


Figure 2-54 Synchronization via a specifiable master value distance

While the position characteristics of the individual synchronization operations do not differ very much, the velocity characteristics differ significantly:

- To accelerate to the synchronous velocity at the synchronous position, motion in the opposite direction following by reversal is required (2).
- It is possible to accelerate directly to the synchronous velocity and synchronous position; the position difference to be applied is minor (3).
- The acceleration to the synchronous velocity in the synchronous position is even (4).

- Direct acceleration to the synchronous velocity in the synchronous position is possible. While the position difference to be applied is greater, a velocity greater than the synchronous velocity and, thus, a velocity reversal is not required for synchronization (5).
- To apply the position difference, a velocity greater than the synchronous velocity and, thus, a velocity reversal is required for synchronization (6).

Recommendation for master value-related synchronization

Recommendation for synchronization via master value distance with 1:1 gear and synchronization from standstill:

- Starting position of the following axis removed from the synchronous position by half the synchronization length.
- Synchronization range symmetrical relative to the synchronous position.

2.2.13.2 Synchronization profile based on specifiable dynamic response parameters

Together with the master value behavior, the starting position of the following axis relative to the synchronous position on the following axis side largely determines the characteristic of the synchronization profile.

Influence of starting position of the following axis with leading synchronization

Example:

- Absolute 1:1 gear without offset
- Constant master value velocity
- Following axis at start of synchronization at standstill

- Synchronization profile based on dynamic response parameters
- Velocity profile type TRAPEZOID for synchronization

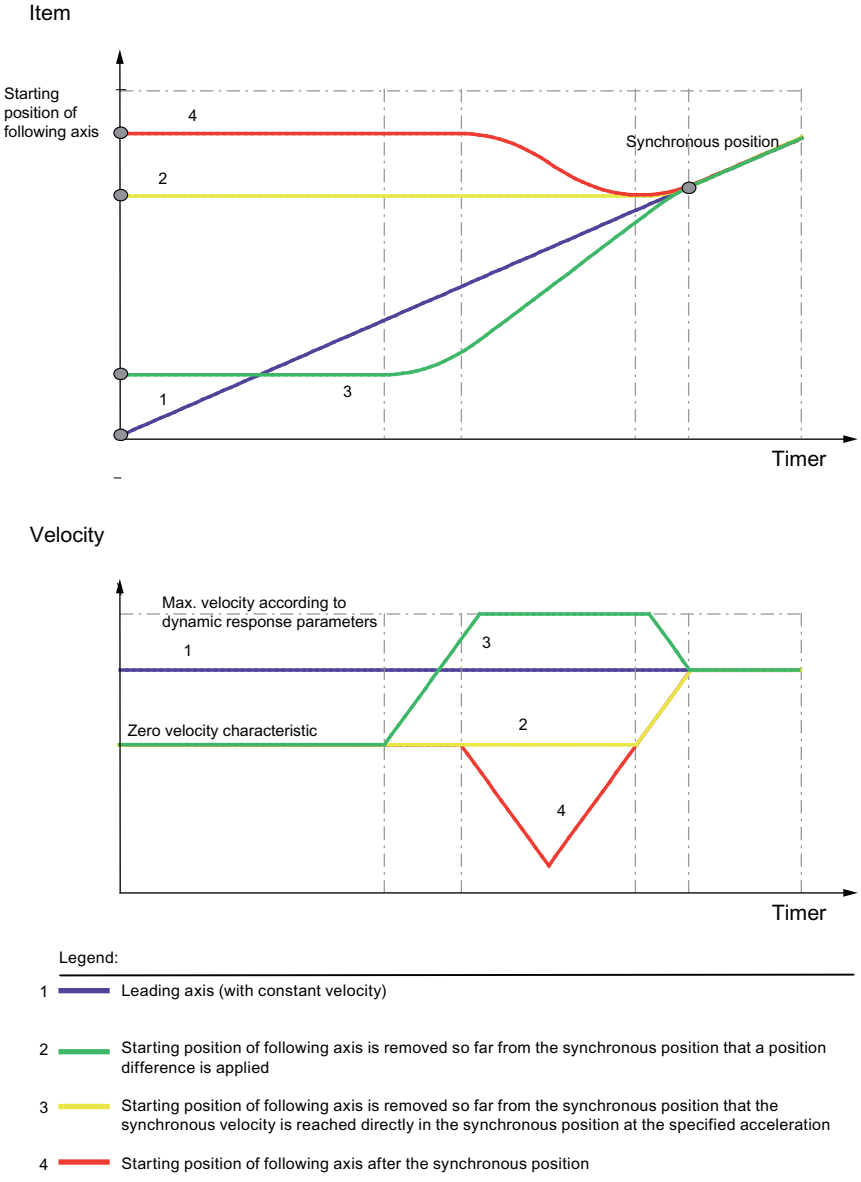


Figure 2-55 Synchronization profile based on specifiable dynamic response parameters and leading synchronization

In the case of leading synchronization based on dynamic response parameters, synchronization starts differently depending on the starting position of the following axis:

- If necessary, the following axis can be accelerated directly to the synchronous velocity and synchronous position (2).
- If the starting position of the following axis is below this point, a position difference must also be applied with the specified dynamic values (3).
- If the starting position is above this point, reversing (i.e. traveling in the opposite direction) is required in order to traverse to the synchronous point at the required synchronous velocity (4).

Influence of starting position of the following axis with trailing synchronization

Together with the master value behavior, the starting position of the following axis relative to the synchronous position on the following axis side largely determines the characteristic of the trailing synchronization profile.

Example:

- Absolute 1:1 gear without offset
- Constant master value velocity
- Following axis at start of synchronization at standstill
- Synchronization profile based on dynamic response parameters
- Velocity profile type **TRAPEZOID** for synchronization

With this type of dynamic response-related synchronization, a master value position is defined from which the synchronization procedure between the master value and following axis is initiated. Here, the desynchronization operation itself is performed based on the dynamic response value settings.

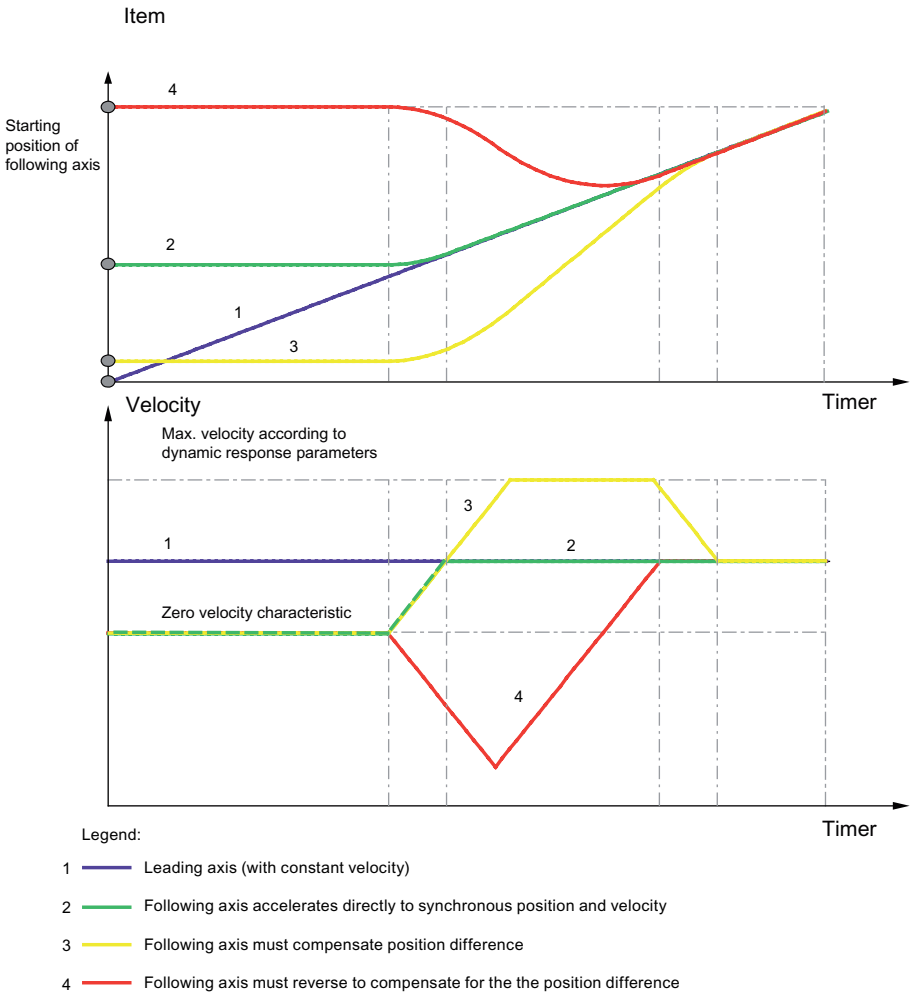


Figure 2-56 Synchronization profile based on specifiable dynamic response parameters and trailing synchronization

With trailing synchronization via dynamic response parameters, "synchronous" status is achieved at different times, depending on the starting position of the following axis.

Depending on the starting position of the following axis:

- If necessary, the following axis can be accelerated directly to the synchronous velocity and synchronous position (2).
- If the starting position of the following axis is below this point, a position difference must also be applied with the specified dynamic values (3).
- If the starting position of the following axis is above this point, reversing (i.e. traveling in the opposite direction) is required in order to traverse to the synchronous point at the required synchronous velocity (4).

2.2.14 Examples

2.2.14.1 Examples of typical synchronization operations

Several examples for synchronization operations of the gearing and their parameterization in MCC and ST commands are listed here.

Note

The function parameters that are not important to function calls are omitted from the examples. The required parameters are entered directly.

Relative synchronization with master value reference

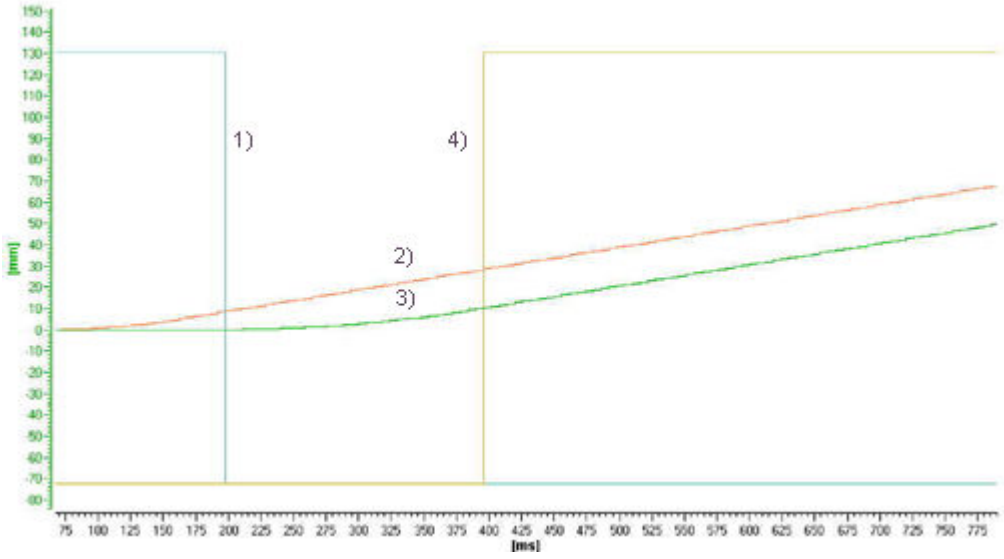
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 0 mm. The synchronization is started immediately; after 20 mm, this should result in relative synchronism between the master axis and following axis.

Table 2-2 ST programming

```
retval:=_enablegearing (
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    direction:=POSITIVE,
    gearingType:=RELATIVE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingDenominator:=1,
    synchronizingMode:=IMMEDIATELY,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE,
    syncLengthType:=DIRECT,
    syncLength:=20.0);
```

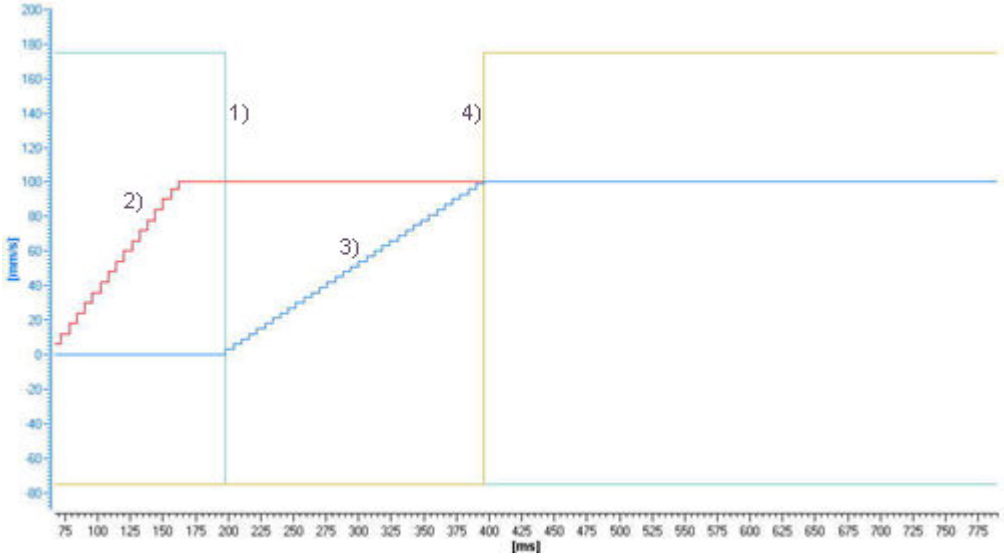
Table 2-3 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	Gearing relative to start position
Synchronization:		
	Synchronization reference:	Leading axis
	Start of synchronization:	Synchronize immediately
	Synchronization length:	20 mm



1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 2-57 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 2-58 Master and following axis velocity

Absolute synchronization with master value reference

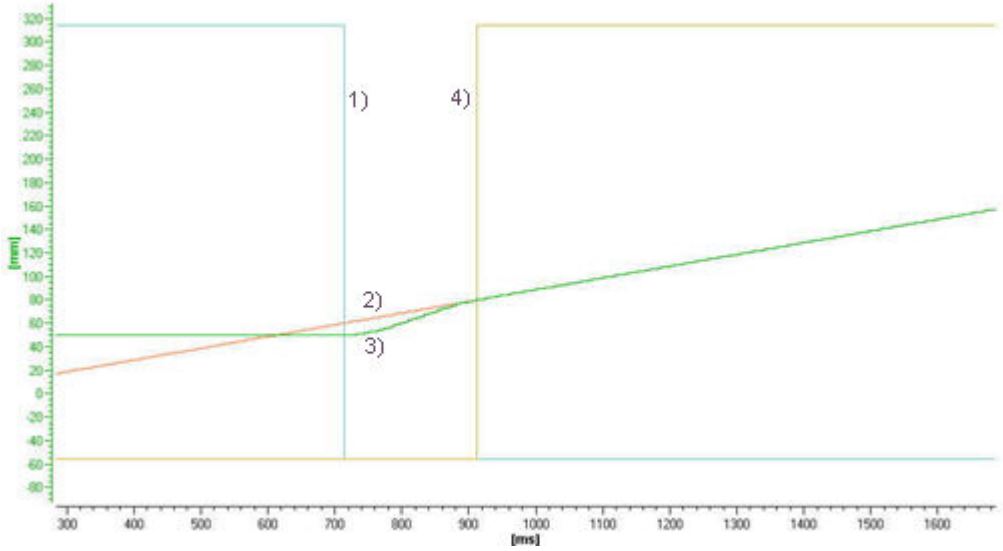
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 50 mm. Synchronization is performed within 20 mm; this means that for a master axis position of 80 mm, absolute synchronism between the master and following axis occurs.

Table 2-4 ST programming

```
retval:=_enablegearing (
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=ABSOLUTE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingRatioType:=DIRECT,
    gearingNumerator:=1,
    gearingdenominator:=1,
    synchronizingMode:=ON_MASTER_POSITION,
    syncPositionReference:=BE_SYNCHRONOUS_AT_POSITION,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE,
    syncLengthType:=DIRECT,
    synclength:=20.0,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=80.0);
```

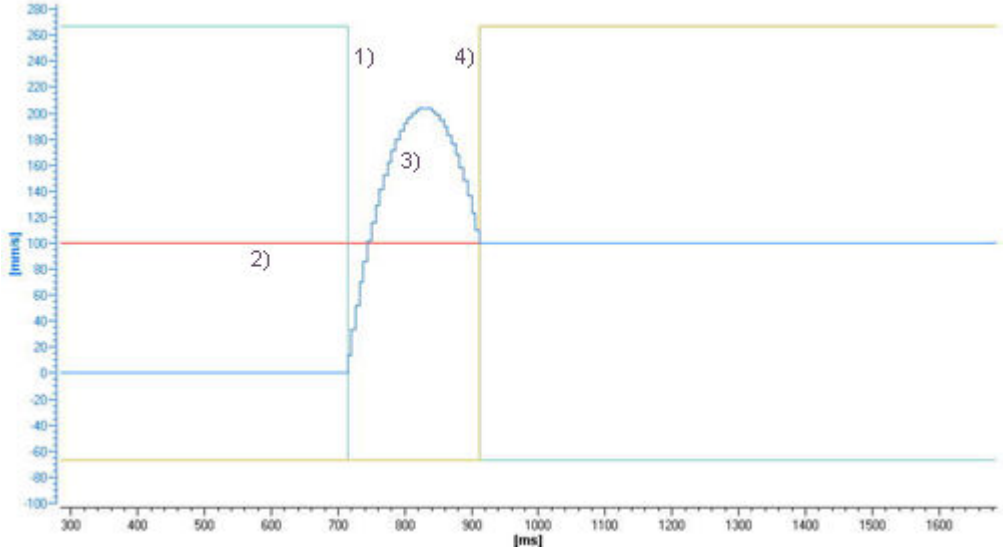
Table 2-5 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	gearing is achieved based on the axis zero point
Synchronization:		
	Synchronization reference:	Leading axis
	Start of synchronization:	at leading axis position
	Reference point of the leading axis position:	Synchronize before synchronization position
	Synchronization length:	20 mm
	Leading axis position:	80 mm



1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 2-59 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 2-60 Master and following axis velocity

Relative synchronization with master value reference and offset

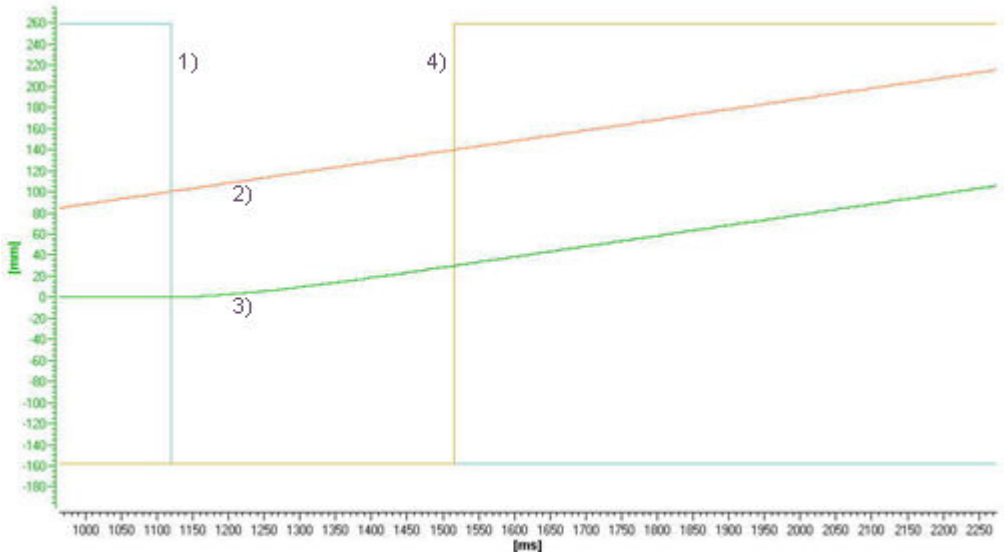
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 0 mm. Starting with a master axis position of 100 mm, the following axis will be synchronized relative to the master axis within 40 mm. When synchronism is achieved, this results in a following axis position based on the position at the start of synchronization and the offset of 30 mm.

Table 2-6 ST programming

```
retval:= _enablegearing(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=RELATIVE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingdenominator:=1,
    synchronizingMode:=ON_MASTER_AND_SLAVE_POSITION,
    syncPositionReference:=SYNCHRONIZE_WHEN_POSITION_REACHED,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE,
    syncLengthType:=DIRECT,
    synclength:=40.0,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=100.0,
    syncPositionSlaveType:=DIRECT,
    syncPositionSlave:=30.0);
```

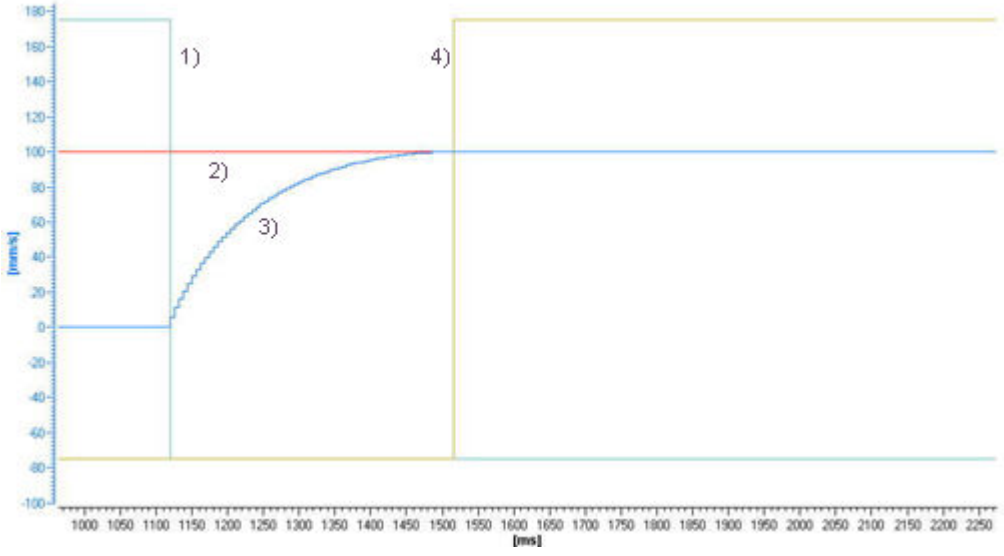
Table 2-7 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	Gearing relative to start position
Synchronization:		
	Synchronization reference:	Leading axis
	Start of synchronization:	at leading axis position with offset
	Offset:	30 mm
	Reference point of the leading axis position:	Synchronize from synchronization position
	Synchronization length:	40 mm
	Leading axis position:	100 mm



1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 2-61 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 2-62 Master and following axis velocity

Absolute synchronization with time reference, trailing synchronization

The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 50 mm. The specified dynamic response parameters (velocity = 300 mm/s and acceleration = 1,000 mm/s^2) are used for synchronization starting from the master axis position 300 mm, in order to achieve absolute synchronism between the master and the following axis.

Note

Depending on the specified offset, the following axis may need to perform a reversing motion.

Table 2-8 ST programming

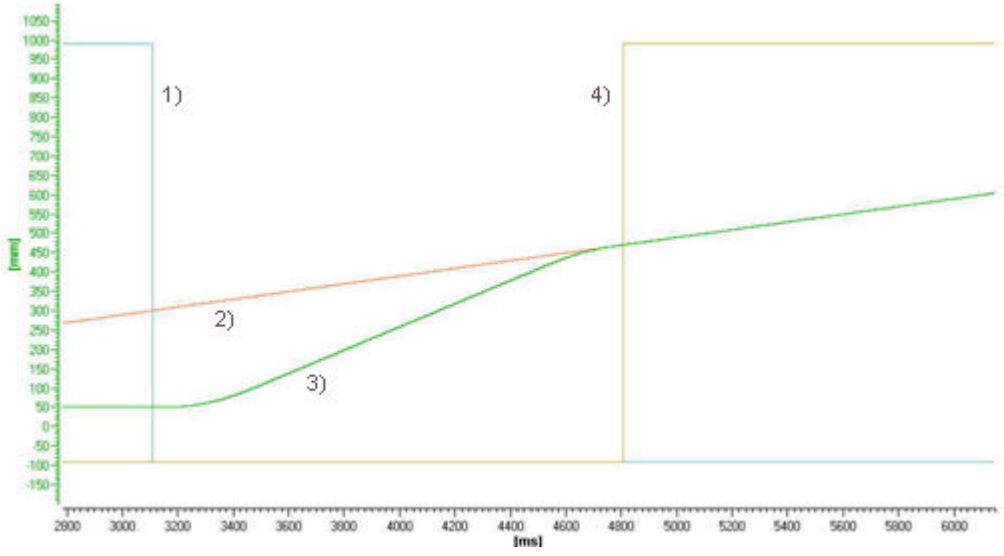
```
retval:= _enablegearing(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=ABSOLUTE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingdenominator:=1,
    synchronizingMode:=ON_MASTER_POSITION,
    syncPositionReference:=SYNCHRONIZE_WHEN_POSITION_REACHED,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=300.0,
    velocityType:=DIRECT,
    velocity:=300.0,
    positiveAccelType:=DIRECT,
    positiveAccel:=1000.0,
    negativeAccelType:=DIRECT,
    negativeAccel:=1000.0,
    positiveAccelStartJerkType:=DIRECT,
    positiveAccelStartJerk:=10000.0,
    positiveAccelEndJerkType:=DIRECT,
    positiveAccelEndJerk:=10000.0,
    negativeAccelStartJerkType:=DIRECT,
    negativeAccelStartJerk:=10000.0,
    negativeAccelEndJerkType:=DIRECT,
    negativeAccelEndJerk:=10000.0,
    velocityprofile:=SMOOTH);
```

Table 2-9 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	gearing is achieved based on the axis zero point
Synchronization:		
	Synchronization reference:	Timer
	Start of synchronization:	at leading axis position
	Reference point of the leading axis position:	Synchronize from synchronization position
	Leading axis position:	300 mm
Dynamic response:		
	Velocity:	300 mm/s
	Deceleration:	1000 mm/s²
	Jerk:	10000 mm/ s³
	Velocity profile:	Constant

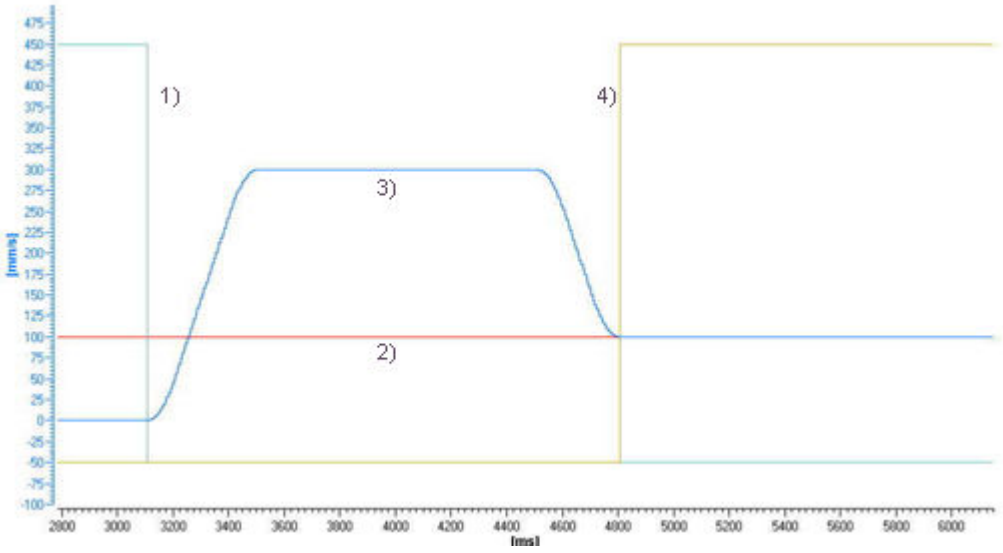
Note

For a motion with a continuous velocity profile, the enable of the jerk-limited synchronization **syncingMotion.smoothAbsoluteSynchronization:=YES** with absolute synchronous operation relationships should be set on the synchronous object.



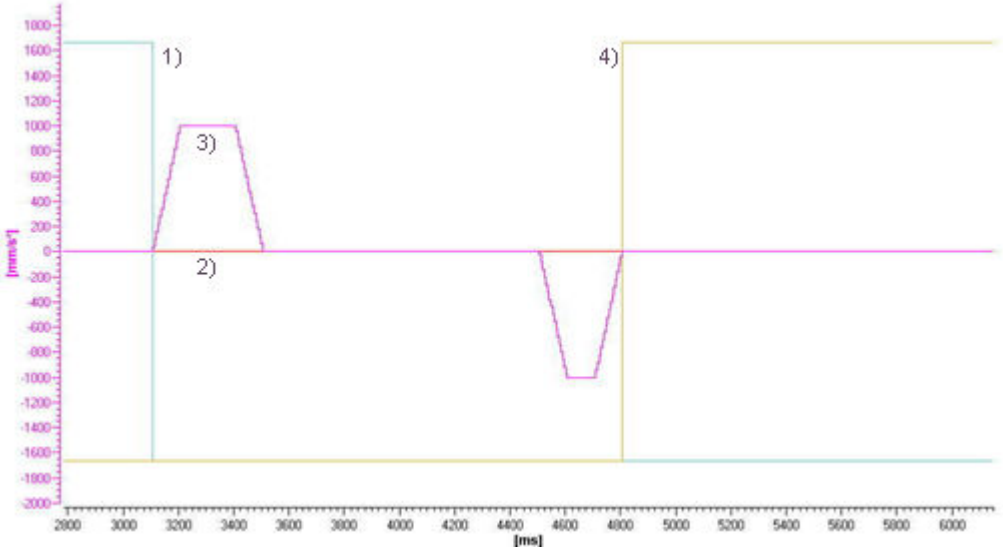
1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 2-63 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 2-64 Master and following axis velocity



1) Start of synchronization
2) Master axis acceleration
3) Following axis acceleration
4) Synchronization status

Figure 2-65 Master and following axis acceleration

Absolute synchronization with time reference, leading synchronization

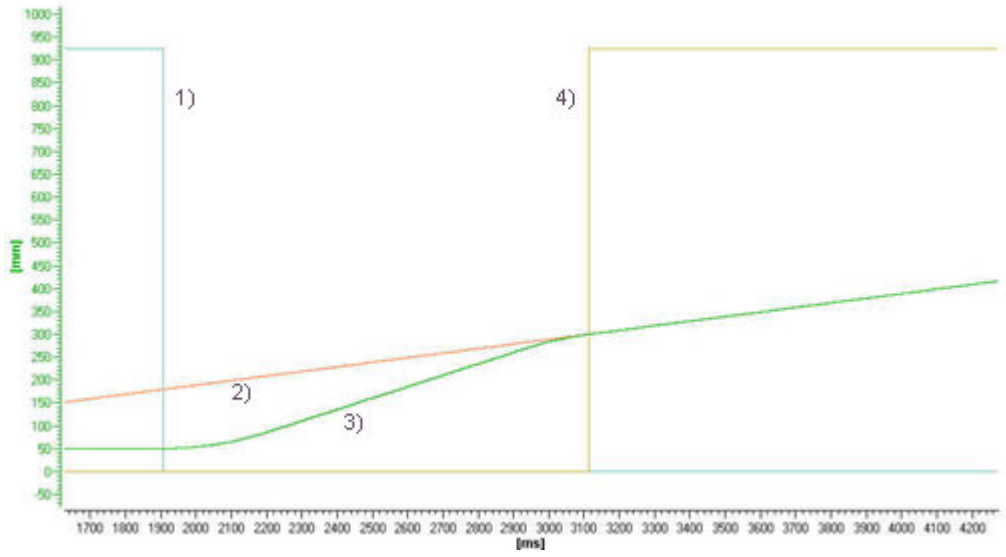
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 50 mm. The specified dynamic response parameters (velocity = 300 mm/s and acceleration = 1,000 mm/s²) are used for synchronization so that, with a master axis position of 300 mm, absolute synchronism exists between the master and following axes. For the synchronization procedure, a maximum change in the master value velocity of 20% is permitted on the synchronous object (**syncingMotion.maximumOfMasterChange**).

Table 2-10 ST programming

```
retval:= _enablegearing(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=ABSOLUTE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingDenominator:=1,
    synchronizingMode:=ON_MASTER_POSITION,
    syncPositionReference:=BE_SYNCHRONOUS_AT_POSITION,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=300.0,
    velocityType:=DIRECT,
    velocity:=300.0,
    positiveAccelType:=DIRECT,
    positiveAccel:=1000.0,
    negativeAccelType:=DIRECT,
    negativeAccel:=1000.0,
    velocityProfile:=TRAPEZOIDAL);
```

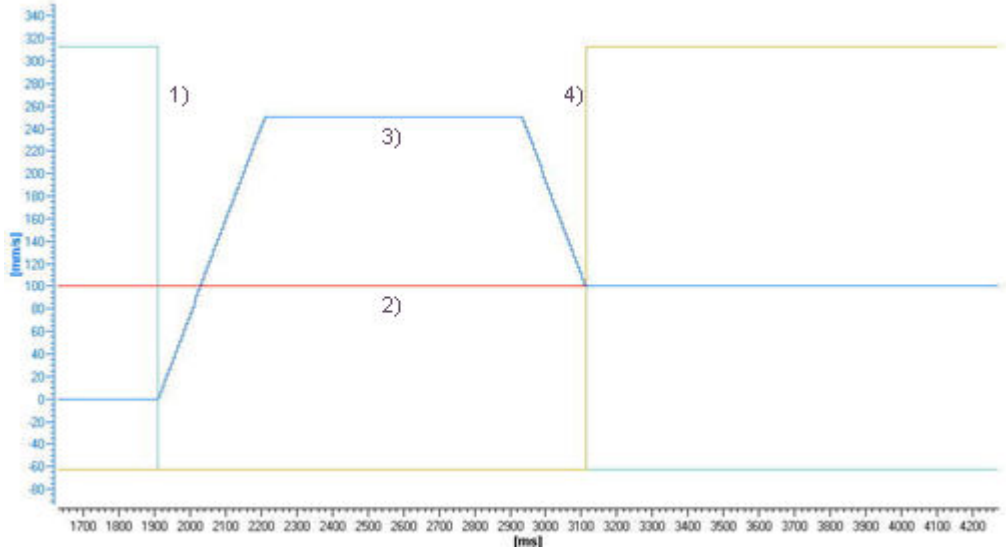
Table 2-11 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	gearing is achieved based on the axis zero point
Synchronization:		
	Synchronization reference:	Timer
	Start of synchronization:	at leading axis position
	Reference point of the leading axis position:	Synchronize before synchronization position
	Leading axis position:	300 mm
Dynamic response:		
	Velocity:	300 mm/s
	Deceleration:	1000 mm/s²
	Velocity profile:	Trapezoidal



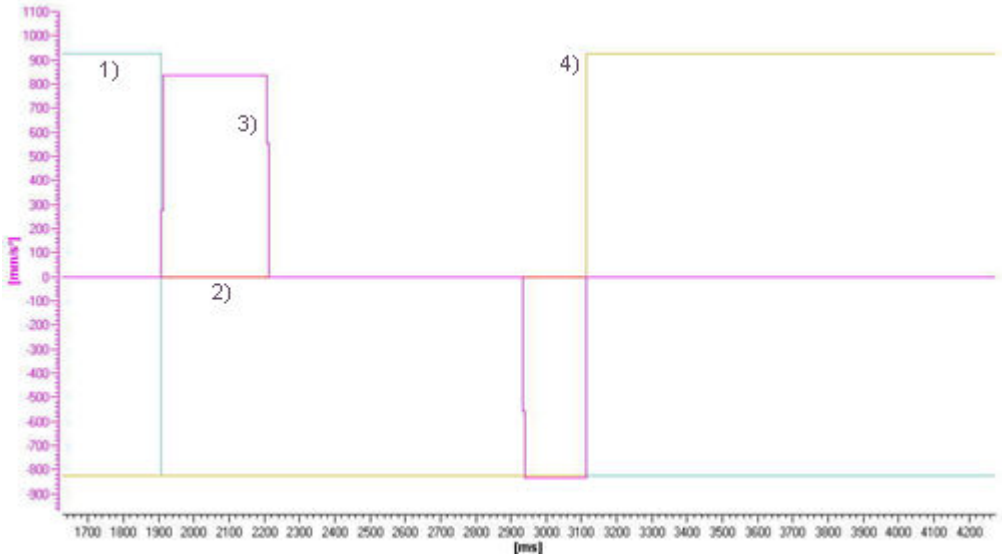
1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 2-66 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 2-67 Master and following axis velocity



1) Start of synchronization
2) Master axis acceleration
3) Following axis acceleration
4) Synchronization status

Figure 2-68 Master and following axis acceleration

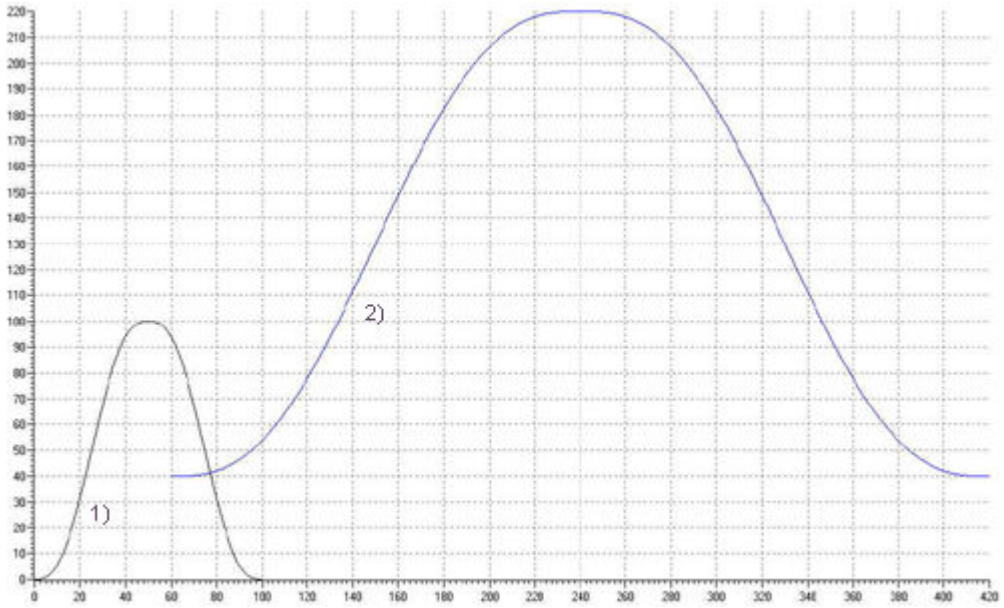
2.2.14.2 Example of offset and scale on the synchronous object

A leading axis provides position values over a range of 360° and 60° phase offset, i.e. from 60...420°. The following axis should move in the range 40...220°.

Definition range of cam: 0...100
Range of cam: 0...100

The definition and value range of the cam can be adapted to the required representation range for the camming using scaling and offset as follows (See also Figure **Equation for scale and offset on the camming** in Chapter **Camming**):

	Scaling	**Offset**
Master value	360 / 100 = 3,6	60
Slave value	(220 - 40) / 100 = 1,8	40



1) unscaled, non-offset function

2) scaled, offset function

Figure 2-69 Example of the scaling and the offset of a cam

In the following programming example, the offset and scale commands for the synchronous object take effect when camming is next activated:

Table 2-12 ST programming

```
(*scaling of the master value*)
    retval:= _setCammingScale(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    scalingRange:= MASTER_RANGE,
    scaleValue:= 3.6,
    activationMode:= DEFAULT_VALUE
);
(*scaling of the slave value*)
retval:= _setCammingScale(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    scalingRange:= SLAVE_RANGE,
    scaleValue:= 1.8,
    activationMode:= DEFAULT_VALUE
);
(*offset of the master value*)
retval:= _setCammingOffset(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    offsetRange:= MASTER_RANGE,
    offsetMode:= ABSOLUTE,
    offsetValue:= 60.0,
```

```
    activationMode:= DEFAULT_VALUE
);
(*offset of the slave value*)
retval:= _setCammingOffset(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    offsetRange:= SLAVE_RANGE,
    offsetMode:= ABSOLUTE,
    offsetValue:= 40.0,
    activationMode:= DEFAULT_VALUE
);
```

MCC programming

Table 2-13 <Set scaling on camming>:master value side

Parameters:		
	Range:	Master Range
	Offset:	3.6
	Effect:	on following commands

Table 2-14 <Set scaling on camming>:slave value side

Parameters:		
	Range:	slave range
	Offset:	1.8
	Effect:	on following commands

Table 2-15 <Set offset on camming>:master value side

Parameters:		
	Range:	Master Range
	Offset:	60.0
	Mode:	Absolute
	Effect:	on following commands

Table 2-16 <Set offset on camming>:slave value side

Parameters:		
	Range:	slave range
	Offset:	40.0
	Mode:	Absolute
	Effect:	on following commands

See also

Camming (Page 26)

2.2.14.3 Example of applying offset as superimposition

The following example shows the two dynamic corrections of the **dynamicReference** parameter in their different effects using an accelerating master value.

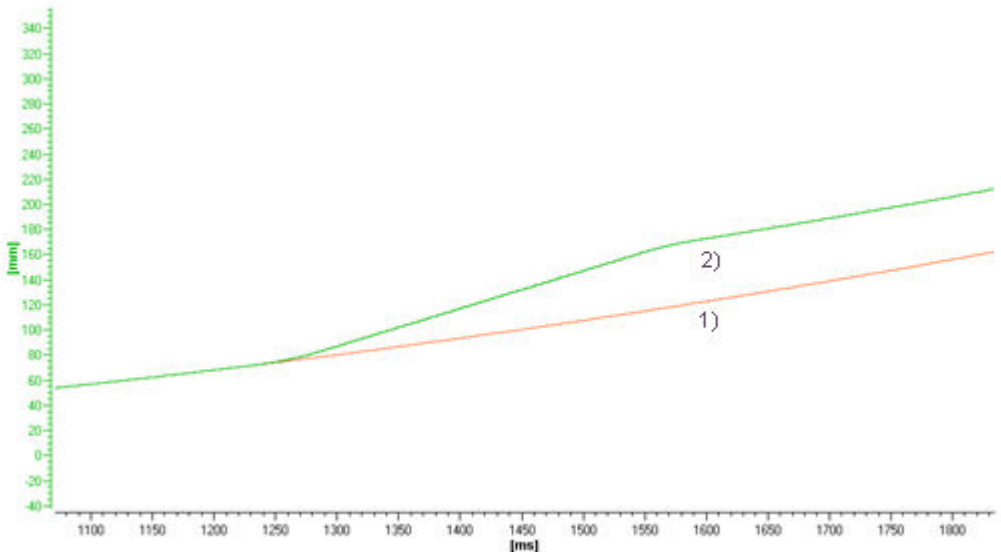
Example: Master and following axis for a synchronous operation of 1:1 are absolutely synchronous, accelerate at 100 mm/s². The offset of -50 mm on the master value side is implemented using _setGearingOffset for a programmed correction velocity of 300 mm/s and a correction acceleration of 3,000 mm/s², by means of a non-continuous-acceleration velocity profile.

Table 2-17 ST programming

```
retval := _setGearingOffset (
    followingObject:= <SYNCHRONOUSOBJECT>,
    offsetRange := MASTER_RANGE,
    offsetMode := RELATIVE,
    offsetValue := -50.0,
    velocityType := DIRECT,
    velocity := 300.0,
    positiveAccelType := DIRECT,
    positiveAccel := 3000.0,
    negativeAccelType := DIRECT,
    negativeAccel := 3000.0,
    velocityProfile := TRAPEZOIDAL,
    activationMode := ACTUAL_VALUE,
    dynamicReference := TOTAL_MOVE / OFFSET_MOVE
);
```

Note

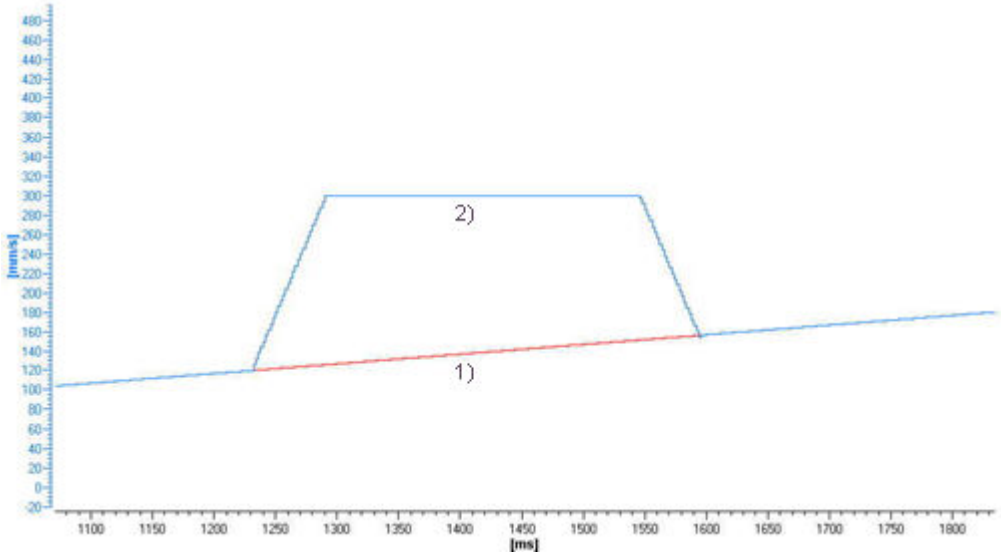
When the master value velocity is constant, the dynamic transitions have a generally identical form and differ only as a result of the dynamic response parameters that act differently.



1) Master axis position

2) Following axis position

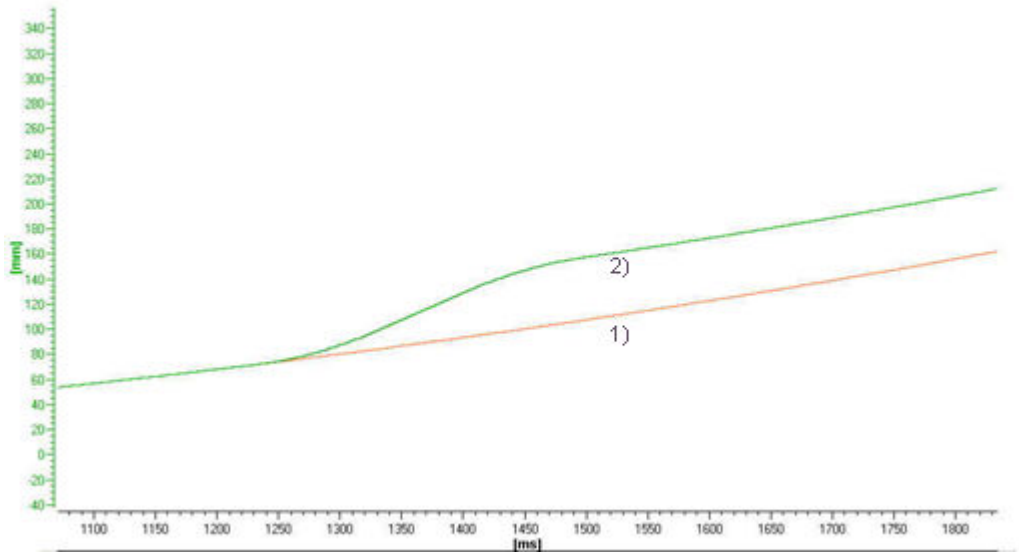
Figure 2-70 Leading and following axis position for dynamicReference:= TOTAL_MOVE



1) Master axis velocity

2) Following axis velocity

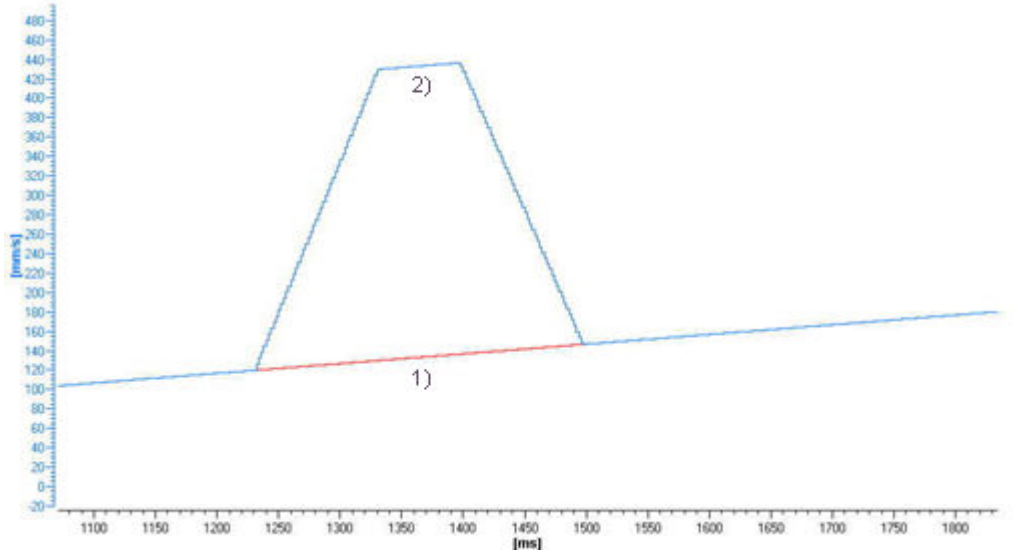
Figure 2-71 Leading and following axis velocity for dynamicReference:= TOTAL_MOVE



1) Master axis position

2) Following axis position

Figure 2-72 Leading and following axis position for dynamicReference:= OFFSET_MOVE



1) Master axis position

2) Following axis position

Figure 2-73 Leading and following axis velocity for dynamicReference:= OFFSET_MOVE

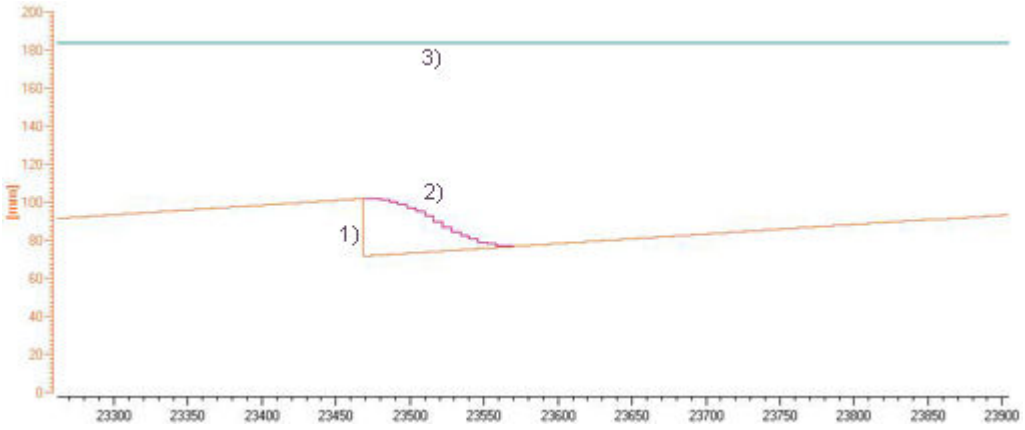
2.2.15 Special actions

2.2.15.1 Redefining the axis position during active synchronous operation

The following options are available for resetting an axis position, e.g. by means of **_redefinePosition()** or **_homing()**:

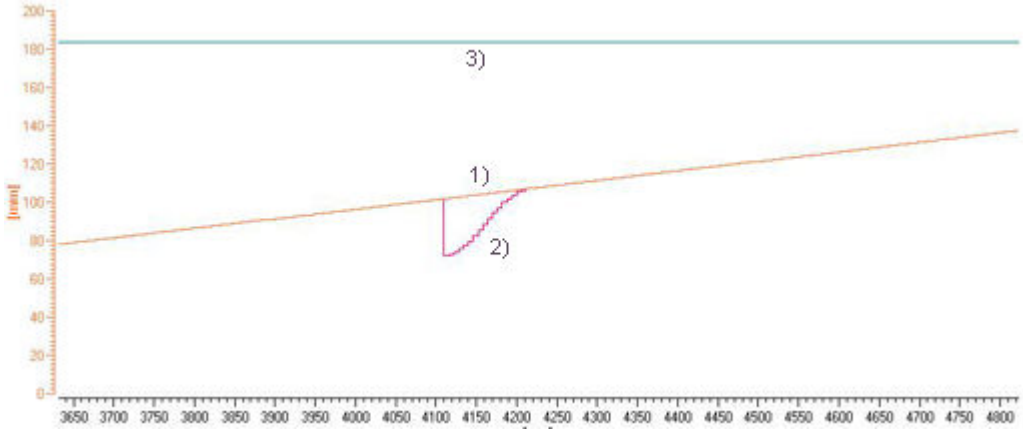
- Redefining the *master axis position* causes a jump in the master value.
  The following axis then performs a compensating movement and finally moves in synchronism again with the master axis. If the position tolerance > the synchronous operation tolerance, the error 40201 "Synchronous operation tolerance on gearing axis exceeded" will be output.
- Redefining the *following axis position* does *not* cause a jump on the master value.
  - For the *absolute synchronous operation*, the slave axis is no longer position-synchronous and so performs a compensating movement.
  - For the *relative synchronous operation*, the following axis does *not* perform a compensation motion, because position synchronization is not necessary.

Examples:



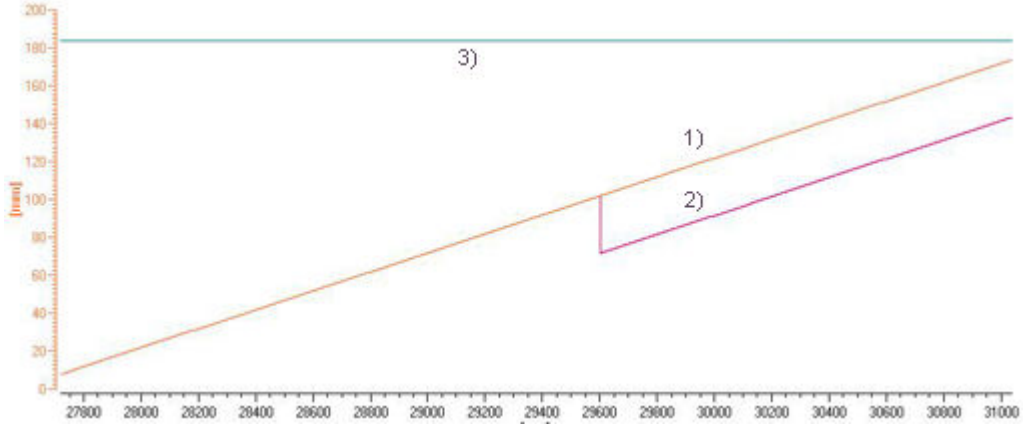
1) Master axis position
2) Following axis position
3) Synchronization status

Figure 2-74 Redefining the leading axis position (absolute or relative gearing) -> The following axis performs a compensation motion



1) Master axis position
2) Following axis position
3) Synchronization status

Figure 2-75 Redefining the following axis position for the absolute gearing -> The following axis performs a compensation motion



1) Master axis position
2) Following axis position
3) Synchronization status

Figure 2-76 Redefining the following axis position for the relative gearing -> The following axis does not perform a compensation motion

2.2.15.2 Retaining a synchronous connection for _disableAxis

If the slave axis is no longer able to apply the generated synchronous operation setpoints, e.g. due to the removal of enables or an error response, an active synchronous operation connection is closed. It is possible to retain the synchronous operation connection by using synchronous operation in simulation mode and setting
the **DecodingConfig.disableSynchronousOperation=YES** configuration data element on the synchronous object.

See Simulation mode (Page 78).

The existing synchronous connection remains active except in the following situations:

- Restart of following axis (**_restartAxis()**)
- Invalid actual values on the following axis

When simulation mode is terminated, the current synchronous operation setpoints are applied immediately as axis setpoints.

Remove the axis from the synchronous operation interconnection, and return it

Example:
Opening and closing protective doors

Removing an axis from the synchronized group

1. Stop the leading axis.
2. Switch the synchronous operation to simulation mode.
3. **DecodingConfig.disableSynchronousOperation** on the synchronous object must be set to **YES**. As a result, the synchronous operation connection is not disconnected on **_disableAxis**.
4. If a superimposed motion had been performed (e.g. for correcting):
   - Save the position of the superimposed coordinate system in a user variable.
   - Perform **_redefinePosition** in the coordinate system 2 to absolute position 0.
5. Cancel the controller enables (**_disableAxis()**). This will clear the drives. The axis goes into follow-up mode, "control" on the axes is deleted.

Returning the axis to the synchronized group

1. Reconnect the controller enables (**_enableAxis()**).
2. If necessary, use **_redefinePosition** to pass the saved actual position as absolute position to the coordinate system 2.
3. Switch synchronous operation back from simulation mode.

A compensation motion will be performed if the setpoints for the synchronous object and the setpoints on the axis do not match.

2.2.15.3 Substitution of velocity gearing with absolute synchronous operation

A synchronous velocity operation cannot be substituted directly with an absolute synchronous operation. Immediately afterwards, only a relative synchronous operation can be executed.

If an attempt is made to execute an absolute synchronous operation immediately after a velocity gearing, the following technological alarm is output:
50110 "Call-up of an absolute synchronous position operation after a synchronous velocity operation not permitted" (as of V3.2)

Procedure:

- Switch a relative position-controlled synchronous operation in between for one IPO cycle. (mergeMode = IMMEDIATELY and then a **waitTime** with Time = 0 sec)

2.2.15.4 Motion with absolute synchronous operation

Substitution of a position axis by an absolute synchronous operation (as of V4.4)

A position axis traversing in speed-controlled mode (_move(movingMode = SPEED_CONTROLLED)) can be substituted directly by an absolute synchronous operation. As a consequence of this, for example, with a stationary master value, the following axis positions itself at the absolute position of the master value.

2.2.15.5 Canceling active and pending synchronous operations

If gearing or camming is active and another gearing or camming operation is started, the first **_disableCamming()/_disableGearing()** command ends the synchronous operation to be synchronized, and the second **_disableCamming()/_disableGearing()** command ends the active synchronous operation.

Alternatively, in V4.1 and higher the synchronous operation commands can be canceled using the commandId associated with the command. The **_cancelFollowingObjectCommand()** command can be used to cancel the synchronous operation command by specifying the commandId in the **commandToBeCancelled** parameter. This removes the synchronous operation command from the command buffer. This means that a pending command for synchronization can be canceled (for example, if the application recognizes that resynchronization is not intended to take place).

2.2.15.6 Adapt the synchronization velocity to the master value velocity

The following axis cannot be synchronous with the master object if the velocity of the following axis for the synchronization process is lower than that of the master value during the synchronization process.

Example: The master axis is, for example, currently travelling at 200 mm/s. A gearing is started for which the synchronization velocity is limited by the default value of 100 mm/s. The following axis cannot overtake the master object.

To adapt the synchronization velocity of the following axis to the master value velocity, proceed as follows:

1. Create a **master axis**.
2. Create a **following axis** (synchronized axis).
   A **Following_Axis_SYNCHRONOUS_OPERATION** object is automatically created below the following axis.
   The project navigator should look like this:

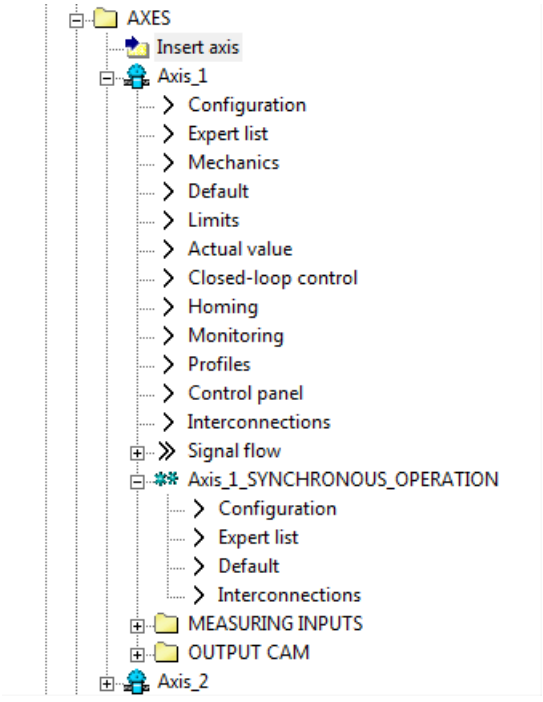


Figure 2-77 Axes for example

3. Make the following settings for **FollowingAxis_SYNCHRONOUSOPERATION -> Settings**:
   – **Overdrive factor for dynamic values**: 150 %

– **Adapting the dynamic values for synchronization**: activated

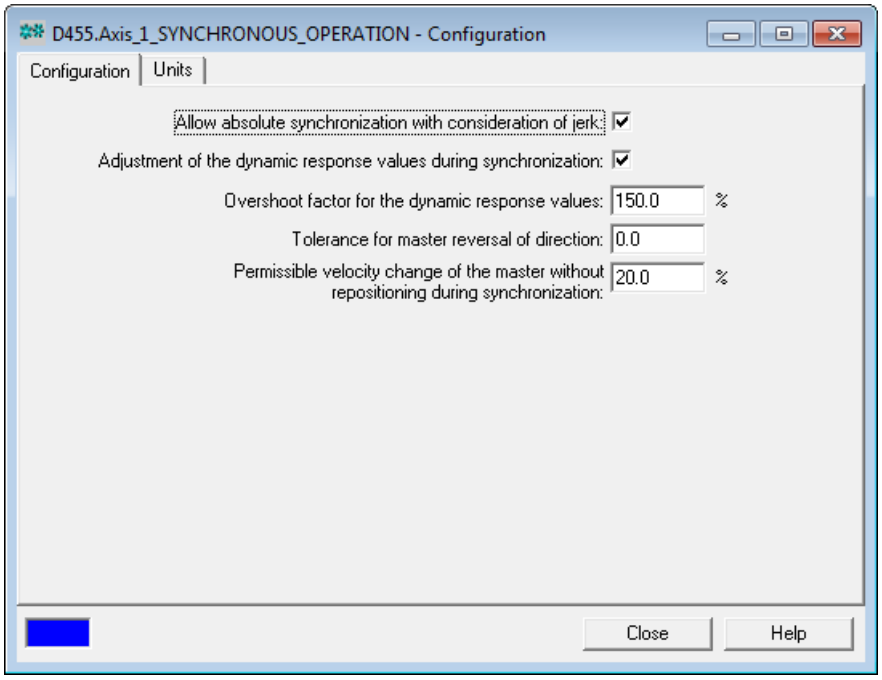


Figure 2-78 Settings for FollowingAxis_SYNCHRONOUSOPERATION

For the synchronization, the following axis uses the dynamic response values of the master axis (increased by the **overdrive factor**) and travels with the maximum velocity of 300 mm/s.

Another application scenario

The master object travels with the constant velocity of 80 mm/s. The dynamic response setting of the following axis allows a synchronization velocity of 100 mm/s. Under these conditions, the following axis can synchronize itself without adaptation of the dynamic response values. If the master object accelerates to 150 mm/s, but the **synchronous** status is not yet attained, the following axis can synchronize itself only with adaptation of the dynamic response values. During the synchronization action, the maximum velocity of the following axis is calculated using the current set velocity of the master object (150% of the current value).

2.2.15.7 Substitution of synchronous operation through _move, _pos, _stop with reset of an existing acceleration to zero

Description

When a motion is replaced with the **_stop()**, **_pos()** and **_move()** functions and a smooth velocity profile with jerk specification is applied during the axis acceleration phase, the velocity can continue to increase until the acceleration is reduced by means of the jerk that has been set. In extreme cases, the velocity can rise still further until it reaches the configured maximum velocity. The axis is only decelerated once acceleration has been reduced by the jerk.

With the **abortAcceleration** parameter, you can set that a possible acceleration does not exceed the jerk limitation, but is immediately cleared.

The **abortAcceleration** parameter is available for **_stop()** as of V4.2.1 and for **_pos** and **_move** as of V4.4.

You will find more information in the *TO Axis Electric/Hydraulic, External Encoder* Function Manual.

2.3 Synchronous Operation Configuration

This section shows you how to create and configure axes with **synchronous operation** in SIMOTION SCOUT.

It is assumed that you have already created cams, master axes, or external encoders.

Note

If the actual values/setpoints are required to be equal during a synchronous operation, the same Ti (actual value acquisition) and To (setpoint acceptance) times must be adopted for all drive units used (e.g. SINAMICS Integrated, CU320).

When configuring a synchronous operation, you must follow the steps below:

- Create an axis with synchronous operation functionality (Page 111).
- Assign master values and cams to the synchronized axis (Page 112).
- Assign synchronous operation parameters (Page 114).
- Define the settings for the synchronization procedure (Page 130).
- Define synchronous operation monitoring (Page 132).

2.3.1 Creating an axis with synchronous operation

Create a synchronous axis as follows:

1. To create an Axis TO with the Synchronous operation technology in **SCOUT**, double-click **Insert axis** under **AXES** in the project navigator. You can also copy an existing Axis technology object using the clipboard and insert it with another name.
2. Activate for creating the axis the **Synchronous operation** technology. A synchronous object will automatically be created.

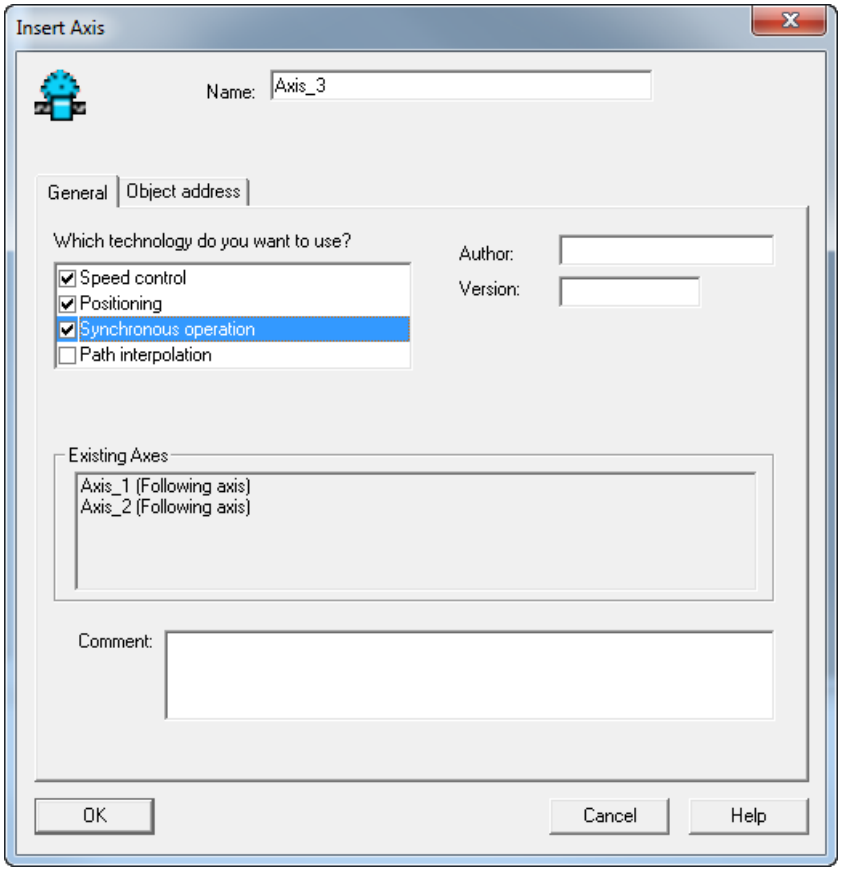


Figure 2-79 Inserting an axis with synchronous operation

Note

If synchronous operation technology is specified for an Axis technology object, the synchronous object will be inserted together with the Axis technology object. The synchronous object is permanently assigned to the Axis technology object. The name of the synchronous object is automatically defined and cannot be changed.

It is *not* possible to subsequently convert a positioning or speed axis to a following axis.

It is *not* possible to add a synchronous object. Only a superimposed synchronous operation can be added to the synchronous axis using expert (see Superimposed synchronous operation (Page 71)).

Representation in the project navigator

The synchronous object is generated automatically when a synchronous axis is created and is displayed below this synchronous axis in the project navigator. The object is automatically given the name of the axis, followed by _SYNCHRONOUS_OPERATION. There the permitted synchronous operation relationships of the synchronous axis can be defined and the preassignments for the synchronous operation coupling to a leading axis assigned.

The assignment of the master values and cams is indicated in the project navigator using links:

- For the **Synchronous object**:
  Links to the master values (axes, external encoders, addition objects, formula objects, and fixed gears) as well as cams
- For the technology objects used:
  Link to the synchronous object
- Below the **master values** (axes, external encoders, addition objects, formula objects, and fixed gears): Link to the synchronous object

Superimposed synchronous operation

If a superimposed synchronous operation relationship to another master value is also to be established, an additional synchronous object can be placed below the following axis via the shortcut menu for the following axis (**Expert > Insert superimposed synchronous object**) (see Superimposed synchronous operation (Page 71)).

The standard and superimposed synchronous operation relationships then affect the following axis additively by means of the two master values.

2.3.2 Assigning master values and cams

When an axis with synchronous operation is created, the synchronous operation configuration still has to be specified, i.e., the master values to be used must be selected and, if required, a cam must be assigned.

Note

Synchronous operation is not possible if a master value is not assigned. Camming is not possible if a cam is not assigned.

Defining the synchronous operation configuration

- In the project navigator, double-click on **Interconnections** under the object <Axis name>**_SYNCHRONOUS OPERATION**. The window below opens.

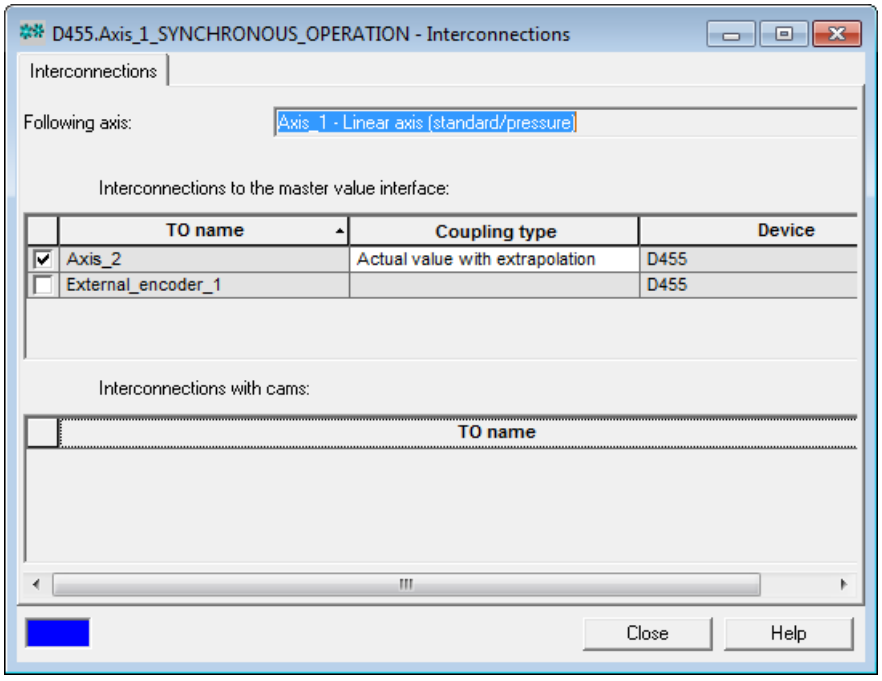


Figure 2-80 Selection of master values and cams

Assign the master values and cams to the following axis in this window.

You can set the following parameters:

Table 2-18 Parameters for configuring synchronous operation

Field/Button	Meaning/Instruction
Following axis	The name of the following axis is displayed here.
Possible master values (leading axis)	The master values available in the project, which you can assign to the following axis, are listed here. The master value can be specified by the following technology objects: • Axis (real or virtual axis) • External encoder • Fixed gear • Addition object • Formula object In accordance with the specified synchronous operation condition (e.g. camming), the slave value is calculated on the basis of the master value and assigned to the following axis as a leading value. When more than one master value is assigned, you must specify which master value is to be used by means of programming in SIMOTION SCOUT (e.g. with MCC).
Possible cams	Cams created in the project are listed here. You can assign cams to the synchronous object for camming. When more than one cam is assigned, you must specify which cam is to be used by means of programming in SIMOTION SCOUT (e.g. with MCC).

1. Assign the desired **master values** to the axis with synchronous operation. You select the current master value to be used in the user program (**_setMaster**).
2. Assign the desired **cams** to the axis with synchronous operation. You select the cam to be used in the user program (**_enableCamming**).
3. For real axes or external encoders, select **setpoint coupling** or **actual value coupling** for axes, or select **actual value coupling** or **actual value coupling with extrapolation** for external encoders.
   See Setpoint/actual value coupling (Page 35).

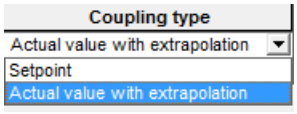


Figure 2-81 Selection of the coupling type

When the window is **closed**, the configuration is accepted and saved automatically.

2.3.3 Assigning parameters/defaults for synchronous operation

In the project navigator, double-click **Default** under the object <Axis name>**_SYNCHRONOUS OPERATION** to change the default settings.

Synchronous operation - Default

In this window, you define the replacement values (default) for calling the synchronous operation functions (**_enableGearing**, **_enableVelocityGearing**, and **_enableCamming** or **_disableGearing**, **_disableVelocityGearing**, and **_disableCamming**).

These replacement values are only evaluated if no special settings are made in the associated function calls for synchronization/desynchronization (ST and LAD or MCC).

You can set parameters for the following functionality:

- Gearing (Page 116)
- Velocity gearing (Page 117)
- Camming (Page 118)
- Gearing synchronization (Page 120)
- Camming synchronization (Page 124)
- Dynamic response (Page 127)
- Master dynamic response (Page 129)

Note

All tabs are always available for default settings. Only parameters used for the relevant functionality are evaluated.

Further information

- For information about this function, refer to Overview of synchronous operation and Fundamentals of synchronous operation.
- For information about programming, refer to Synchronous operation programming/ references (Page 133).
- The meaning of the dialog window parameters and their permissible value ranges can be found in the **SIMOTION reference lists**.

2.3.3.1 Gearing

Gearing is characterized by a constant coupling between the master value source and following axis. This coupling (via the gear ratio) can be specified as the ratio of two decimal numbers (numerator/denominator) or as a floating-point number.

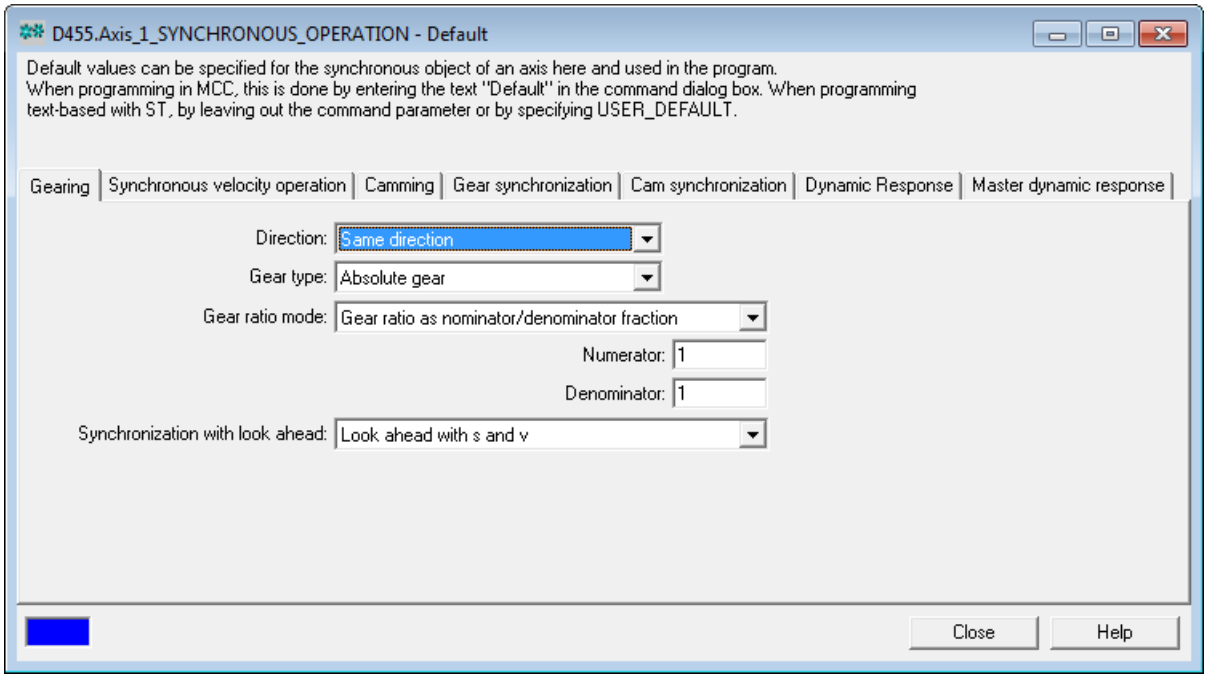


Figure 2-82 Synchronous object: Default setting for gearing

Gearing - default

In the **Gearing** tab, select the **direction**, **absolute** or **relative synchronous operation** and the **gear ratio**. These settings are only relevant when **gearing** mode is used.

You can set the following parameters:

Table 2-19 Parameters for gearing

Field/Button	Meaning/Instruction
Direction	Here, you specify the direction of the gearing.
Gear type	Here, you select the gear type (absolute or relative).
Gear ratio mode	Specify the gear ratio mode here. Further parameters are displayed depending on the selected mode (gear ratio as floating-point number or gear ratio as numerator/denominator ratio).
Gear ratio	You can enter the gear ratio as a floating-point number here.
Counter instructions	Here, you can enter the numerator of the gear ratio in the form of a numerator/denominator ratio.
Denominator	Here, you can enter the denominator of the gear ratio in the form of a numerator/denominator ratio.
Synchronization with look-ahead	You can set here whether a constant acceleration/deceleration of the master value is to be used for the synchronization.

See also

Gearing (Page 20)

2.3.3.2 Velocity gearing

In contrast to gearing or camming, which relate to the position of an axis, velocity gearing relates to the velocity of an axis with a constant coupling between the master value source and the following axis. With velocity gearing, a non-position-controlled, i.e. speed-controlled synchronous operation with the velocity of a master axis, is activated on an axis that is set as a position-controlled axis (synchronized axis).

Possible master values are:

- Velocity of a master axis configured as a position-controlled axis
- Velocity of an external encoder

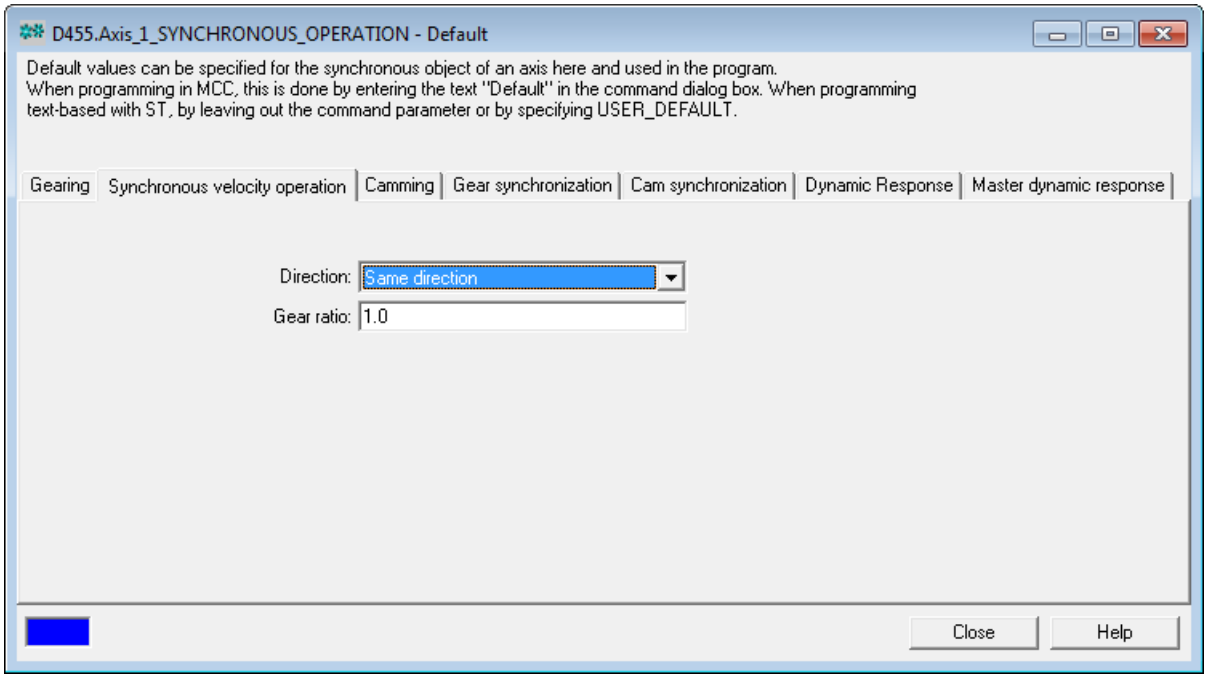


Figure 2-83 Synchronous object: Default setting for synchronous velocity operation

Velocity gearing - default

On the **Velocity gearing** tab, you set the **direction** and the **gear ratio**. These settings are only relevant when **velocity gearing** mode is used.

You can set the following parameters:

Table 2-20 Parameters for velocity gearing

Field/Button	Meaning/Instruction
Direction	Specify the direction of the synchronous velocity operation here.
Gear ratio	Enter here a coupling ratio for the synchronous velocity operation as floating-point number.

See also

Velocity gearing (Page 25)

2.3.3.3 Camming

Camming is characterized by variable coupling between the master value source and the following axis. The coupling is described by a cam (transmission function). Camming can be scaled and offset on both the master value source side and on the following axis side. This enables a cam to be custom-adjusted in its definition range and value range.

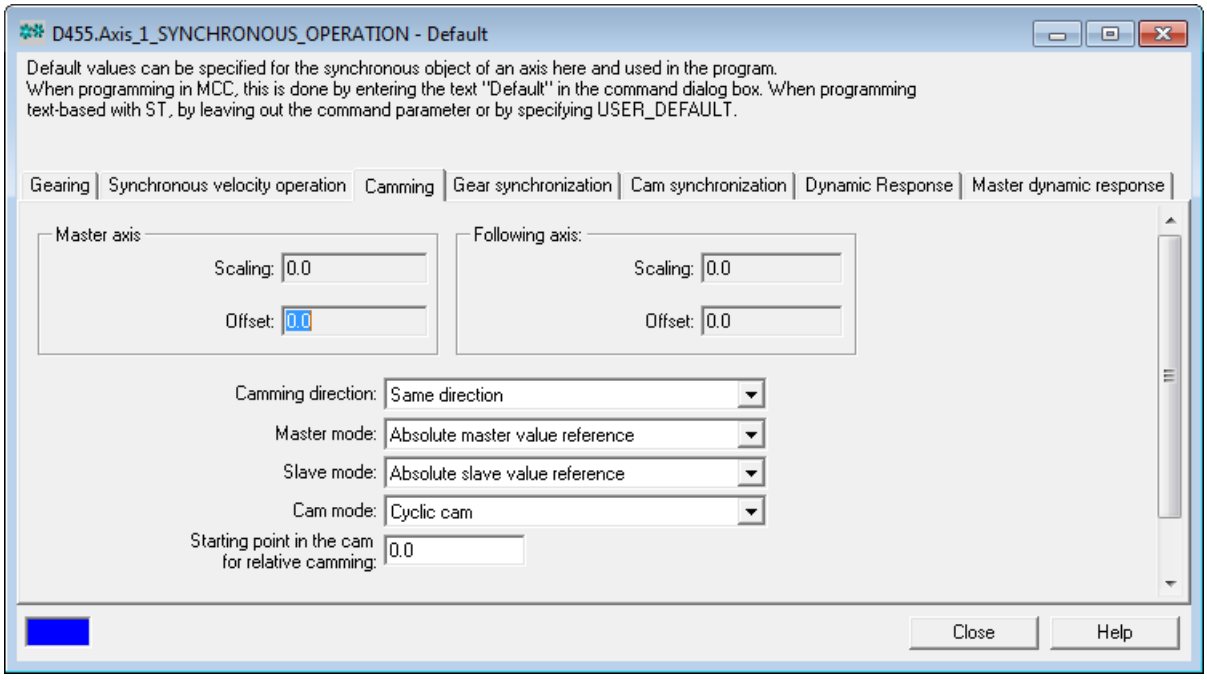


Figure 2-84 Synchronous object: Default setting for camming

Camming - Default

On the **Camming** tab, the **scaling** and **offset** for each master axis and following axis are shown. The **Scaling** and **Offset** setting can be made on the cam or via **_setCammingScale** or **_setCammingOffset**. You select the **direction**, **absolute** or **relative synchronous operation** for the master value and the slave value, as well as the **cam mode**. These settings are only relevant when **camming** mode is used.

You can set the following parameters:

Table 2-21 Parameters for camming

Field/Button	Meaning/Instruction
Leading axis	
Scaling	The scaling of the master value is displayed here.
Offset	The offset of the master value is displayed here.
Following axis	
Scaling	The scaling of the following axis is displayed here.
Offset	The offset of the slave axis is displayed here.
Camming direction	Specify the direction in which the cam is run.
Master mode	Specify the mode in which the master value runs through the cam (absolute or relative).
Slave mode	Specify the mode in which the following axis runs through the cam (absolute or relative).
Cam mode	Specify the execution type for the cam here (cyclic or non-cyclic).
Starting point in the cam for relative camming	Specify here for relative master value reference the reference of the master values by specifying a starting position (camStartPositionMaster) in the cam. The position is always an absolute reference in relation to the definition range of the cam.

See also

Camming (Page 26)

2.3.3.4 Gearing synchronization

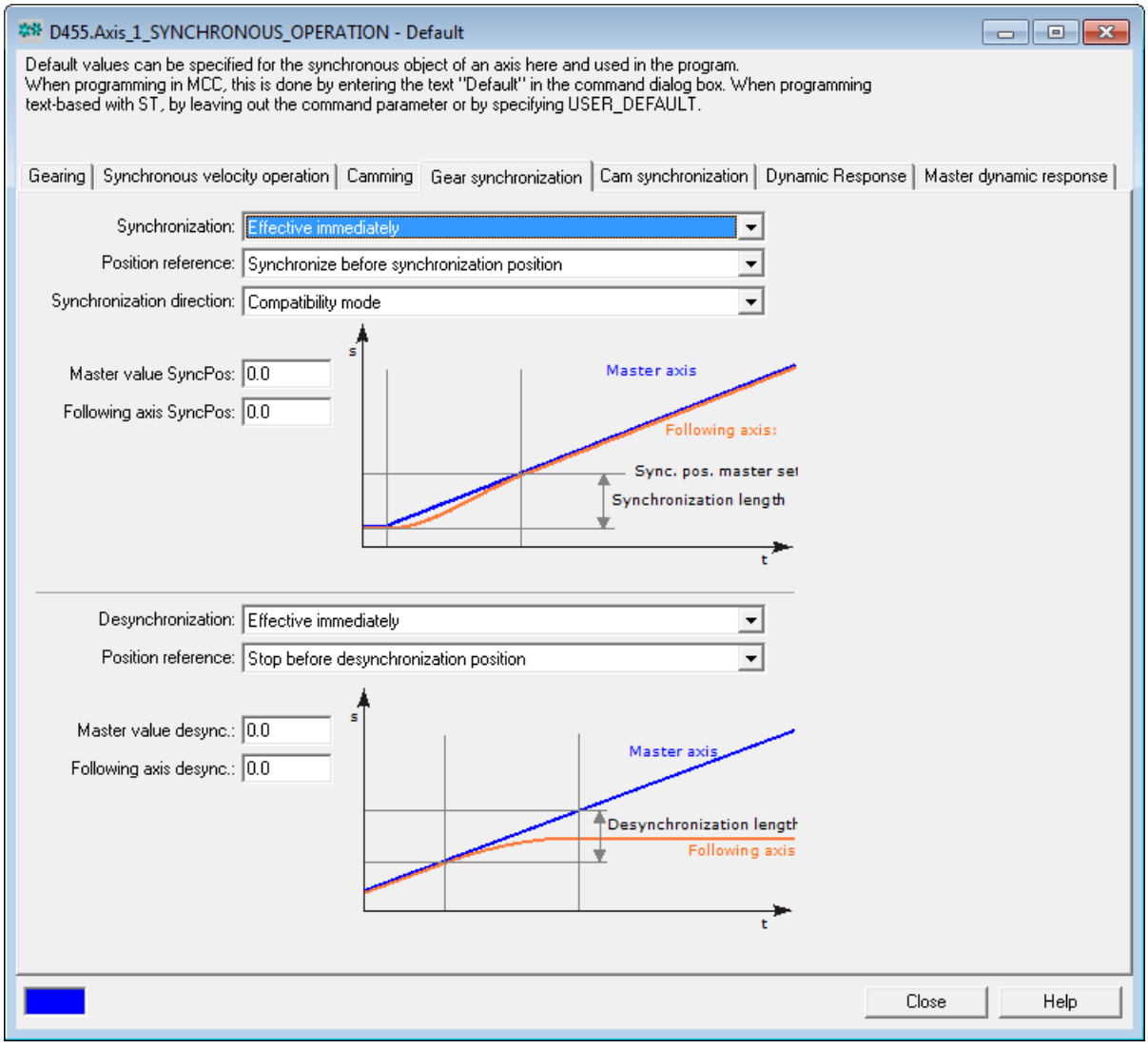


Figure 2-85 Synchronous object: Default setting for gear synchronization

Gear Synchronization - Default

You use the **Gear synchronization** tab to set the parameters for **synchronization and desynchronization**. These settings are only relevant when **gearing** mode is used.

You can set the following parameters:

Table 2-22 Parameters for gear synchronization

Field/Button	Meaning/Instruction
Synchronization	Specify when synchronization of the following axis with the master value is performed. For more information, see Synchronizing the gear (Page 40)
Position reference	Here, you specify the position of the synchronization profile relative to the position of the synchronization point. For more information, see Position reference during synchronization (Page 122)
Synchronization direction	Specify the synchronization direction here. See Synchronization direction (Page 46).
Master value SyncPos	Here, you enter the position of the synchronization point of the master value.
SyncPos slave axis	Here, you enter the position of the synchronization point of the slave axis
Desynchronization	Specify when desynchronization of the following axis with the master value is performed. For more information, see Desynchronizing the gear (Page 123).
Position reference	Here, you specify the position of the desynchronization profile relative to the position of the desynchronization point For more information, see Position reference during desynchronization (Page 123).
Desync. Master value	Here, you enter the position of the desynchronization point of the master value.
Desync. Following axis	Here, you enter the position of the desynchronization point for the following axis

2.3.3.5 Synchronizing the gear

You can set the synchronization condition using the **Synchronization** drop-down list box on the **Gear synchronization** tab:

Table 2-23 Parameters for gear synchronization

Setting	Meaning
Effective immediately	Synchronization takes immediate effect. The start point is derived from the position of the current master value. The settings in **Master value SyncPos** and **Following axis SyncPos** are not evaluated.
Default synchronization position of the leading axis	This setting is only appropriate for an absolute master value coupling. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. The setting in **Following axis SyncPos** is not evaluated.

Setting	Meaning
Specified by the synchronization position of the following axis	This setting is only appropriate for an absolute following axis. The synchronization criterion depends on the position of the following axis. The synchronization position is specified in **Following axis SyncPos**. The setting in **Master value SyncPos** is not evaluated.
Default synchronization position of the leading axis and following axis	This setting is only appropriate for an absolute master and an absolute following axis. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. An offset is additionally created on the following axis by the setting in **Following axis SyncPos**. That is, the following axis is not synchronized to the programmed slave position but to the position **Following axis SyncPos** plus absolute position value of the following axis.
Last programmed setting	Setting of the last active command

See also

Synchronization criterion/synchronization position (Page 42)

2.3.3.6 Position reference during synchronization

How synchronization is to performed is set via the **Position reference** picklist:

Table 2-24 Position reference parameters for synchronization

Setting	Meaning
Synchronize from synchronization position	Synchronization starts at the synchronization position. The following axis runs synchronously after the synchronization length resulting from the dynamic response data.
Synchronize before synchronization position	The synchronization is performed so that the following axis runs synchronously to the master value at the synchronization position. If the following axis is stopped, it is accelerated before the synchronization position so that it runs synchronously to the master value at the synchronization position. The position at which the axis is accelerated is the synchronization position minus the synchronization length.
Synchronize symmetrically to synchronization position	The synchronization is performed so that the synchronization position is exactly in the middle of the synchronization length. Therefore, a stopped following axis is already accelerated before the synchronization position and only runs synchronously to the master value after half the synchronization length.
Last programmed setting	Setting of the last active command

You will find further information in the section Position of synchronization range relative to synchronization position (Page 48).

2.3.3.7 Desynchronizing the gear

You can set the position at which desynchronization is to start using the **Desynchronization** drop-down list box on the **Gear synchronization** tab:

Table 2-25 Parameters for desynchronization

Setting	Meaning
Effective immediately	Desynchronization takes immediate effect.
Specified by the desynchronization position of the leading axis	The desynchronization is performed as of the **Master value desync** value. The setting in **Following axis desync** is not evaluated.
Default desynchronization position of the following axis	The desynchronization is performed as of the **Following axis desync** value of the following axis. The setting in **Master value desync** is not evaluated.
Last programmed setting	Setting of the last active command

See also

Desynchronization criterion/desynchronization position (Page 64)

2.3.3.8 Position reference during desynchronization

How desynchronization is to performed is set via the **Position reference** picklist:

Table 2-26 Parameters for the position reference

Setting	Meaning
Stop after desynchronization position	The desynchronization is started at the desynchronization position. The desynchronization is completed after the synchronization length resulting from the dynamic response data.
Stop before desynchronization position	The desynchronization is performed so that the desynchronization is completed at the desynchronization position. The position at which the desynchronization is started is the desynchronization position minus the desynchronization length.
Stop symmetrically to desynchronization position	The desynchronization is performed so that the desynchronization position is exactly in the middle of the desynchronization length.

See also

Position of synchronization range relative to desynchronization position (Page 65)

2.3.3.9 Camming synchronization

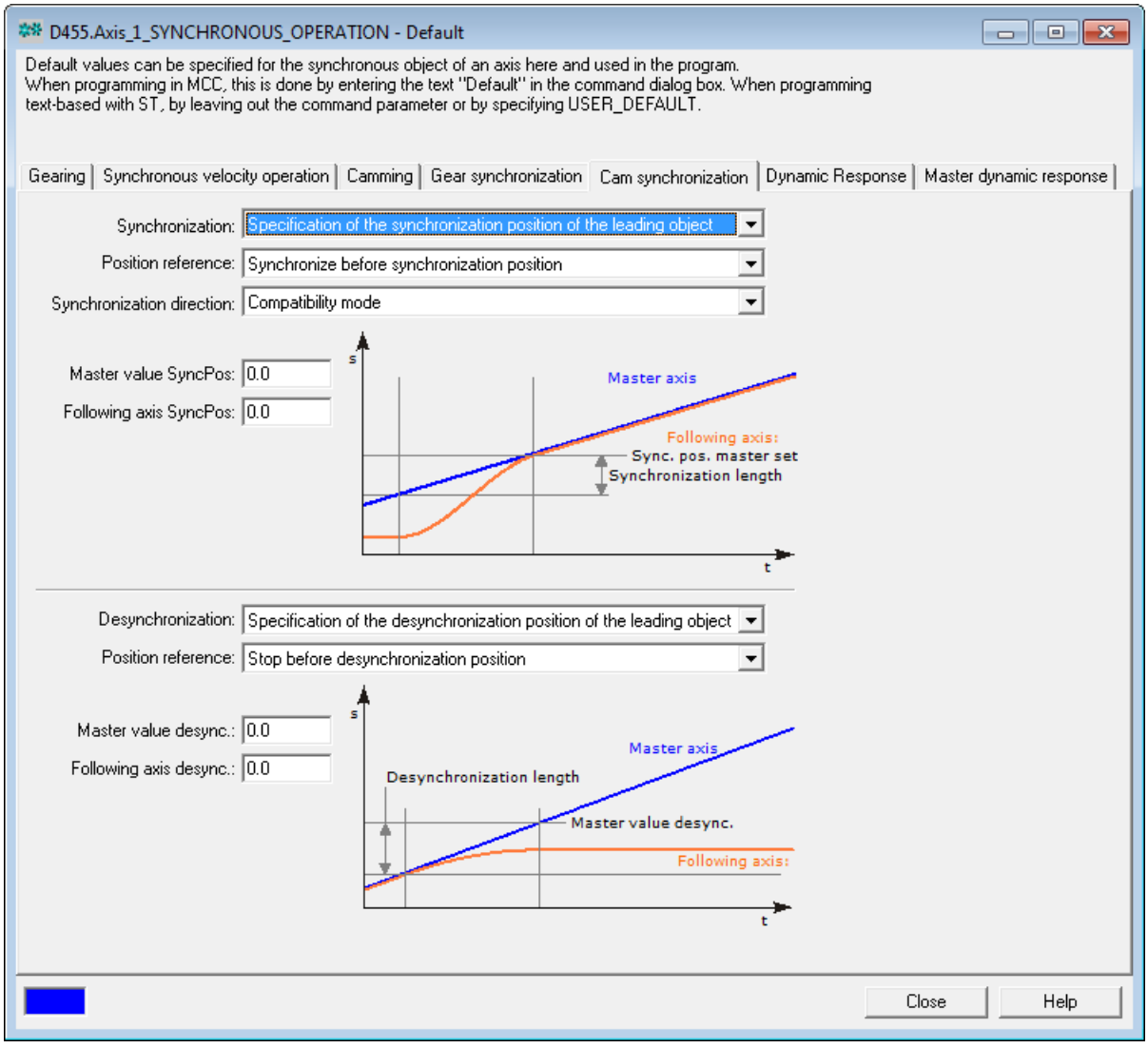


Figure 2-86 Synchronous object: Default setting for cam synchronization

Cam Synchronization - Default

You use the **Cam synchronization** tab to set the parameters for **synchronization and desynchronization**. These settings are only relevant when **camming** mode is used.

You can set the following parameters:

Table 2-27 Parameters for camming synchronization

Field/Button	Meaning/Instruction
Synchronization	Specify when synchronization of the following axis with the master value is performed. For more information, see Synchronizing the curve (Page 125).
Position reference	Here, you specify the position of the synchronization profile relative to the position of the synchronization point. For more information, see Position reference during synchronization (Page 122)
Synchronization direction	Specify the synchronization direction (Page 46) here.
Master value SyncPos	Here, you enter the position of the synchronization point of the master value.
SyncPos slave axis	Here, you enter the position of the synchronization point of the slave axis
Desynchronization	Specify when desynchronization of the following axis with the master value is performed. For more information, see Desynchronizing the curve (Page 126).
Position reference	Here, you specify the position of the desynchronization profile relative to the position of the desynchronization point For more information, see Position reference during desynchronization (Page 123).
Desync. Master value	Here, you enter the position of the desynchronization point of the master value.
Desync. Following axis	Here, you enter the position of the desynchronization point for the following axis

See also

Synchronization (Page 40)

2.3.3.10 Cam synchronization

You can set the synchronization condition using the **Synchronization** drop-down list box on the **Cam synchronization** tab:

Table 2-28 Parameters for synchronization

Setting	Meaning
Effective immediately	Synchronization takes immediate effect. The start point is derived from the position of the current master value. The settings in **Master value SyncPos** and **Following axis SyncPos** are not evaluated.
Default synchronization position of the leading axis	This setting is only appropriate for an absolute master value coupling. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. The setting in **Following axis SyncPos** is not evaluated.

Setting	Meaning
Transition at the end of the active cam	This setting is only possible for a relative master value coupling. The synchronization criterion is the master setpoint position at the end of the current cam disk cycle. The setting in **Following axis SyncPos** is not evaluated.
Default synchronization position of the leading axis and following axis	This setting is only appropriate for an absolute master and following axis. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. An offset is additionally created on the following axis by the setting in **Following axis SyncPos**. That is, the following axis is not synchronized to the programmed (e.g. via cam) slave position but to the position **Following axis SyncPos** plus absolute position value of the following axis in relation to the cam.
Last programmed setting	Setting of the last active command

See also

Synchronization criterion/synchronization position (Page 42)

2.3.3.11 Cam desynchronization

You can set the position at which desynchronization is to start using the **Desynchronization** drop-down list box on the **Cam synchronization** tab:

Table 2-29 Parameters for desynchronization

Setting	Meaning
Effective immediately	Desynchronization takes immediate effect.
At position of the leading axis	The desynchronization is performed as of the **Master value desync** value. The setting in **Following axis desync** is not evaluated.
At position of the following axis	The desynchronization is performed as of the **Following axis desync** value of the following axis. The setting in **Master value desync** is not evaluated.
End of cam cycle	The desynchronization is performed at the end of the current cam cycle.
Last programmed setting	Setting of the last active command

See also

Desynchronization criterion/desynchronization position (Page 64)

2.3.3.12 Dynamic response

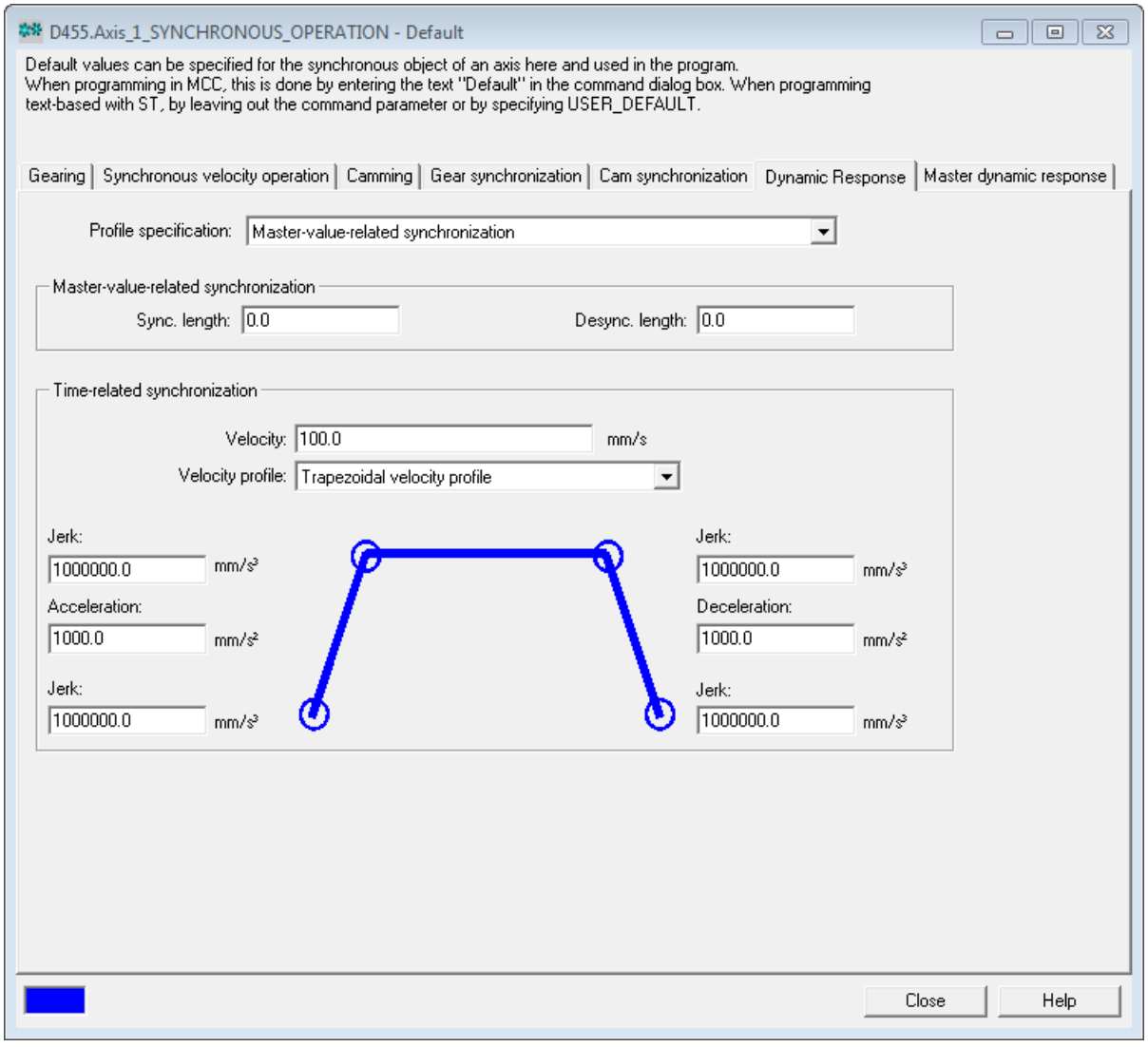


Figure 2-87 Synchronous object: Default setting for dynamic response

Dynamic response - Default

In the **Dynamics** tab, you make the basic settings for synchronization and desynchronization.

You can enter the following profile settings:

- **Length-related synchronization**; see Synchronization via a specifiable master value distance (Page 81).
- **Time-related synchronization**; see Synchronization profile based on specifiable dynamic response parameters (Page 83).

Dynamic response parameters are used for:

- Time-related synchronization
- Compensatory motions on the synchronous object

You can set the following parameters:

Table 2-30 Dynamic response parameters

Field/Button	Meaning/Instruction
Profile specification	Specify the reference for the synchronization profile here.
Leading axis-related synchronization	
Sync. Length	Here, you enter the path length for synchronization.
Desync. Length	Here, you enter the path length for desynchronization.
Time-related synchronization	
Velocity profile	Select the velocity profile here.
Velocity	Enter the maximum velocity here.
Acceleration	Enter the maximum acceleration here.
Deceleration	Enter the maximum deceleration here.
Jerk	Enter the maximum jerk here.

The synchronization length and desynchronization length are only evaluated for master-axis-related synchronization profiles. Velocity profile, velocity, acceleration, deceleration and jerk are only evaluated for time-related synchronization profiles.

See also

Synchronization (Page 40)

2.3.3.13 Master dynamic response

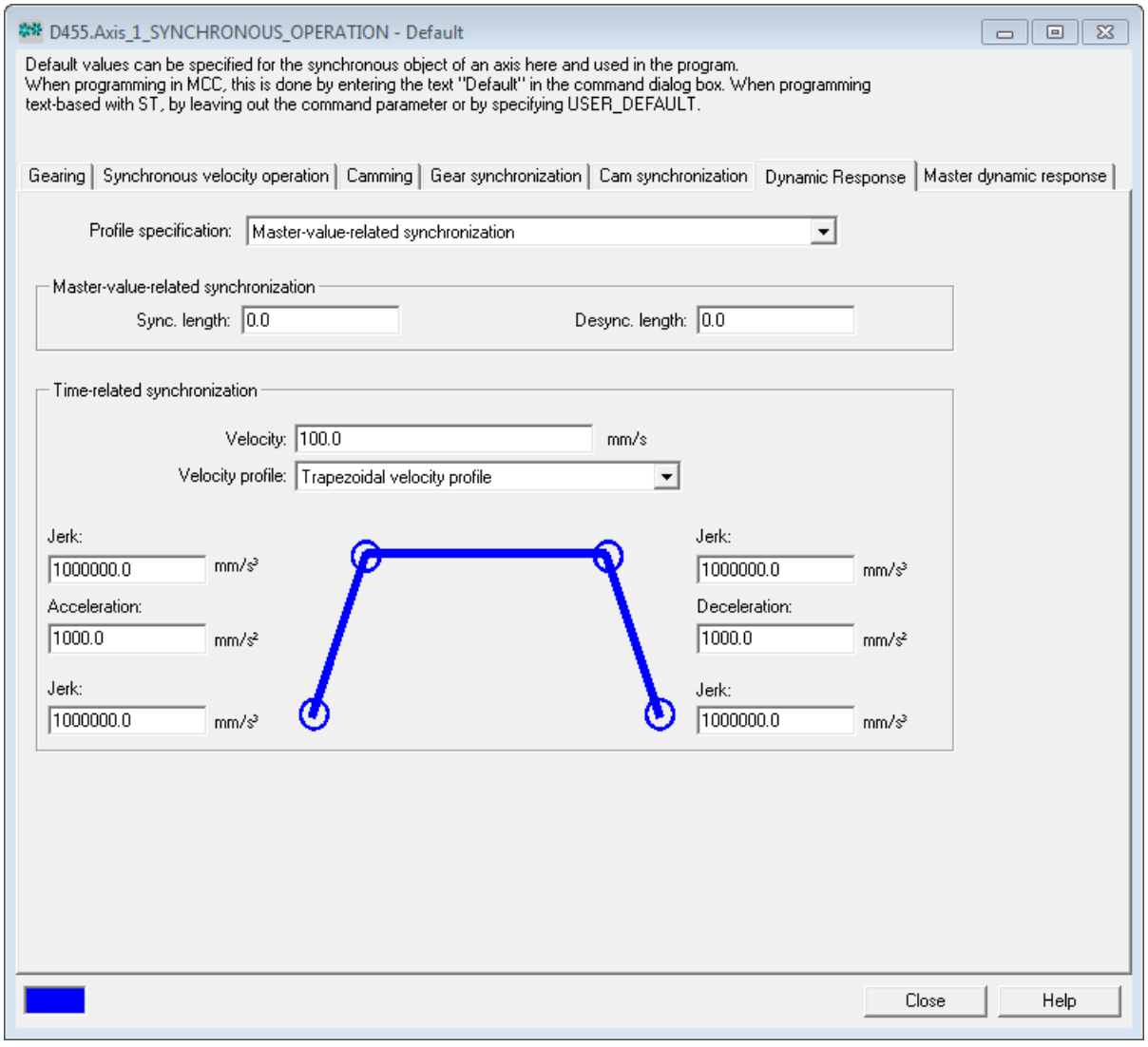


Figure 2-88 Synchronous object: Default setting for master dynamic response

Master dynamic response - default

In the **Master dynamic response** tab, you make the settings for the dynamic response for the master value switchover.

You can set the following parameters:

Table 2-31 Dynamic response parameters - Master

Field/Button	Meaning/Instruction
Master switchover with dynamic response values	
Velocity profile	Select the velocity profile here.
Velocity	Enter the maximum velocity here.

Field/Button	Meaning/Instruction
Acceleration	Enter the maximum acceleration here.
Deceleration	Enter the maximum deceleration here.
Jerk	Enter the maximum jerk here.

See also

Switching over the master value source – Overview (Page 68)

2.3.4 Set synchronization

Certain settings for synchronization can be defined on the synchronous object.

- In the project navigator, double-click **Settings** under the synchronous object.

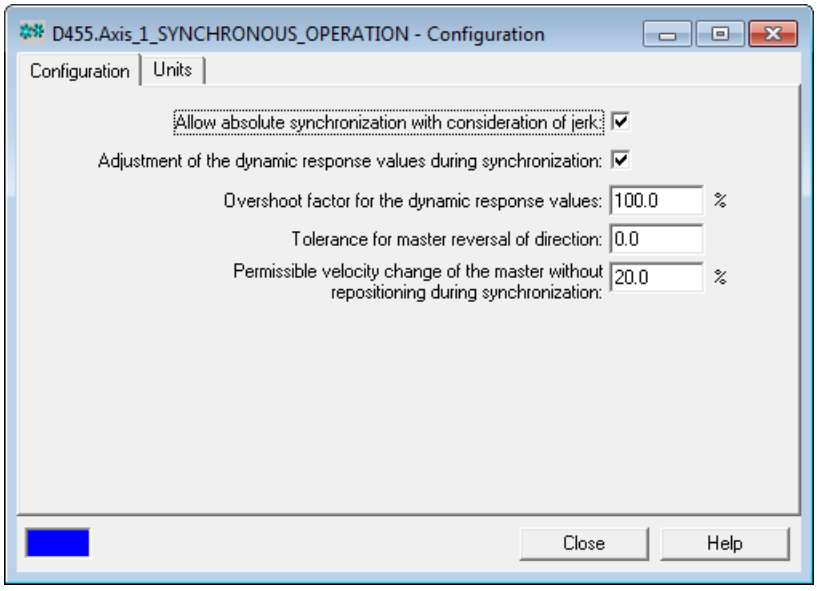


Figure 2-89 Settings on the synchronous object

Synchronous object - Settings

In this window, define the parameters for the synchronization.

Table 2-32 Synchronization parameters

Field/Button	Meaning/Instruction
Permit absolute synchronization with provision for jerk	For absolute synchronization, the "Yes" setting of the jerk is used. If "No" is set, the set jerk will not be used in spite of velocity profile = SMOOTH being selected. Travel follows a trapezoidal path. (syncingMotion.smoothAbsoluteSynchronization) See also section Trailing synchronization with synchronization profile based on dynamic response parameters (Page 53).
Adapt the dynamic response values for the synchronization	Adaptation to the dynamic response in the synchronous position (syncingMotion.synchronizingAdaption) If "Yes" is set, the following parameter is available: See also section Synchronization profile based on specifiable dynamic response parameters (time reference) (Page 50).
Magnification factor for the dynamic response values	Overdrive factor for the adapted dynamic response values to compensate for a remaining path difference (syncingMotion.overdriveFactor) Value as percentage (%) Reference to the current master value velocity for the synchronization start. See also section Synchronization profile based on specifiable dynamic response parameters (time reference) (Page 50) and section Adapt the synchronization velocity to the master value velocity (Page 107).
Tolerance for master direction reversal	Tolerance window for canceling the synchronization for direction reversal of the master values (syncingMotion.masterReversionTolerance) Position value in the user unit of the master values. See also section Settings for evaluation of the master value behavior during synchronization (Page 55).
Permitted velocity change of the master without restarting for synchronization	Maximum permitted change of the master value velocity (syncingMotion.maximumOfMasterChange) Refers to the current master value velocity for the synchronization start. Value as percentage (%). See also section Settings for evaluation of the master value behavior during synchronization (Page 55).

See also

Dynamic response effect on slave values (Page 66)

Synchronization (Page 40)

2.3.5 Configuring synchronous operation monitoring

Synchronous operation monitoring between the master value/following object and the following axis can be configured on the synchronized axis.

1. In the project navigator, double-click **Monitoring** under the axis object.
2. Set the required parameters on the **Synchronous operation** tab.

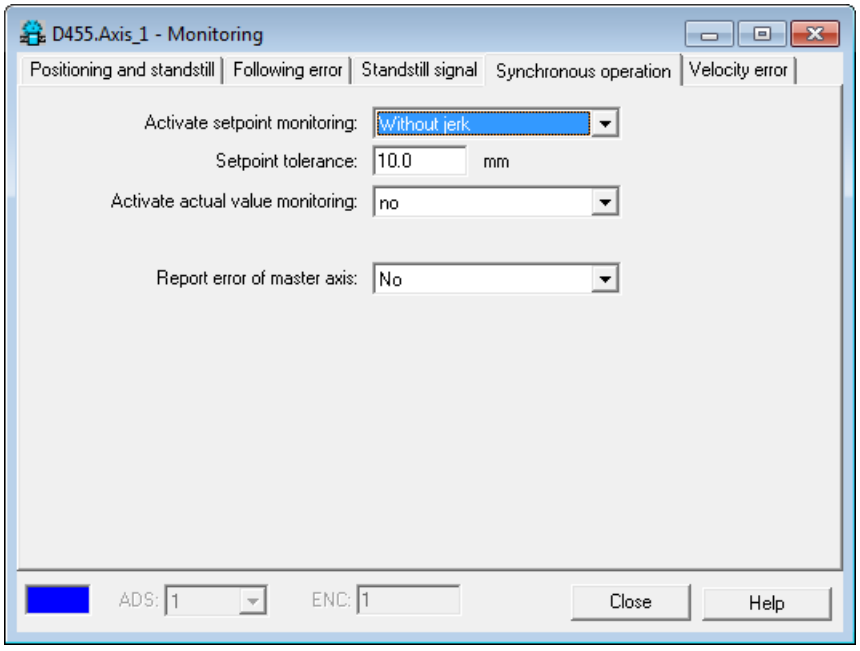


Figure 2-90 Monitoring of a following axis

Here, you specify the synchronous operation monitoring of the axis.

Table 2-33 Monitoring parameters

Field/Button	Meaning/Instruction
Activate setpoint monitoring	Activate the setpoint monitoring of the following axis here. (TypeOfAxis.GearingPosTolerance.enableCommandValue)
Setpoint tolerance	Specify the setpoint tolerance with activated setpoint monitoring. (TypeOfAxis.GearingPosTolerance.commandValueTolerance)
Activate actual value monitoring	Activate the actual value monitoring of the following axis here. (TypeOfAxis.GearingPosTolerance.enableActualValue)
Actual value tolerance	Specify the actual value tolerance with activated actual value monitoring. (TypeOfAxis.GearingPosTolerance.actualValueTolerance)
Signal error of the leading axis	Specify here which tolerance-exceeded messages (actual values / setpoints) are signaled to the leading axis. Take into account that the tolerance-exceeded messages are signaled to all higher-level leading axes (if, for example, the leading axis is also a following axis in a synchronous operation configuration). (TypeOfAxis.GearingPosTolerance.enableErrorReporting)

See also

Synchronous operation monitoring (Page 75)

Error handling (Page 143)

2.4 Synchronous Operation Programming/References

This chapter contains an overview of the **Synchronous Operation** technology object commands and information on the local alarm response. You will find a complete list of all commands and their syntax, the system variables, and error messages, in the SIMOTION Reference Lists.

See also

Overview of commands (Page 133)

Command processing (Page 138)

Error handling (Page 143)

Menus (Page 146)

2.4.1 Overview of commands

ST commands

Table 2-34 Commands on the synchronous object

Command type / command	Description
Information and conversion	See Commands for reading out function values (Page 135).
_getMasterValue	The **_getMasterValue** command supplies the master value at a specified slave value position.
_getSlaveValue	The **_getSlaveValue** command supplies the slave value on a specified master value position.
_setMaster	The **_setMaster** command is used to assign a new master value source to a synchronous object. See Switching of the master value source (Page 68).
Command tracking	See Commands for command tracking (Page 136).
_getStateOfFollowingObjectCommand	The **_getStateOfFollowingObjectCommand** command returns the execution state of a synchronous operation command.
_getMotionStateOfFollowingObjectCommand	The **_getMotionStateOfFollowingObjectCommand** command returns a structure with the state of a synchronous operation command. The state of a synchronous operation command must then also be read out.

Command type / command	Description
_bufferFollowingObjectCommandId	The **_bufferFollowingObjectCommandId** command enables the **commandId** and the associated command status to be saved beyond the processing time of the command. The **commandId** parameter is used to define the command for which the respective status is to be saved. The maximum number of command states that can be saved is set in configuration data **decodingConfig.numberOfMaxBufferedCommandId**.
_removeBufferedFollowingObjectCommandId	The **_removeBufferedFollowingObjectCommandId** command discontinues saving of the **commandId** and the associated command status beyond the processing time of the command.
Motion	
_enableGearing	The **_enableGearing** command produces gearing with a constant transfer ratio. See Gearing (Page 20).
_disableGearing	The **_disableGearing** command terminates gearing with a constant transfer ratio.
_enableVelocityGearing	The **_enableVelocityGearing** command produces velocity gearing with a constant coupling. See Velocity synchronous operation (Page 25).
_disableVelocityGearing	The **_disableVelocityGearing** command terminates velocity gearing with a constant coupling.
_enableCamming	The **_enableCamming** command produces camming with a variable transfer ratio. See Camming (Page 26).
_disableCamming	The **_disableCamming** command terminates camming with a variable transmission ratio.
Correction and superimposition	
_setGearingOffset	The **_setGearingOffset** command offsets the gearing with reference to the master value or the slave value (V3.1 and higher). See **Changing the offset** in chapter Gearing (Page 20)
_setCammingScale	The **_setCammingScale** command scales the camming relative to the master value or the slave value. See **Scaling and offset** in Camming (Page 26).
_setCammingOffset	The **_setCammingOffset** command offsets the camming with reference to the master values or the slave values. See **Scaling and offset** in Camming (Page 26).
Object and Alarm Handling	See Commands for resetting of states and errors (Page 138).
_resetFollowingObject	The **_resetFollowingObject** command resets the synchronous object, i.e. an active synchronous grouping will be cancelled, any pending errors will be cleared.
_resetFollowingObjectError	The **_resetFollowingObjectError** command acknowledges and resets errors on the synchronous object.
_resetFollowingObjectErrorState	The **_resetFollowingObjectErrorState** command acknowledges and resets the error status on the synchronous object.
_getFollowingObjectErrorNumberState	The **_getFollowingObjectErrorNumberState** command is used to fetch the status of a specific error
Simulation	See Simulation mode (Page 78).

Command type / command	Description
_enableFollowingObjectSimulation	The **_enableFollowingObjectSimulation** command sets a synchronous operation into simulation mode.
_disableFollowingObjectSimulation	The **_disableFollowingObjectSimulation** command disables simulation mode of the synchronous grouping.

PLCopen commands

MultiAxis commands of PLCopen are relevant for synchronous operation. These commands are provided for use in cyclic programs/tasks and enable motion control programming in a view shaped by a PLC. They are used primarily in the LAD/FBD programming language. The commands are certified in accordance with "PLCopen Compliance Procedure for Motion Control Library V1.1".

Table 2-35 PLCopen commands for synchronous operation

Command	Description
_MC_CamIn	Synchronize and engage cam; included implicitly:
_MC_CamOut	Remove cam with desynchronization
_MC_GearIn	Synchronize synchronous operation
_MC_GearOut	Desynchronize synchronous operation
_MC_Phasing	Phase offset

For further information, refer to the **PLCopen Blocks** Function Manual.

MCC commands

MCC commands are available for synchronous operation. For information on this see the Programming and Operating Manual for the **SIMOTION MCC Motion Control Chart**.

2.4.1.1 Commands for reading out function values

Commands that calculate values and supply them as return values are available for reading out master value positions and slave value positions in pairs:

- The **_getSlaveValue** command supplies the slave value at a specified master value position. If several master values may have the same slave value, an approximate value can be specified for the master value.
  - **slavePositionType:=CURRENT** returns the current value.
  - **slavePositionType:=DIRECT** returns in **slavePosition** the specified approximation value.
- The **_getMasterValue** command supplies the master value at a specified slave value position. If several slave values may have the same master value, an approximate value can be specified for the slave value.
  - **masterPositionType:=CURRENT** returns the current value.
  - **masterPositionType:=DIRECT** returns the approximate value specified in **masterPosition**.

The commands only supply the correct position if a synchronous relationship is active and synchronized.

2.4.1.2 Commands for command tracking

The following commands are available for command tracking:

- The **_getStateOfFollowingObjectCommand** command returns the execution state of a synchronous operation command:
  - **ACTIVE**: The command is being executed.
  - **NON_EXISTENT**: The command is complete or the commandID is unknown.
  - **WAITING**: The command is decoded, but execution not yet started.
  - **WAITING_FOR_SYNC_START**: The command is waiting for synchronous start.
- The **_getMotionStateOfFollowingObjectCommand** command returns a structure with the execution state of a command.
  - **functionResult** specifies the error code.
  - **motionCommandIdState** returns the current phase of the motion of the queried command.
  - **abortId** specifies the reason for aborting the command.
    The reason for aborting is specified with alarm 30002 "Command aborted (reason: <abortId>, command type: ...)".
- With **_bufferFollowingObjectCommandId**, the command status can be queried after completion or an abort of the command.
- With **_removeBufferedFollowingObjectCommandId**, the command should be explicitly removed from the command management of the technology object after evaluation is complete.

The number of motion commands that the MotionBuffer can accept can be specified using the **followingObjectType.DecodingConfigInfo.numberOfMaxbufferedCommandId** configuration data.

System variables

The sequence of programmed commands can be read out by means of system variables; see Monitoring the synchronization (Page 59).

System variable syncMonitoring:

The structure elements belonging to **syncMonitoring** indicate the monitoring functions and status information for synchronization.

The variable is used for the monitoring of the synchronous operation relationship to the synchronous operation setpoint, see also Synchronous operation monitoring (Page 75).

- The **limitCommandValue** variable indicates whether the setpoint differences have exceeded the maximum limit for synchronous operation errors.
- The **differenceCommandValue** variable shows the difference between the setpoint generated at the synchronous object and the executable setpoint at the axis, with dynamic limits taken into account.
- The **limitActualValue** variable indicates whether the actual value differences have exceeded the maximum limit for synchronous operation errors.
- The **differenceActualVelocity** variable shows the pending position lag at the following axis, including the setpoint error, caused by any acting dynamic response of the following axis (**syncMonitoring.differenceCommandValue**). The dynamic response also acts on superimposed motions and can possibly trigger monitoring of the actual value. The position lag for DSC operation refers to the position lag present at the position controller in the drive. For versions prior to V4.2, the difference between the setpoint calculated on the synchronous object and the actual value present in the interpolator is indicated.
- The **syncErrorReported** variable indicates whether an alarm has already been issued to the master axis.
- The **syncState** variable indicates the synchronous operation status of the setpoint; it is syncState:=NO during synchronization and desynchronization. The setpoint synchronous operation error is the error that arises from limiting the setpoint to the maximum slave axis dynamic response values for the synchronous operation component.
  The **followingMotionState** indicates the state of the synchronous operation motions.
  The **differenceCommandVelocity** variable shows the difference, during velocity synchronous operation, between the set velocity calculated on the synchronous object and the set velocity that is executable on the axis taking into account the dynamic limits.

Canceling/deleting a synchronous operation command

The **_cancelFollowingObjectCommand** command can be used to cancel a synchronous operation command by specifying the **commandId** in the **commandToBeCancelled** parameter (as of V4.1). This removes the synchronous operation command from the command buffer and, if necessary, also from the IPO.

The synchronous motion is canceled with the local response **FOLLOWING_OBJECT_DISABLE**; see also **_cancelAxisCommand/_cancelExternalEncoderCommand** for the Axis/External Encoder technology object.

Additional references

You can find detailed information in both the **Motion Control Technology Objects Axis Electric/ Hydraulic, External Encoder** Function Manual and the **SIMOTION reference lists**.

2.4.1.3 Commands for resetting states and errors

The following commands are available for resetting states and errors:

- The **_resetFollowingObject** command resets the synchronous object, i.e. an active synchronous operation relationship is canceled and any pending errors cleared.
  The command can be executed when an error response (e.g. **DECODE_STOP**) is pending. All active commands and commands pending in the buffer are canceled. Changed default values can be reset using the **userDefaultData:=ACTIVATE_CONFIGURATION_DATA** parameter. Canceling commands by resetting the synchronous operation function does not generate any warnings.
  The **deleteSynchronizingCommandsOnly** parameter (V3.2 and higher) can be used to remove the waiting and active commands directly, without the need to reset the entire technology object.
- The **_resetFollowingObjectError** command acknowledges and resets errors on the synchronous object.
  The command can be executed when an error response (e.g. **DECODE_STOP**) is pending. One particular error or all errors can be ordered to be reset. The error response changes to the response with the highest priority of the still pending errors. It is terminated with a negative acknowledgment for any errors that cannot be acknowledged at this point. If all errors can be reset, the error response changes to **NONE**.
- The **_getFollowingObjectErrorNumberState** command is used to fetch the status of a specific error

2.4.2 Command processing

2.4.2.1 Command execution

Command advance

A criterion for the command advance is specified in the commands. If the condition for the command advance is satisfied, the next command in the user program is executed.
*****Specifying a command advance condition in a command influences the time at which the next programmed command will be executed.

Possible step enabling conditions on the synchronous object

The step enabling condition is indicated in the command parameter **nextCommand**.

Table 2-36 Step enabling conditions on synchronous object commands

Step enabling condition for next command	Time of advance
IMMEDIATELY	As soon as the command is issued
WHEN_BUFFER_READY	When the command enters the command buffer
AT_MOTION_START	When the command changes over to the interpolator
WHEN_ACCELERATION_DONE	When the acceleration phase is complete

Step enabling condition for next command	Time of advance
AT_DECELERATION_START	When deceleration starts
WHEN_INTERPOLATION_DONE	When setpoint interpolation for this command is complete
WHEN_AXIS_SYNCHRONIZED	When axis is synchronized with the master value
WHEN_MOTION_DONE	When motion generation is complete

Group classification of commands and command buffers

Each synchronous object has two dedicated command buffers for each command, which can be fetched for each IPO cycle. These command buffers are the buffer for synchronous operation commands and the buffer for *parallel effective commands*. The behavior of the buffers corresponds to the behavior of the axis.

Commands are classified in groups as follows:

- **Group 1**: *Synchronous operation commands*
  The following commands belong to this group: **_enableGearing**, **_disableGearing**, **_enableVelocityGearing**, **_disableVelocityGearing**, **_enableCamming**, and **_disableCamming**
  Depending on the transition behavior, new commands to be entered may behave in the following ways if the buffer is full:
  - Return with an error
    (**nextCommand** = **IMMEDIATELY** and **mergeMode** = **SEQUENTIAL**),
  - Wait for the buffer to become available
    (**nextCommand** ≠ **IMMEDIATELY** and **mergeMode** = **SEQUENTIAL**) or
  - Supersede the command pending in the buffer
    (**mergeMode** = **IMMEDIATELY**).
- **Group 2**: *Concurrently active commands*
  Commands that supersede each other in the command buffer. These commands are executed in the IPO. If there are more than one parallel effective command, only the last one is executed, because these commands overwrite each other. In this case, technology alarm 30002 "Command aborted (reason: ..., command type: ...)" is output.
  The following commands belong to this group: **_setGearingOffset**, **_setCammingScale**, **_setCammingOffset**, **_enableFollowingObjectSimulation**, **_disableFollowingObjectSimulation**, and **_setMaster**
- **Group 3**: Directly executed commands without entry in a command buffer
  Commands in this group do not abort each other.
  This group includes:
  _getSlaveValue, _getMasterValue,
  _resetFollowingObjectError,
  _getStateOfFollowingObjectCommand,
  _getMotionStateOfFollowingObjectCommand,
  _bufferFollowingObjectCommandID,
  _removeBufferFollowingObjectCommandId,
  _resetFollowingObject,
  _resetFollowingObjectConfigDataBuffer,
  _getFollowingObjectErrorNumberState

Command execution of sequential commands in the IPO cycle clock

If enabled to this effect, the commands are read out from the command buffer in every interpolation cycle. Within the slave value generation, up to two synchronous operation commands (_enableGearing(), _disableGearing(), _enableVelocityGearing(), _disableVelocityGearing(), _enableCamming(), _disableCamming()) are active at the same time and are executed in the IPO.

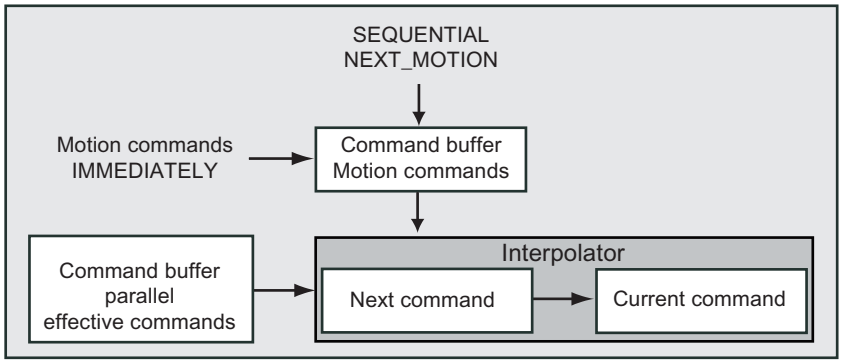


Figure 2-91 Command buffer and execution of sequential commands

Behavior regarding programming of a motion command:

- No motion command is active:
  An **_enable...** command is processed as the current command in the interpolator first as a waiting command, while a **_disable...** command is aborted immediately. If the synchronization criterion is satisfied, the waiting command changes its state to active (synchronizing/synchronous).
- If a motion command is waiting or active, the new motion command is processed as the next command in the interpolator first as a waiting command. A new **_disable...** command is immediately canceled if the current command is not a complementary **_enable...** command. If the next command is active, the current command is canceled. The next command is thereafter processed as the current command.
- If two motion commands are active or waiting and **mergeMode:=SEQUENTIAL** or **mergeMode:= NEXT_MOTION**, the new command is prevented from being read in until at least one command has been aborted or completed. With **mergeMode:= IMMEDIATELY**, the next command up to that point is aborted and replaced with the new command. Exception: If the new command is a complementary **_disable...** command for the next command, the new command is canceled immediately and only the current command is waiting/active.

See Command transition conditions (Page 141) for further information.

Behavior regarding programming of a parallel effective command:

- **_setCammingScale()/_setCammingOffset()/_setGearingOffset()** act on the current (**_enable...**) command when **activationMode:= ACTUAL_VALUE** or **activationMode:= ACTUAL_AND_DEFAULT_VALUE**.
  If no command is active or the active command does not correspond to the correction being executed, the parallel effective command is canceled.
- In general, commands for simulation mode and resetting of current master value source are active.

2.4.2.2 Command transition conditions

The transition behavior is set in the **mergeMode** command parameter, on the commands of the synchronous object. The transition behavior affects the execution of the queued commands on the synchronous object. An active command can thus affect the execution of a command of another task.

The **mergeMode** on the synchronous object commands also determines the behavior of the following axis commands.

- A command issued with **mergeMode** = **IMMEDIATELY** clears the command buffer and overwrites the IPO (next command). The current command is executed. The next command is substituted.
- A command issued with **mergeMode** = **NEXT_MOTION** will be executed once the active command is complete and the pending commands have been deleted. It overwrites the command buffer.
- A command issued with **mergeMode** = **SEQUENTIAL** will be executed after the completion of the active command and the motion. The command will be entered in the command buffer when it is empty; if the command buffer is not empty, the command waits.

Influences between axis and synchronous object

Table 2-37 Transition behavior on synchronous object commands

Transition behavior for synchronous motion	Effect
IMMEDIATELY	The active command is aborted. The starting point for the slave value in the synchronous operation is the current axis setpoint. If the synchronous motion is active, the second synchronous function to be programmed or the non-active synchronous function is overwritten. If the synchronous function is already effective, then it remains effective. An active motion function in the following axis is overridden. It is therefore possible for motion, positioning, and synchronous operation to override each other another. If a synchronous operation command is issued with **mergeMode:=IMMEDIATELY**, the command waits in the command buffer until the synchronization criterion is fulfilled. Only then will the command enter the **active** state. A synchronous operation command will only be aborted by another motion command if the motion command is already active. In the same way, motion commands that are already active will only be aborted once a waiting synchronous operation command is activated.
SEQUENTIAL, NEXT_MOTION	If a main motion is active on the following axis, this main motion is first carried out completely. The starting point is the current axis setpoint for the main motion or, in the case of multiple synchronous commands, the internally generated slave value. If a synchronous motion is already active, the synchronous function is set to wait until the active synchronous function is completed or aborted. This requires the **nextCommand=WHEN_BUFFER_READY** setting. When **nextCommand=IMMEDIATELY** is set, the command is not executed and is returned with error information in the return value. The synchronous function is added to an active motion function in the following axis. A synchronous function that is in effect but not yet active can be deleted • With the **_disable...** command with **mergeMode=IMMEDIATELY** and **Synchronization criterion=IMMEDIATELY** or • With the **_resetFollowingObject** command

Table 2-38 Transition behavior on axis commands

Transition behavior for axis motion	Effect on the synchronous object
IMMEDIATELY	All active synchronous motions are aborted. None of the synchronous motions that are not active (i.e. they are neither in the synchronization motion nor in the "synchronous" status, but are instead waiting for the synchronization criterion) are aborted.
SEQUENTIAL, NEXT_MOTION	The synchronous operation commands/synchronous motions are not aborted and the axis motion only becomes active once the synchronous operation is complete.

The following special feature applies when motion commands are pending on the synchronous object and on the axis in the same interpolation cycle clock:

- If **mergeMode**(axis)=**SEQUENTIAL**, the synchronous command is executed
- If **mergeMode**(axis)=**IMMEDIATELY**, the axis command is executed

2.4.3 Error handling

2.4.3.1 Local alarm reaction on the synchronous object

Local **alarm responses** are specified by means of the system.

The following responses are possible:

- **NONE**
  No response
- **DECODE_STOP**
  Abort of the command preparation, the current synchronous operation function remains **active**.
- **FOLLOWING_OBJECT_DISABLE**
  Abort of the command preparation, abort of the current synchronous operation function.

An error can be reset with **_resetFollowingObject** or **_resetFollowingObjectError**

Note

The stop responses are listed in order of increasing priority. Global alarm responses can be set in the **alarm configuration** in the technology object; these can also be set to require **Power On**. For further information, see the **Motion Control Basic Functions** Function Manual, "Configuring technological alarms".

2.4.3.2 Local alarm reaction of the following axis and its mutual effects with the synchronous object

The synchronous object and the Axis TO have a reciprocal effect on each other depending on their respective operating modes and which commands are in effect. Thus, errors and alarms in the TO axis have a direct effect on the synchronous object functions. For example, if a technology alarm triggers a stop response in the following axis, the synchronous motion is also stopped. If an error is pending on the synchronous object only, the following axis can still position but can no longer perform synchronous operation.

The following responses by the Axis TO, which can be read in the **errorReaction** system variable, affect the synchronous object:

- **MOTION_STOP**
  Controlled motion stop with programmed ramp values.
- **MOTION_EMERGENCY_STOP**
  Controlled motion stop with maximum ramp values/limits for the axis.
- **MOTION_EMERGENCY_ABORT**
  Controlled motion stop with maximum axis dynamic values (starting from the current setpoint).

- **FEEDBACK_EMERGENCY_STOP**
  Motion stop with preassigned braking ramp.

  Note

  Problems with FEEDBACK_EMERGENCY_STOP on the leading axis

  Switchover to the actual value of the leading axis is performed. An existing setpoint-actual value different results in a step response of the speed or position on the synchronous object and therefore also on the synchronized axis.

  - Configure Motion_Emergency_Abort on the master axis

- **OPEN_POSITION_CONTROL**
  Motion stop with speed setpoint zero and abort.
- **RELEASE_DISABLE**
  Motion disable with controller disable and abort of all commands.

Synchronous operation is aborted for all responses indicated.

Note

Errors on the synchronous object do not have any effect on the enables/error response of the following axis.

Note

If the **disableSynchronousOperation** configuration data item is set to **YES** and simulation is active on the synchronous object, the synchronous operation function is *not* aborted on the stop response or when the **_disableAxis** command is executed.

The synchronous operation function is aborted with alarm "20005 Device type:..., log. address:... failed" if no valid actual value is present, e.g. if the axis is restarted.

2.4.3.3 Error handling in the user program

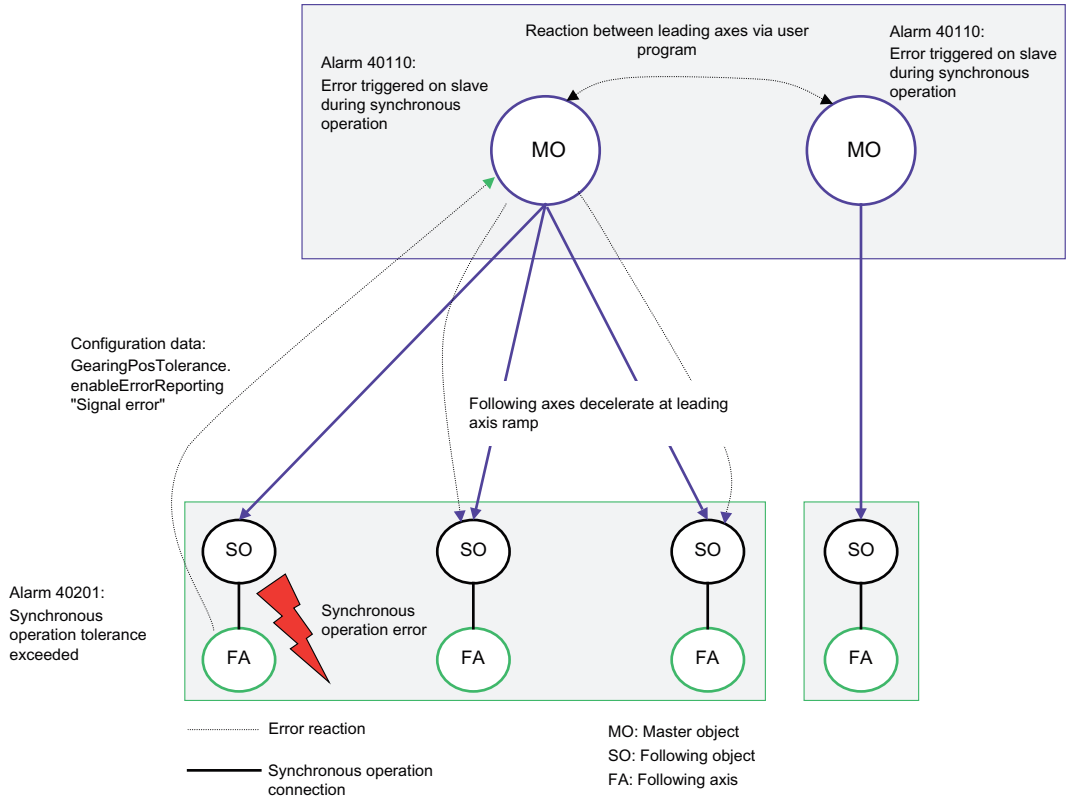


Figure 2-92 Error response in user program in distributed synchronous operation

The starting point is a synchronous operation error in a following axis (synchronous operation tolerance exceeded). Alarm 40201 "Synchronous operation tolerance of the gearing axis exceeded" is issued. The following axis changes to STOP mode.

The leading axis/external encoder responds with an error. The alarm 40110 "Error triggered on slave during synchronous operation (error number: ...) is signaled. The master object enters the STOP state.

- The local following axes also change to STOP mode.
- However, distributed synchronized axes continue to follow the master setpoint if measures are not taken in the user program to initiate an appropriate response to this error response.

Note

With V4.2 and higher, an error message is also output on the master object when the following axis disconnects the synchronous coupling for any reason in the event of an error.

For additional information, see Synchronous operation monitoring (Page 75).

2.4.4 Menus

2.4.4.1 Synchronous Operation - Menu

Grayed out functions cannot be selected.

Table 2-39 You can select the following functions:

Function	Meaning/Note
Close	Use this function to close the active window in the working area.
Properties	Properties displays the properties of the synchronous object selected in the project navigator. You can enter the object name plus author and version in this window.
Configuration	This function opens the configuration for the synchronous object selected in the project navigator. Assign the master values and cams to the following axis in this window.
Expert List	This function opens the expert list for the synchronous object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list; see Basic functions - Expert list.
Default setting	This function opens the default settings for the synchronous object selected in the project navigator. In this window, you define the replacement values for calling synchronous operation functions (**_enableGearing**, **_enableVelocity-Gearing**, and **_enableCamming** or **_disableGearing**, **_disableVelocityGear-ing**, and **_disableCamming**).
Settings	This function opens the settings for the synchronous object selected in the project navigator. You can define some synchronization settings in this window.
Interconnections	This function opens the interconnections for the synchronous object selected in the project navigator. You can see the inputs of the axis in this window.
Expert	This function opens the submenu for the expert settings.
Configure units	This function opens the Configure units of the object window in the working area. You can configure the units used for the selected object here.

See also

Synchronous Operation Configuration (Page 110)

2.4.4.2 Synchronous Operation - Context Menu

Grayed out functions cannot be selected.

Table 2-40 You can select the following functions:

Function	Meaning/Note
Open configuration	This function opens the configuration for the synchronous object selected in the project navigator. Assign the master values and cams to the following axis in this window.
Default setting	This function opens the default settings for the synchronous object selected in the project navigator. In this window, you define the replacement values for calling synchronous operation functions (**_enableGearing**, **_enableVelocityGearing**, and **_enableCamming** or **_disableGearing**, **_disableVelocityGearing**, and **_disableCamming**).
Settings	This function opens the settings for the synchronous object selected in the project navigator. You can define some synchronization settings in this window.
Interconnections	This function opens the interconnections for the synchronous object selected in the project navigator. You can see the inputs of the axis in this window.
Expert	This function opens the submenu for the expert settings.
Insert script folder	Insert script folder enables you to insert a folder for scripts.
Import object	Use **Import object** to open a window for the XML import. You can define the parameters for the XML import in this window.
Save project and export object	Use **Save project and export object** to open a window for the XML export. You can define the parameters for the XML export in this window.
Insert external master value	External master value (**ExternalMasterType**): Proxy object for an external master value.

See also

Synchronous Operation Configuration (Page 110)

Part II - Distributed Synchronous Operation

3

3.1 Overview of distributed synchronous operation

This chapter describes the **Distributed Synchronous Operation** function (V3.0 and higher). It introduces you to the operating principle and provides information about the technological supplementary conditions as well as the operating characteristics of distributed synchronous operation. You are shown how to create and configure a distributed synchronous operation.

Function overview

The **Distributed synchronous operation** functionality allows you to create a master value source and a synchronized axis on different controls. In a project, it is possible to form function groups and thus a machine structured on a modular basis. Synchronized axes no longer need to be present in a single control, but can instead be distributed among several controls.

Isochronous (clock-synchronized) bus coupling

The coupling between the master axis (or the external encoder) and the following axis is performed using an isochronous bus coupling between the controls, via PROFIBUS DP or PROFINET IO with IRT.

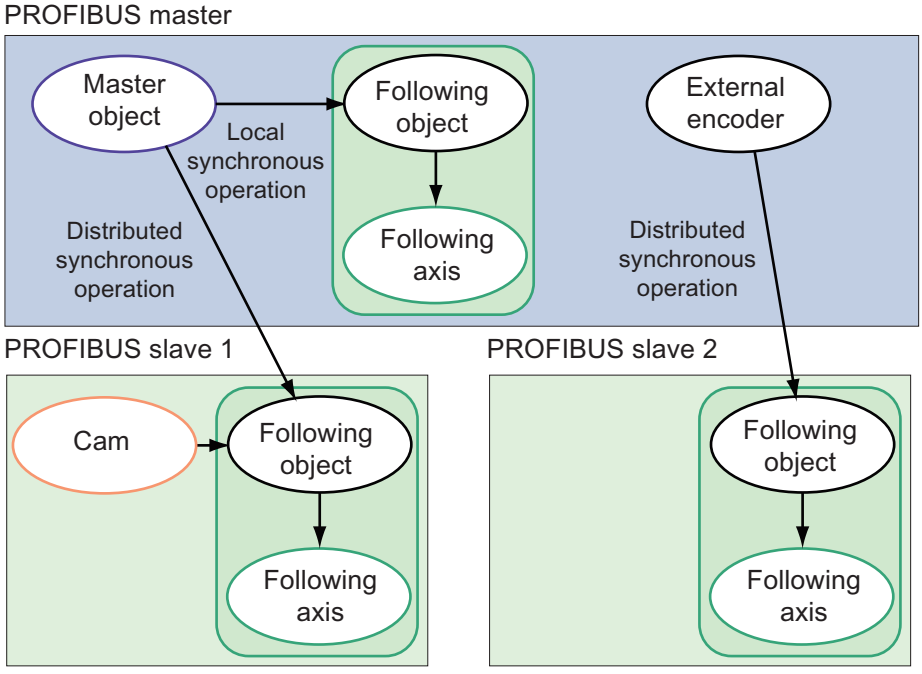


Figure 3-1 Distributed synchronous operation using PROFIBUS DP as an example

Synchronizing the bus interfaces

The DP/PN interfaces must be synchronized with each other for distributed applications using the isochronous PROFIBUS or PROFINET IO with IRT.

Information on this subject is available in the **Motion Control Basic Functions for Modular Machines** Function Manual and the **Communication** Configuration Manual.

Application

You can use distributed synchronous operation to create function groups in your project and set up a machine on a modular basis. Instead of having to use the same control system to control synchronously operating axes, you can now distribute the axes across a number of modules.

Operating principle/Compensation

With distributed synchronous operation, the interpolator cycle clocks of the master object and the following axis may be offset. As a result of the communication required, there is also an offset in the calculation of related signals (master value source and remote following axis).

The cycle clock offset can be compensated using the following measures:

- Compensation on the master value side by means of setpoint output delay
- Compensation on the slave value side by means of master value extrapolation

See Compensations for distributed synchronous operation (Page 157).

See also

Fundamentals of Distributed Synchronous Operation (Page 150)

Distributed Synchronous Operation Configuration (Page 168)

Configuring distributed synchronous operation across projects (Page 177)

3.2 Fundamentals of Distributed Synchronous Operation

3.2.1 Boundary Conditions

3.2.1.1 Rules for the communication/topology for distributed operation using PROFIBUS

The following rules apply for the PROFIBUS topology with distributed synchronous operation:

- Distributed synchronous operation is only possible via equidistant master/slave communication.
- The leading axis or external encoder must be located in the PROFIBUS master, and the distributed following axis must be located in the PROFIBUS slave.
  Further local synchronizations with the master control are possible.

- Distributed synchronous operation can only be created on one PROFIBUS level. Consequently, cascaded distributed synchronous operation is not possible.
- Different IPO cycle clocks and position control cycle clocks can be used in the SIMOTION devices involved.
- The same DP cycle clock must be used in the SIMOTION devices taking part in the distributed synchronous operation.

Exception: See **Cycle clock scaling for SIMOTION D** in this chapter

Data transmission for distributed synchronous operation using PROFIBUS

A total of 24 bytes are transmitted and received via the PROFIBUS interface for each synchronous operation connection and cycle clock (bi-directional connection for synchronous operation data). Only a certain amount of data can be transmitted in each DP cycle for each master-slave connection (a maximum of 244 bytes can be sent and received). This also enables a maximum of 10 connections with 24 bytes each. In addition, the amount of data in the PROFIBUS master is limited to 1 Kbyte for inputs and 1 Kbyte for outputs per PROFIBUS interface, irrespective of the number of devices connected; in other words, 40 connections are theoretically possible.

In addition to the transmitted data, the following must be noted:

- Drive connection
- I/O connection
- Application Data

This limits the number of possible connections with distributed synchronous operation.

The system can be optimized. Instead of several distributed connections, for example, a virtual axis on the slave can first be coupled to a (real or virtual) master axis which then serves several following axes.

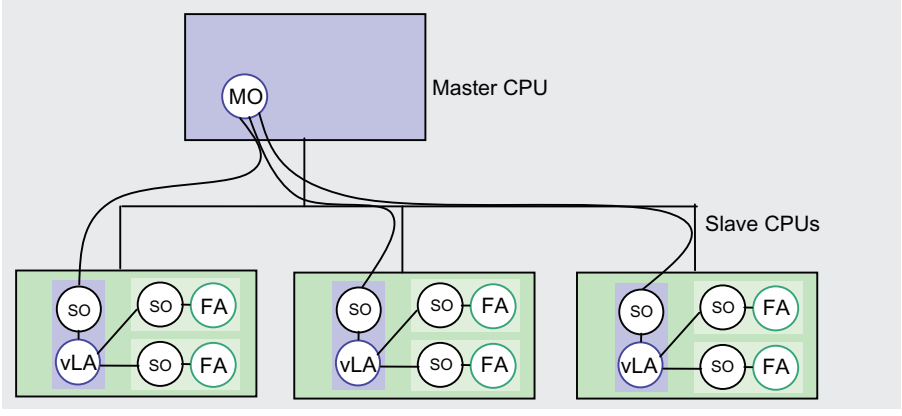


Figure 3-2 Example of optimizing connections using virtual master axes on slave devices

The virtual master axes on each slave allow axis groupings of the individual machine modules to also be operated "independently" (e.g. for the commissioning of individual modules).

Master-slave relationship

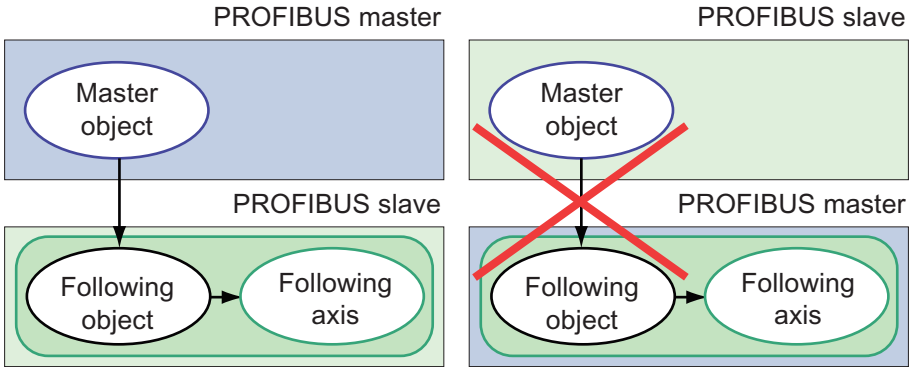


Figure 3-3 Master-slave relationship in distributed synchronous operation

Connection between axis and synchronous object

The synchronous object and, if appropriate, the cam must be located on the slave controller, together with the slave axis. The master value source (axis or external encoder) is always located on the master control.

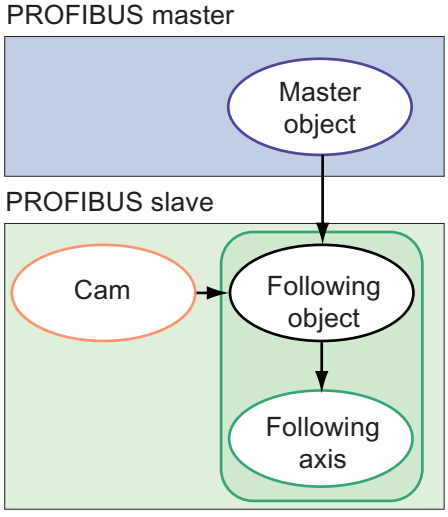


Figure 3-4 Following axis and synchronous object on the same control

Cascading

A distributed synchronous operation can be connected to following local synchronous operation on the slave controller.

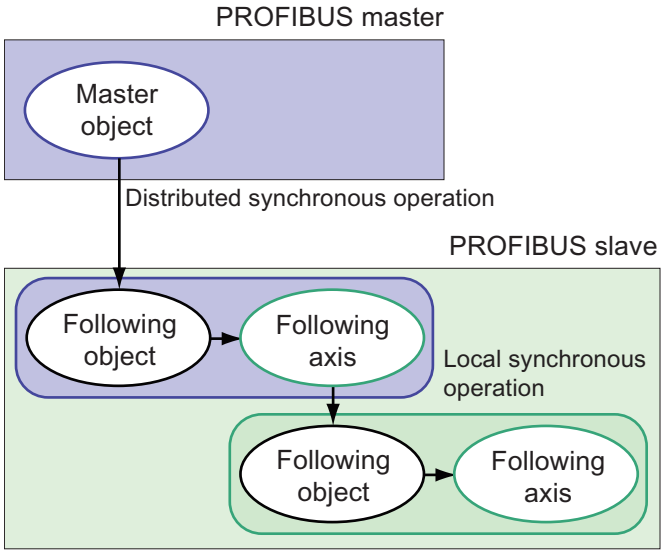


Figure 3-5 Cascading: distributed synchronous operation to following local synchronous operation

However, it is not possible to cascade two distributed synchronous operations one after the other, i.e. the following axis in distributed synchronous operation 1 cannot be used as the master axis in distributed synchronous operation 2. This is also true if the second PROFIBUS interface configured as a master is used.

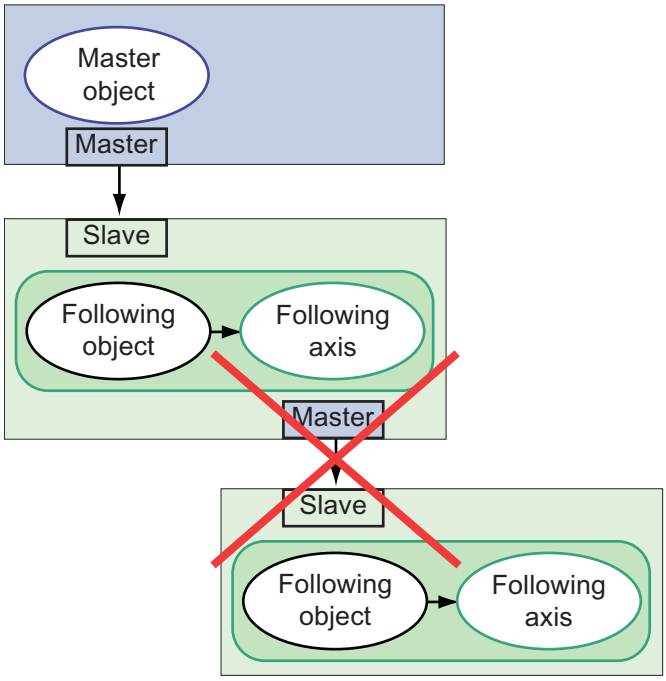


Figure 3-6 Distribution via one PROFIBUS level only

No feedback

It is not permitted to configure distributed synchronous operation from Device1 to Device2 and back again. This is true even if two appropriately configured PROFIBUS interfaces are used.

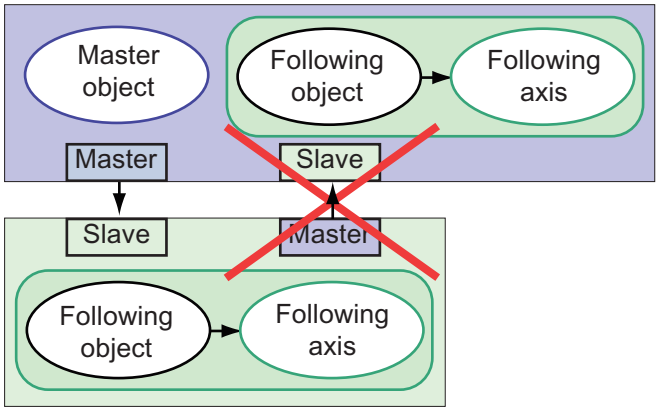


Figure 3-7 No feedback in distributed synchronous operation

Example hierarchy with synchronized isochronous PROFIBUS interfaces

The following is an overview of requirements for distributed synchronized operation:

All PROFIBUS connections:

- Must have the same DP cycle clock settings
  Exception: See **Cycle clock scaling for SIMOTION D** in this chapter
- Must have isochronous cycle clock settings
- Must have master and slave synchronization
  if a master and slave are used on the same device

- Distributed synchronous operation is only possible over one shared bus segment.
- Possible number of slaves: See **Data transmission for distributed synchronous operation using PROFIBUS** in this chapter

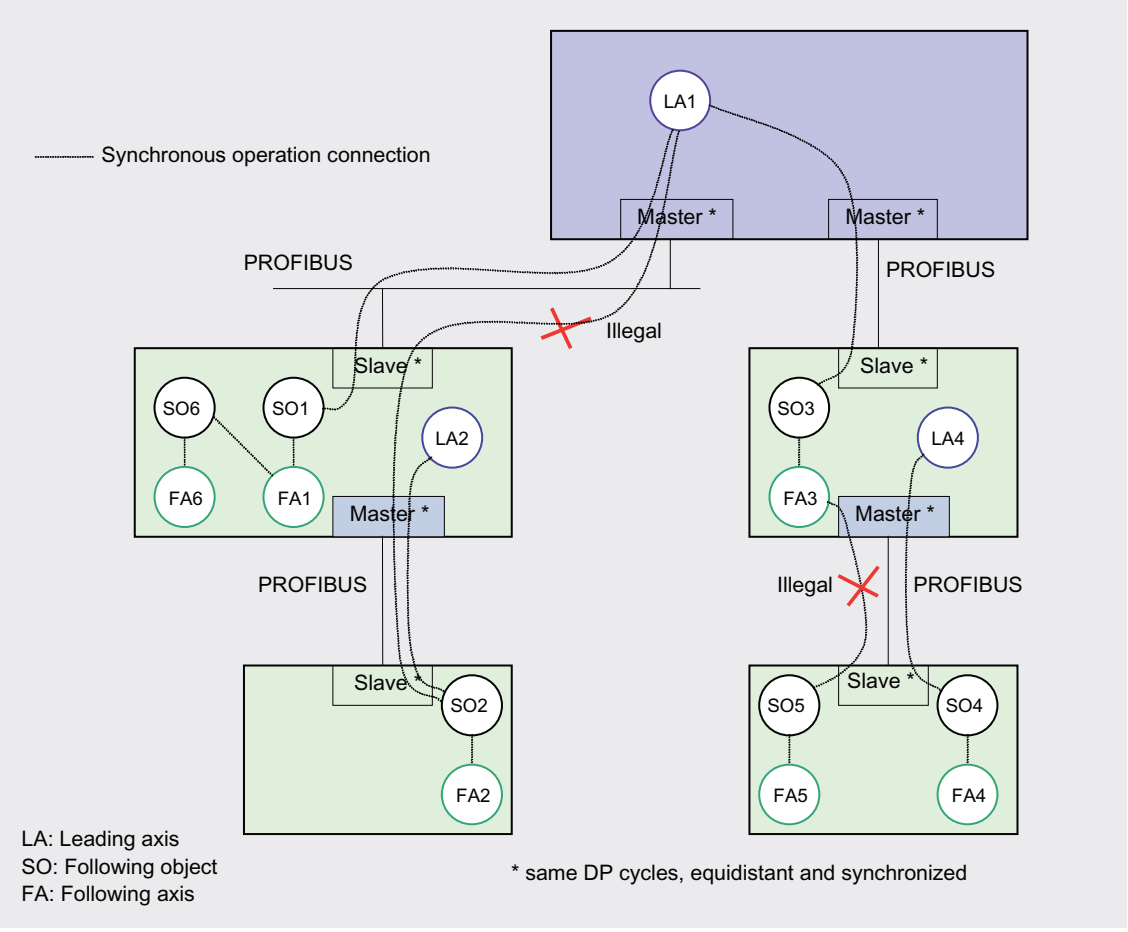


Figure 3-8 PROFIBUS topology: Hierarchy with synchronized isochronous PROFIBUS interfaces

Cycle clock scaling for SIMOTION D

With SIMOTION SCOUT V3.2 SP1 and higher, a distributed synchronous operation with cycle clock scaling between the two external DP interfaces (DP1/DP2) and the internal DP interface for SIMOTION D is possible.

If the master value changes only very slowly or the external DP interface requires a faster cycle time than the internal DP interface, a decoupling of the fast internal DP cycle from the slower external DP cycle is desirable.

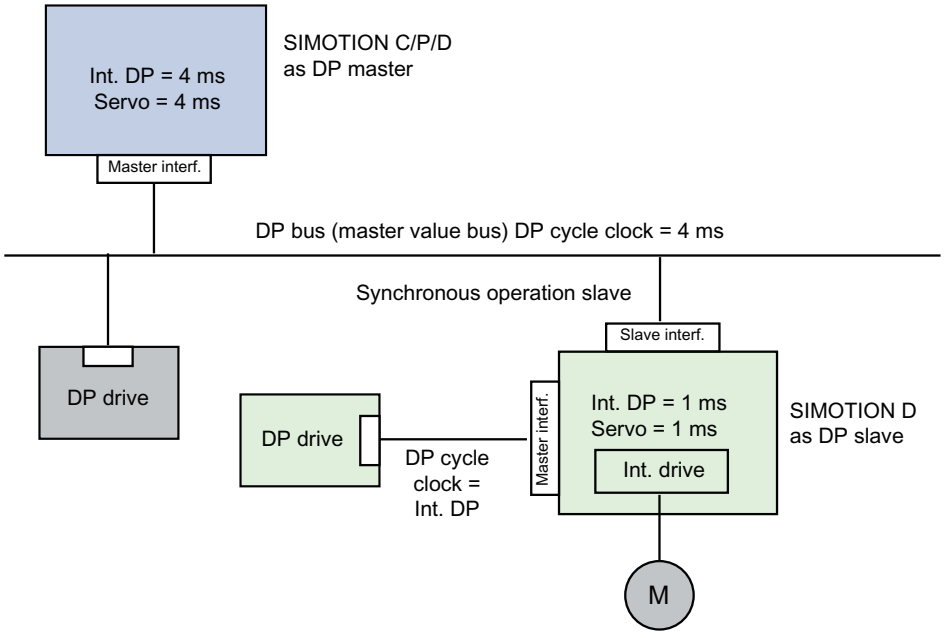


Figure 3-9 Cycle clock scaling for SIMOTION D

This is possible under the following boundary conditions:

- An external DP interface of SIMOTION D is used as an isochronous slave interface. Only in this case can an integer cycle clock scaling of isochronous external DP slave interface to internal interface be specified.
- For SERVO, IPO, and IPO_2, settings can be made for all permissible cycle clocks. The master axis and following axis can run on different IPO levels.
  The IPO cycle clock of the synchronous object, however, must be set equal to the cycle clock of the isochronous external DP slave interface (otherwise the error message "50205 - Offset cannot be determined" will be issued).
- In addition, the second external DP interface can be operated as isochronous master (the first is isochronous slave), for example, to operate external drives. In this case, the cycle clock must be the same as the cycle clock of the internal DP cycle.
- The second external DP interface can also be operated as a "non-isochronous, free-running interface". In this case, there is no effect on the cycle clock settings.
- The reduced cycle clocks of the external DP interfaces must be set in HW Config.
- For D4xx, DP to Servo reduction ratio is possible
- For D4xx-2, a DP to Servo reduction ratio is possible from V4.3 and higher
- A DP to Servo reduction ratio is generally only permissible if no PROFINET IO with IRT has been configured.

3.2.1.2 Rules for the communication/topology for the distribution using PROFINET IO with IRT (V4.0 or later)

Distributed synchronous operation between SIMOTION devices via PROFINET IO with IRT uses the controller-controller data exchange broadcast for PROFINET IO to exchange the synchronous operation data.

Differences to PROFIBUS

Regarding distributed synchronous operation with PROFIBUS, see **Rules for the communication/topology for distributed operation using PROFIBUS** in this chapter, the following differences exist:

- Master object and following axis / synchronous object can be located on any controllers. (PROFINET IO with IRT does not have any communications master and communications slave as for PROFIBUS.)
- Cascading of distributed synchronous operations is possible over more than one level.
- Recursive interconnection with PROFINET is possible (see Note)
  Note:
  slave value compensation via master value extrapolation is possible.
  Master value compensation by delaying setpoint output is not possible.

Note

For the following Control Units, a second Servo task is optionally available:
- SIMOTION D435-2 DP/PN (V4.3 and higher)
- SIMOTION D445-2 DP/PN (V4.2 and higher)
- SIMOTION D455-2 DP/PN (V4.2 and higher)
For further information, see *D4x5-2 Commissioning and Hardware Installation Manual and Basic Functions Function Manual*.

3.2.2 Compensations for distributed synchronous operation

In a distributed synchronous coupling, calculation of related signals between the master value source and the remote following axis is offset due to the distribution and the associated communication requirements. Compensation of this offset is supported by the system.

The following compensations are available in the system:

- Compensation on the master value side by means of setpoint output delay on the component that provides the master value for the distributed synchronous operation
- Compensation on the slave value side by means of master value extrapolation on the component containing the remote slave objects

Note

For distributed synchronous operation with extrapolation on the following axis, the **setpoint monitoring with jerk** setting is not appropriate.

The compensations are set and displayed via the system variable **distributedMotion**.

- The output delay is displayed on the leading axis.
- The master value delay is displayed on the synchronous object.
- The cycle clock offset is displayed on the synchronous object.

Sign-of-life monitoring is required for the compensation using master value extrapolation.

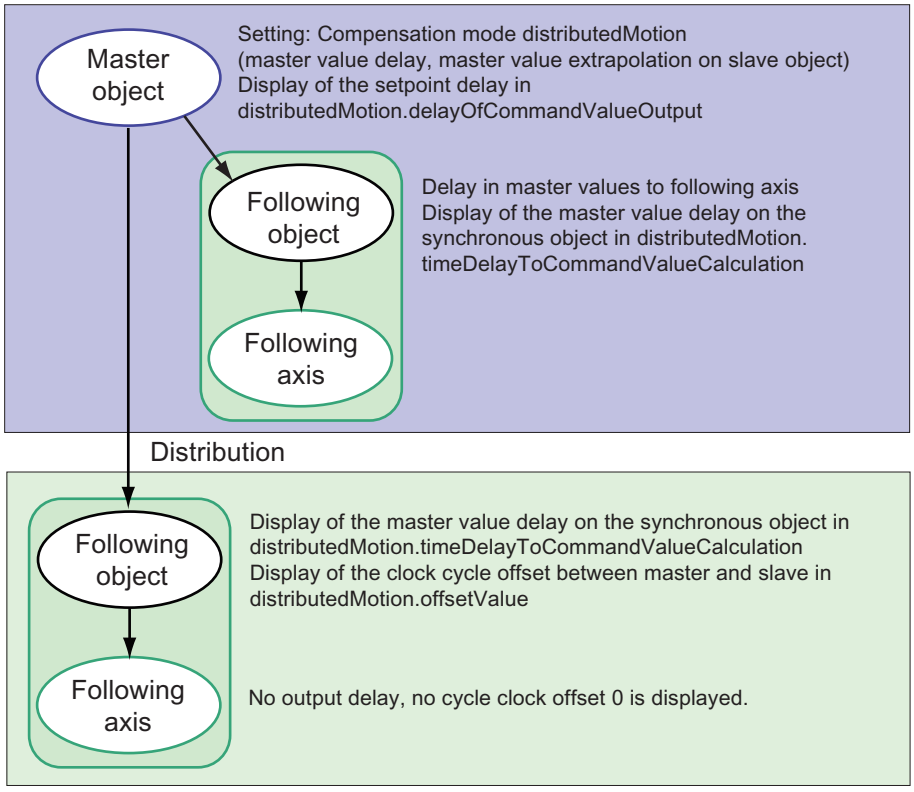


Figure 3-10 Overview: Compensations for distributed synchronous operation

Applications and results

- It is useful to activate a setpoint output delay on the master value side if, for example, synchronism of distributed synchronous operation is of primary importance and a rapid response on the master value side to local events is of lesser importance.
- It is useful to activate compensation on the slave value side by means of master value extrapolation without a setpoint output delay on the master value side if, on the master side, the master values and slave values need to be output without a delay due to a short response time, for example, and if, on the slave side, synchronism or a secondary error resulting from the larger extrapolation range is permissible.

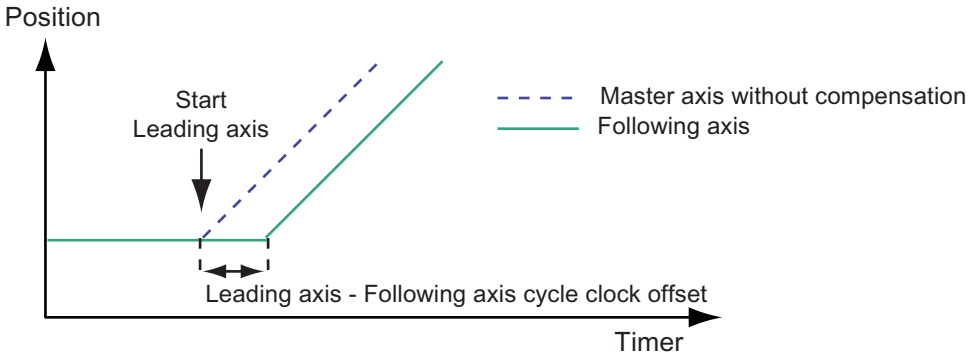


Figure 3-11 Distributed synchronous operation without setpoint output delay on the leading axis side and without compensation on the following axis side

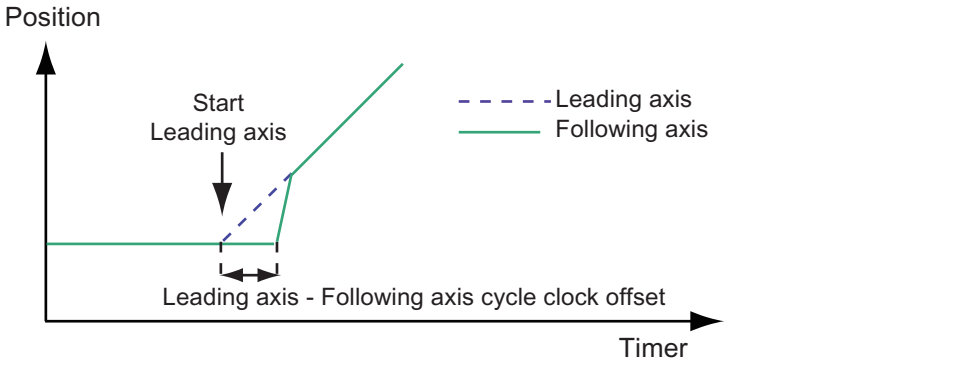


Figure 3-12 Master value extrapolation on the slave value side without setpoint output delay on the master value side

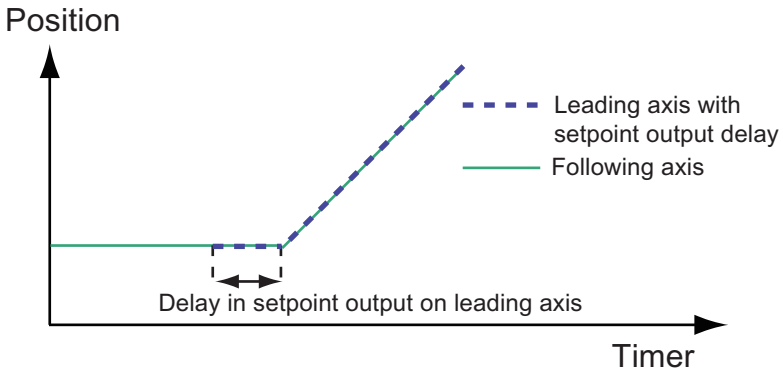


Figure 3-13 Setpoint output delay on the leading axis side

Activating

- When output delay on the master value side is activated, the signal output on the master value side is delayed by the calculated IPO clock cycles and any resulting IPO phase offset is compensated by means of interpolation on the slave value side.
- When master value extrapolation on the slave value side is activated without output delay on the master value side, the total cycle clock offset between the master value calculation and the slave value calculation is compensated for by means of the master value extrapolation on the slave value side.

Scope

- The setpoint output delay on the master value side is applicable to the following:
  - Axis setpoints calculated directly on the leading axis that are delayed on the axis prior to being passed on to the servo
  - Axis setpoints calculated by local synchronous objects; the synchronous object forwards the calculated setpoints to the axis
- Compensation on the slave value side by means of interpolation/extrapolation takes place on the master value of the remotely interconnected synchronous object.

See also

Compensation on master value side by means of setpoint output delay (Page 160)

Compensation of the slave value side by means of master value extrapolation (Page 162)

Permissible combinations for cycle clock offset compensation in distributed synchronous operation (Page 163)

Cycle clock offset calculation using a command (Page 164)

3.2.2.1 Compensation on master value side by means of setpoint output delay

Compensation on the master value side by means of setpoint output delay of the distributed synchronous operation will be activated for the master object.

Compensation on the master value side results in the following:

- The output of setpoints calculated for the axis, delayed by "n" IPO cycle clocks, to the servo/ position controller of the axis
- The output of setpoints from a local synchronous object interconnected with the master axis or external encoder, delayed by "n" IPO cycle clocks, to the interconnected synchronized axis

The number of IPO cycle clocks is calculated from the maximum cycle clock offset across all distributed synchronous relationships with the master value source. The number of IPO cycle clocks (integer), which contains the total delay, is calculated.

When compensation on the master value side by means of setpoint output delay is deactivated, setpoints are output to the axis and the master value is forwarded to and evaluated on local synchronous objects without delay.

Activating compensation on the master value side by means of setpoint output delay

Master value compensation by delaying setpoint output is activated using the following configuration data on the master axis and/or an external encoder:

- **(TypeOfAxis.)distributedMotion.enableOffsetCompensation:=YES**: Calculation of offset is activated
- **(TypeOfAxis.)distributedMotion.enableDelayOfCommandValueOutput:= YES**: Compensation on master value side is activated Delay to setpoint output on master value side, simultaneous start and no overshoots. The local following axis also has a delayed start. Disadvantage of "YES": There is less likely to be a response to an event on the master. When activated, a setpoint output delay on the axis is always effective.
- **enableDelayOfCommandValueOutput:= NO** is used to deactivate compensation on the master side. The master immediately outputs the setpoint value.

When activated, a setpoint output delay on the axis is always effective.

Cycle clock offset calculation

The system automatically calculates the maximum cycle clock offset after a transition from **STOP**/ **STOPU** to **RUN**. Cycle clock offset calculation also runs after one of the axes or an external encoder involved has been restarted and after disconnection/reconnection. The status of the cycle clock offset calculation is indicated in the **distributedMotion.stateOfOffsetCalculation** system variable on the master axis and on the remote distributed synchronous object. The cycle clock offset is not yet determined for status **INVALID**, the cycle clock offset cannot be determined. The master axis/external encoder issues the technological alarm "40304 Offset cannot be determined".

Compensation on the master value or slave value side requires
that offset calculation be activated in configuration data element
(TypeOfAxis.)distributedMotion.enableOffsetCompensation, on the master axis or external encoder.

setpoint output delay

The absolute value of the setpoint output delay can be read out by means of
the **distributedMotion.delayOfCommandValueOutput** system variable on the master axis.

The time is indicated in the **distributedMotion.timeDelayToCommandValueCalculation** system variable on the synchronous object of the remote following axis.

The status of the setpoint output delay is indicated in the **distributedMotion.stateOfDelayValue** system variable on the master axis or external encoder and on the remote synchronous object.

- If the status is **INVALID**, the setpoint output delay is not activated.
- If the status is **VALID**, the setpoint output delay is active.

The maximum permissible delay for setpoint output is 10 interpolation cycle clocks. If the delay exceeds this, two alarms are output on the master value axis or external encoder: "40124 Offset cannot be compensated" and "40125 Setpoint output delay on the master side is deactivated."

If a local interconnected synchronized axis is acting as the master value for a distributed synchronous operation, the same setting in terms of the effective compensations on the master value side should be made on the first master value and on the local synchronized axis.

3.2.2.2 Compensation of the slave value side by means of master value extrapolation

When compensation on the master value side by means of setpoint output delay is not activated, the compensation on the slave value side performs a linear extrapolation using the two most recent master values received in order to compensate for the cycle clock offset.

In the event that master values are lost, the two most recent master values received are used for the extrapolation.

Activating compensation on the slave value side using master setpoint extrapolation

Slave value compensation (interpolation/extrapolation) is activated using the following configuration data on the master axis and/or an external encoder:

- **(TypeOfAxis.)distributedMotion.enableOffsetCompensation:=YES**: Calculation of offset is activated.
- **(TypeOfAxis.)distributedMotion.enableDelayOfCommandValueOutput:= NO**: Compensation on master value side is deactivated.

Display of setpoint output time delay on the master value side

The setpoint output delay time on the master value side is indicated in the **distributedMotion.timeDelayToCommandValueCalculation** system variable on the synchronous object. This delay time is generally greater than the offset calculated across all remote synchronous relationships.

The **distributedMotion.timeDelayToCommandValueCalculation** system variable on the remote synchronous object corresponds to the **distributedMotion.delayOfCommandValueOutput** system variable of the associated master value object.

Status of calculation of compensation on the slave value side

The status of the compensation calculation on the slave value side is indicated by means of the **distributedMotion.stateOfOffsetCalculation** system variable on the synchronous object as well as on the master value source/master axis.

Cycle clock offset calculation

The system automatically calculates the maximum cycle clock offset after a transition from **STOP**/**STOPU** to **RUN**. Cycle clock offset calculation also runs after one of the axes or an external encoder involved has been restarted and after disconnection/reconnection. The status of the cycle clock offset calculation is indicated by means of the **distributedMotion.stateOfDelayValue** system variable on the synchronous object as well as on the master value source.

If a command is transmitted to the synchronous object before completion of the cycle clock offset calculation, a technological alarm is output ("50204 Offset calculation is active"). If the cycle clock offset cannot be calculated, the synchronous object outputs a technology alarm ("50205 Offset cannot be calculated").

Cycle clock offset between master value calculation and slave value calculation

The clock cycle offset between the master value calculation and slave value calculation is indicated on the synchronous object by means of the **distributedMotion.offsetValue** system variable. The cycle clock offset displayed does not depend on the output delay on the master value side.

Comment:

- All system variables on the master value source indicate the status or the respective value across all interconnected following axes.
- All system variables on the synchronous object indicate the status or the respective value for the interconnection with the current master value source.

3.2.2.3 Permissible combinations for cycle clock offset compensation in distributed synchronous operation

The compensation settings are made on the master value side (on the master control) using configuration data elements **(TypeOfAxis.)-distributedMotion.enableDelayOfCommandValueOutput** and **(TypeOfAxis.)distributedMotion.enableOffsetCompensation**.

The following combinations are possible:

Table 3-1 Permitted settings of compensation on the master value side and slave value side (setting on the leading axis or external encoder)

enableOffset Compensation	enableDelay OfCommand ValueOutput	
NO	**NO**	No compensation activated
NO	**YES**	Not permitted. An output delay on the master value side without interpolation on the remote following axis is not permitted.
YES	**NO**	Linear extrapolation in the following axis on the last two master setpoints using the cycle clock offset
YES	**YES**	Leading setpoint output delay in the local following axis, linear interpolation in the following axis of the master setpoints transmitted using the cycle clock and phase offset

3.2.2.4 Cycle clock offset calculation using a command

The cycle clock offset calculation can be initiated explicitly (V4.1 and higher), for example, when adding an axis for modular machine concepts.

The cycle clock offset can be calculated on the leading axis with the **_enableDistributedMotionDelayValueCalculation** command. This command acts on the master value and all cascades below it for which this master value applies in its cascade. Active synchronous operation commands are aborted when the offset calculation is started. If there are multiple master values at the top level, the command must be called on each object.

See also

Rules for the communication/topology for the distribution using PROFINET IO with IRT (V4.0 or later) (Page 156)

3.2.3 Operating axes with distributed synchronous operation

3.2.3.1 Sign-of-life monitoring

The master axis/external encoder and the remote synchronous object exchange signs of life to provide confirmation that the application is running correctly on the other side. For example, a distributed synchronous operation connection can be adversely affected by a fault on the bus (such as, message frame repetition).

Life-sign monitoring is implemented in the form of one life-sign counter in the master axis/external encoder and the synchronous object for each distributed synchronous operation. The process could be described as a "clock comparison". The sign of life is sent from the synchronous object to the master axis/external encoder and vice versa (bidirectional sign-of-life monitoring).

The life-sign counters are incremented on the source side in each IPO cycle clock, and the current value is transmitted to the relevant partner, where it is compared to an expected value. If the life-sign counter values differ from the expected values, an error will be output. Life-sign monitoring is only active if both SIMOTION devices are in **RUN** mode.

Note

A life-sign failure is not the same as an interrupted connection. The life-sign failure occurs if the life-sign is not received by the communication partner. This is the case with IPO overflow, for example. The connection is interrupted when the two communication partners are physically separated from one another. Further information can be found under "Connection interruption" in this chapter.

Failure limit

A parameter can be assigned to permit the life-sign to fail "n" times before an error response is output. This is set using the **(TypeOfAxis.)DistributedMotion.numberOfLifeSignFailures** configuration data (default: **1**).

The failure limit can be set on the leading axis or external encoder and on the following axes.

Extrapolation in the event of a failure

If a sign of life (and therefore also the master value) is absent and no error response is triggered (n > 0), the last available master setpoint is extrapolated.

Error reaction

An error response is triggered if the number of failures exceeds the parameterized value "n". The life-sign error is indicated on the following axis and the leading axis/external encoder. Consequently, the error reaction is applied to both of the following:

- **Leading axis/external encoder**: **40301** Loss of slave connection to the distributed control in the distributed synchronous operation
- **Following axis**: **50201** Loss of connection on the assigned control to the master in the distributed synchronous operation

Activating/deactivating life-sign monitoring

Life-sign monitoring can be enabled/disabled using the **(TypeOfAxis.)distributedMotion.enableLifeSignMonitoring** configuration data element on the master axis/external encoder and the synchronous object (default: **YES**).

Status signal

Life-sign monitoring must be activated on all participating objects or deactivated on all participating objects. If this is not the case, a technological alarm is issued on the master axis/external encoder or synchronous object to indicate that life-sign monitoring has been deactivated:

- **Leading axis/external encoder**: **40302** Life-sign monitoring for slave deactivated in distributed synchronous operation
- **Following axis**: **50202** Life-sign monitoring for master deactivated in distributed synchronous operation

The **distributedMotion.lifeSignError** system variable on both the leading axis/external encoder and synchronous object can be used to check whether life-signs have failed.
When life-sign monitoring is activated, the assigned tolerated life-sign monitoring failures in **(TypeOfAxis.)DistributedMotion.numberOfLifeSignFailures** must be identical.

Connection interruption

Without a user-programmed response, following axes that obtain a master value from a controller that is no longer reached due to the disconnection will fail. If you do not want the following axes to fail, perform the following steps:

- Response to alarm "50201" (connection failure to current master (assignment: /l/%d)) on the controller with the following axes in the technology fault task.
- Acknowledge the alarms occuring.
- Send a motion command in the IPO task for the following axis.
- The physical connection must not be restored or the failed controller(s) switched on again until the controller(s) with the following axes are in the STOP operating state.

Note

If PROFISafe is also in use, the above description only works if the controller(s) with the following axes still has/have a connection to the F-CPU.

After the physical connection between the motion controls has been restored, the 1940 / 1951 drive faults can occur with SINAMICS drives. These can only be acknowledged by Power Off/On.

3.2.3.2 Operating states

Please note that in order for distributed synchronous operation to function, the master CPU *and* the slave CPU must both be in **RUN** mode. The connection for distributed synchronous operation does not occur automatically unless both are in **RUN** mode.

Association with life-sign monitoring

If life-sign monitoring is *deactivated* (**(TypeOfAxis.)distributedMotion.enableLifeSignMonitoring** = **No**) and a transition occurs from **RUN** to **STOP**/**STOPU**, the master CPU outputs a technological alarm (life-sign error **50201**) to all connected slave CPUs.

Synchronization status

The following device-related system variables on the slave indicate the status of synchronization:

- **StateOfDpinterfaceSynchonization**: only relevant if DP of PROFIBUS is equidistant
- **StateOfDpSlaveSynchonization**: only relevant if DP of PROFIBUS is not equidistant

Direction reversal of the master value for the synchronization

A maximum master value reversal can be defined in the configuration data **syncingMotion.masterReversionTolerance** on the synchronous object (V4.0 and higher). Specifying a tolerance is particularly useful for a distributed synchronous operation, where master value noise for actual value coupling can cause a master value reversal due to extrapolation.

See Actual value coupling with tolerance window (Page 39).

Error handling

During distributed synchronous operation, the following axis sends the "Synchronous operation tolerance exceeded" error to the master axis. With V4.2 and higher, an error message is also output on the master object when the following axis disconnects the synchronous coupling for any reason in the event of an error. The error transfer must be configured: By means of the synchronous operation monitoring wizard on the following axis of the slave or by means of the **TypeofAxis.GearingPosTolerance.enableErrorReporting** configuration data.

Other errors (e.g. if the following axis is unable to synchronize) are not communicated to the master object. It is recommended that the 4xxxx errors be sent by the application to the master via the bus.

In HW Config of the I-slave, for example, a byte can be added to the configuration created by the system. This means that the following axis can report every error to the master axis by means of an application.

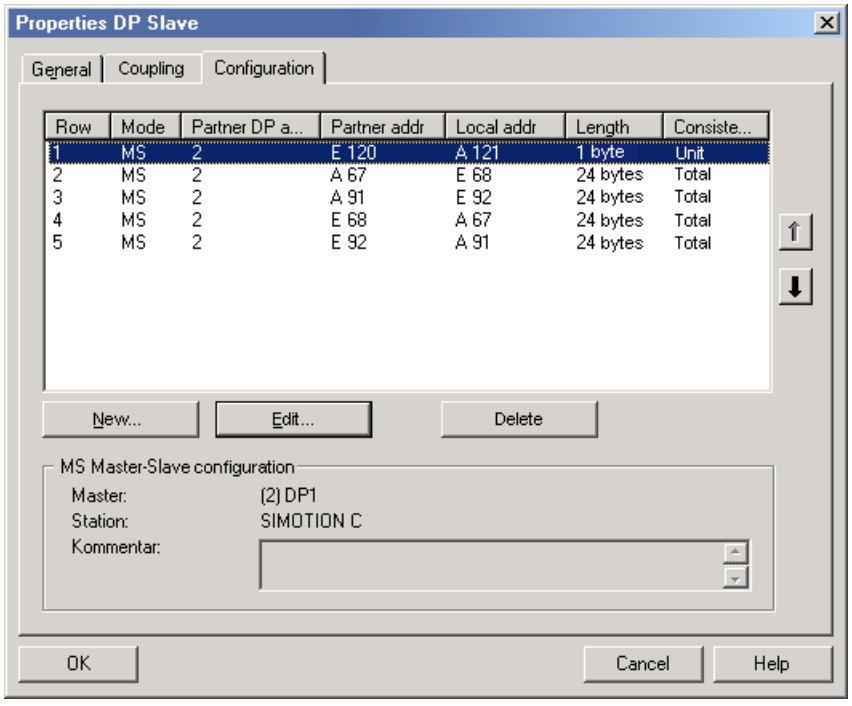


Figure 3-14 Configuration of an error byte for signaling the leading axis

See Error handling (Page 143).

3.3 Distributed Synchronous Operation Configuration

This section describes how to create and configure SIMOTION devices and objects with distributed synchronous operation, and how to download them to the target system.

It is assumed that you have already created cams, master axes, or external encoders. A sample project for distributed synchronous operation can be found in the FAQ on the **SIMOTION Utilities & Applications DVD**.

To configure a distributed synchronous operation, you must carry out the following tasks:

- Creating SIMOTION devices with SCOUT (Page 169)
- Creating connections with HW Config (Page 169)
- Creating synchronous operation connections with SCOUT (Page 171)
- Synchronizing the interfaces (Page 175)
- Generating a synchronous operation configuration (Page 174)
- Identify Possible error (Page 174)

3.3.1 Creating SIMOTION devices with SCOUT

Creating a slave and master

In SIMOTION SCOUT, create two SIMOTION devices in your project:

- A SIMOTION device configured as a slave.
- A SIMOTION device configured as a master.
- Both SIMOTION devices must be connected and configured so that the HW Config can be compiled without errors.

Note

The SIMOTION devices must always be created with the same kernel version for the master and all slaves.

For exceptions, see information at Overview of distributed synchronous operation with different RT versions (Page 213).

Note

Addresses entered by SIMOTION SCOUT for axis communication of distributed synchronous operations in HW Config may not be changed.

3.3.2 Creating connections with HW Config

Requirement

You have created a SIMOTION project containing two SIMOTION devices.

Creating connections using PROFIBUS

1. Set the "Operating mode" property for SIMOTION devices in HW Config. Open the "Properties" window at the relevant interface, e.g. DP/MPI. On the "Operating mode" tab, set one SIMOTION device to **DP Slave** and the second SIMOTION device to **DP Master**.
2. Set PROFIBUS addresses for both SIMOTION devices.
3. Set the DP mode to **DPV1** for both SIMOTION devices.
4. Insert a subnet master system on the PROFIBUS master.
5. Under **Properties**, set the transmission rate.
   For PROFIBUS communication, a transmission rate of **12 Mbit/s** is recommended.
6. Under **Options** in **Properties**, select **Equidistant bus cycle** and set the cycle time.
7. Check that the check box for **Programming, status/control or other PG functions and non-configured communication connection possible** under **Operating mode** in **PROFIBUS properties** for the slave has *not* been selected. Clear it if necessary.
   If the check box is selected, isochronous operation is *not* possible.
8. From the hardware catalog of **PROFIBUS DP**, **already configured stations**, add the **C23x**/**P3xx**/**D4xx ISlave** to the existing master system.

9. The DP slave which has already been configured is offered under Coupling. Use **Connect** to connect the slave to the master.

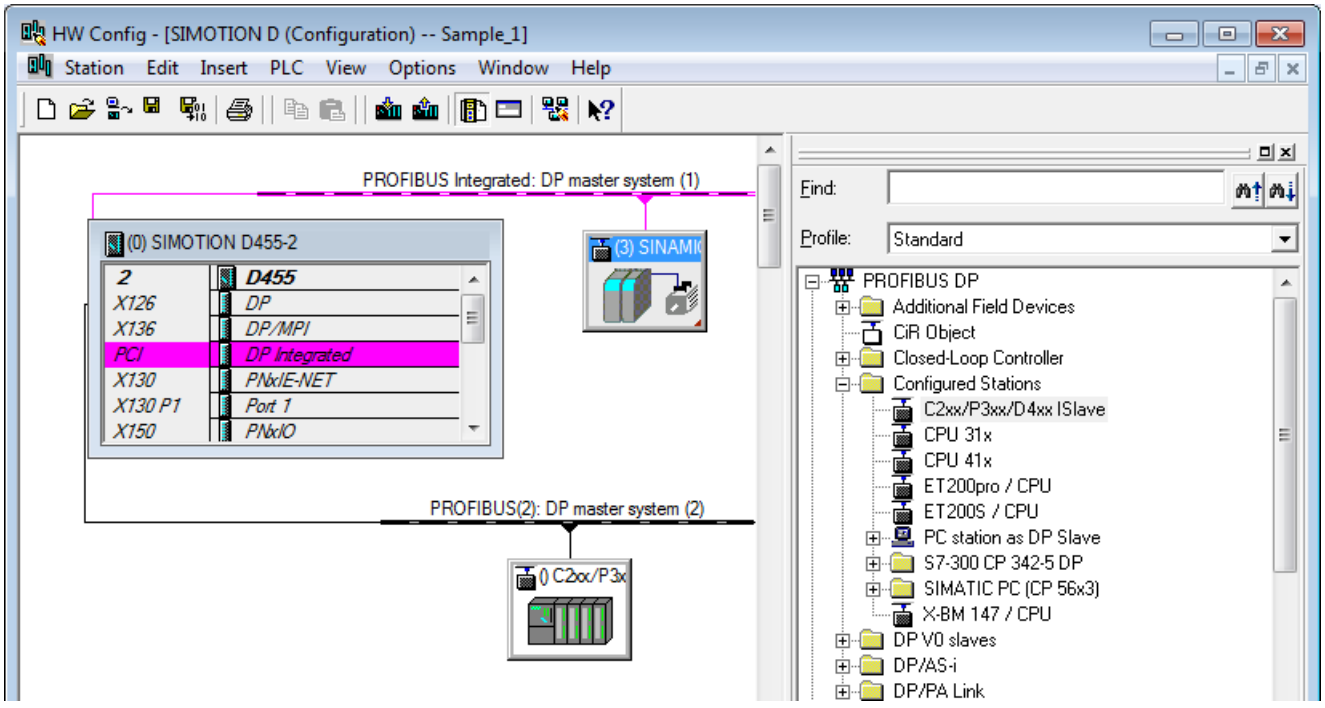


Figure 3-15 Device as PROFIBUS master

10. Establish a connection between the two devices (e.g. one byte for synchronization tasks between user programs on the devices); alternatively, you can ignore compilation errors in HW Config resulting from the absence of configured connections.
    The connection is configured only once the axes on various devices in SIMOTION SCOUT have been interconnected.

Detailed information on cross-project distributed synchronous operation illustrated with an example configuration can be found in Section PROFIBUS communication configuration (Page 179). You will find detailed information on communication configuration in SCOUT TIA in Section Communication between SIMOTION two controllers in the SIMOTION SCOUT TIA Configuration Manual.

Creating connections using PROFINET IO with IRT

The creation of a distributed synchronous operation connection requires that at least two IO controllers are connected with each other using IRT; a complete IRT configuration does not need to have been performed:

1. Set the IP addresses. Recommendation: fixed IP addresses
2. Create the two SIMOTION devices.
3. Configure the IRT operation by setting synchronization type **SyncSlave** for one of the devices and **SyncMaster** for the other device in HW Config.

This creates HW Config for distributed synchronous operation. The configuring of the axes and the interconnection of the axes is the same as for PROFIBUS.

Detailed information on cross-project distributed synchronous operation illustrated with an example configuration can be found in Section Communication configuration via PROFINET IO (Page 190).

Saving and compiling in SIMOTION SCOUT automatically creates the data to be exchanged between the SIMOTION devices for the distributed synchronous operation.

3.3.3 Creating synchronous operation connections with SCOUT

Master axis/external encoder

1. Create the master axis or the external encoder (in the PROFIBUS master for the distribution using PROFIBUS).
2. Configure the master object (master axis/external encoder) using the wizard.

Following axis

1. Create a following axis (in the PROFIBUS slave if distribution is taking place via PROFIBUS). Make sure that **Synchronous operation** is activated as the technology when creating the following axis.

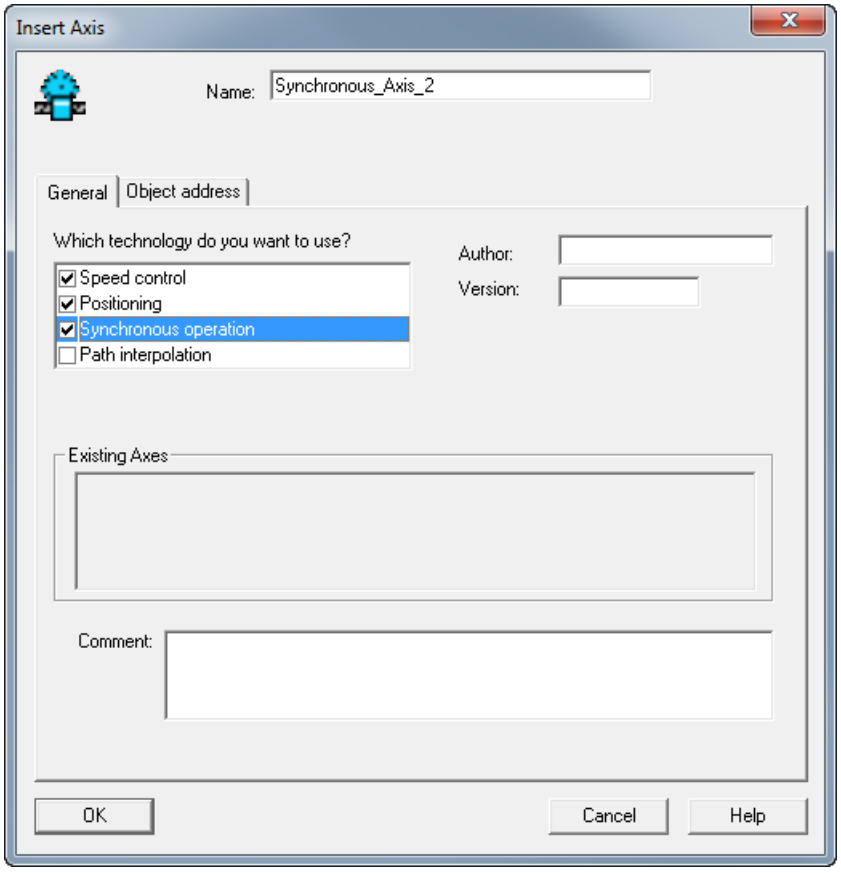


Figure 3-16 Inserting an axis with synchronous operation

2. Configure the following axis using the wizard.
3. Connect the synchronous object for the following axis to the master object (master axis/ external encoder) by selecting the master object under Interconnections.

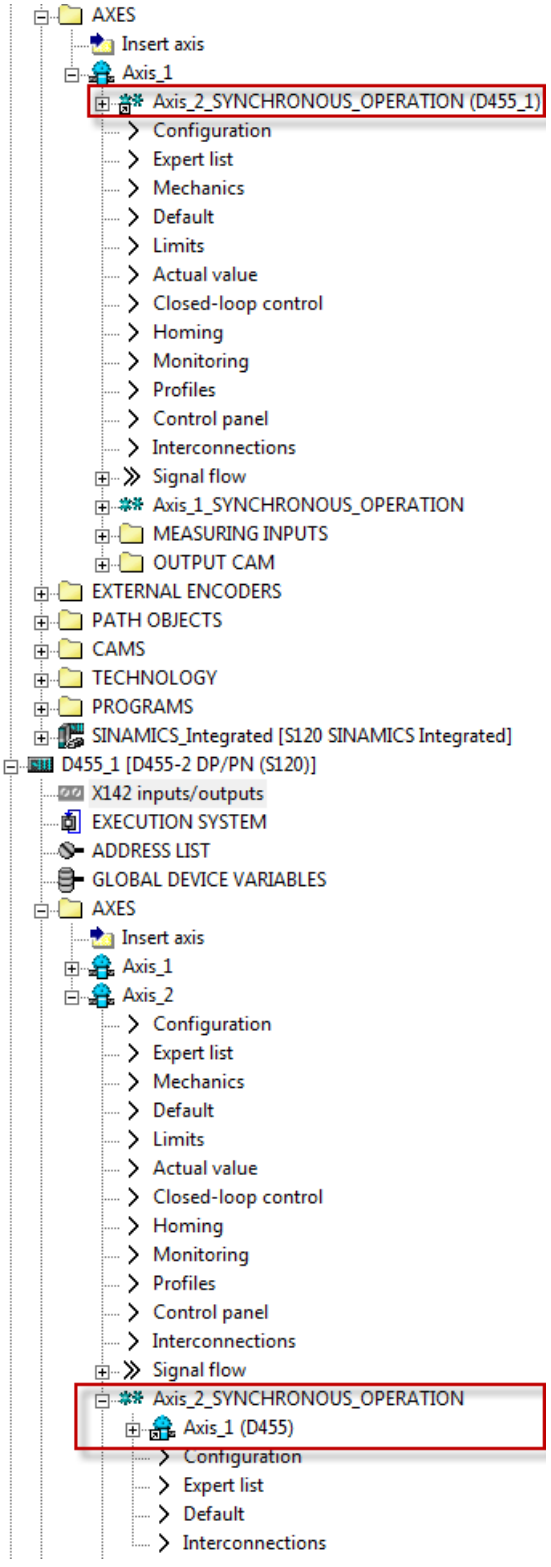


Figure 3-17 Distributed synchronous operation: Connecting a leading axis to a following axis

3.3.4 Generating a synchronous operation configuration

Compiling the project

Save the project via the menu **Project -> Save and compile changes**.

The system automatically generates the PROFIBUS IO configuration data in HW Config for the 24 bytes of send and receive info for the distributed synchronous operation. The configuration is generated and checked for consistency. Both SIMOTION devices must be connected and configured so that the HW Config can be compiled with no errors.

Note

Only at this point can it be determined whether the network resources are sufficient. If they are not, an error message will be generated to this effect. The compilation process must be free of errors; otherwise, it will not be possible to download the project to the target system.

Downloading the project to the target system

Load the project to both SIMOTION devices, so that these are displayed consistently. The configuration for the synchronous operation is also transferred to the SIMOTION devices during loading.

Note

There is no mechanism integrated into the SIMOTION Kernel for checking the consistency of downloaded projects among multiple devices, e.g. during ramp-up. Consistent downloading can only be achieved by means of "Download" in SIMOTION SCOUT.

3.3.5 Possible error

I/O resources unavailable (PROFIBUS)

If the system detects that the required resources (number of bytes) are no longer available in the PROFIBUS I/O data, SCOUT will generate an error message during the compilation process.

Check resources

- Call the PROFIBUS **DP Properties** for the relevant slave in HW Config.

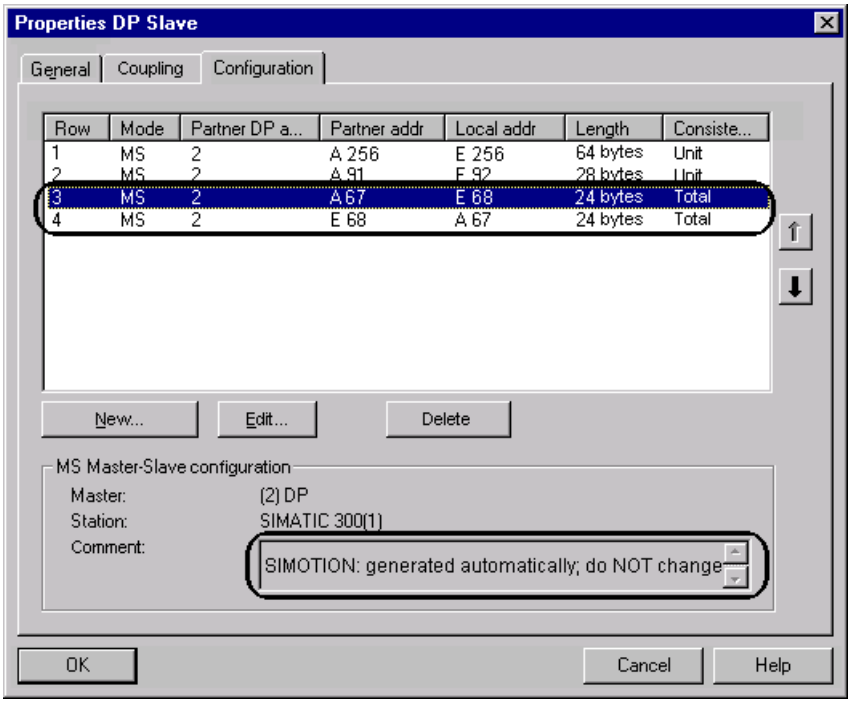


Figure 3-18 View of PROFIBUS resources with automatically generated entries

Note

Do *not* change or delete the entries in the I/O PROFIBUS configuration generated automatically by SIMOTION SCOUT. If you do, the distributed synchronous operation connection can no longer be used.

Faulty/incomplete configuration

If the system detects a fault in the configuration, SIMOTION SCOUT will output an error message during the compilation process.

3.4 Programming distributed synchronous operation

3.4.1 Synchronizing the interfaces

The interfaces are synchronized in the StartupTask for the slave by means of the **_enableDpInterfaceSynchronizuationMode** function call, either automatically by transferring the **dpInterfaceSyncMode** function parameter with the **AUTOMATIC_INTERFACE_SYNCHRONIZATION** option, or manually by transferring the **MASTER_SLAVE_ALARMMESSAGES_1** option.

If the **_enableDpInterfaceSynchronizationMode** function is called with the **MASTER_SLAVE_ALARMMESSAGES_1** option, the **_synchronizeDpInterface** function must be called at a later point in time to synchronize the interfaces.

If the "automatic" function is used and the connection is interrupted, the interface synchronizes itself automatically once the connection is restored.

If the "automatic" function is not used, the interface must be synchronized manually every time the connection is interrupted.

The **stateOfDpInterfaceSynchronization** = **DP_INTERFACES_SYNCHRONIZED** system variable on the slave indicates whether the two interfaces are synchronized.
Exception for SIMOTION C240:
If drives are connected to a SIMOTION C240 via the analog interface (onboard, e.g. for hydraulic applications), the synchronization is displayed via the **stateOfDpSlaveSynchronization = DP_INTERFACES_SYNCHRONIZED** system variable.
The interfaces must be synchronized to ensure error-free distributed synchronous operation.

Further information is available in the **Motion Control Basic Functions for Modular Machines** Function Manual.

3.4.2 Synchronous commands

If your commands include the object name of the master value source that is located on the other SIMOTION device, you must specify the name of this device as a prefix (e.g. D445.axis_1)

If you require data from technology objects that are located on the other device, you can use the application as a vehicle for this, by expanding synchronous operation communication between the devices.

See also the section titled *Error handling*.

Checking to determine whether the following axis is ready

When the distributed synchronous operation is to be started on the following axis (e.g. through **_enableGearing**), he application must ensure that the synchronized axis is ready.

Compensation on the following value side:

The following conditions must be met on the following axis if the following are activated on the master axis **TypeOfAxis.DistributedMotion.enableOffsetCompensation = YES** and

TypeOfAxis.DistributedMotion.enableDelayOfCommandValueOutput = NO:

- The **stateOfDpInterfaceSynchronization** system variable for the following axis must indicate **DP_INTERFACES_SYNCHRONIZED**.
- The **distributedmotion.stateOfOffsetCalculation** system variable for the synchronous object must indicate **valid**.

Compensation on the master value side:

The following conditions must be satisfied on the following axis if the following are activated on the master axis **TypeOfAxis.DistributedMotion.enableOffsetCompensation = YES** and **TypeOfAxis.DistributedMotion.enableDelayOfCommandValueOutput = YES**:

- The **stateOfDpInterfaceSynchronization** system variable for the following axis must indicate **DP_INTERFACES_SYNCHRONIZED**.
- The **distributedmotion.stateOfOffsetCalculation** system variable for the synchronous object must indicate **valid**.
- The **distributedMotion.stateOfDelayValue** system variable for the synchronous object must indicate **valid**.

No compensation:

When **TypeOfAxis.DistributedMotion.enableOffsetCompensation = NO** and **TypeOfAxis.DistributedMotion.enableDelayOfCommandValueOutput = NO** is activated on the master axis, the function is the same as for compensation on the following value side.

If this is not the case (for example, if **_enableGearing()** is executed on the following axis even though the master axis is not ready), the command is aborted with error "50102 Master is not assigned/configured or is faulty (reason: ..." aborted).

3.5 Configuring distributed synchronous operation across projects

3.5.1 Overview

As of V4.1, it is possible to configure a distributed synchronous operation across projects; this is carried out using proxy objects. The SIMOTION devices for the master object and the following axis are located in separate SIMOTION projects.

In each case, there is a proxy "External Synchronous Operation" technology object, which is assigned to the master axis, and a proxy "External Master Value" technology object, which is assigned to the slave axis. The external interface of the proxy technology objects is formed by means of coupling via I/O addresses. The network configuration must be set in HW Config. The send and receive data must be interconnected with one another. From the point of view of the runtime system, *distributed synchronous operation* and *cross-project distributed synchronous operation using proxy technology objects* are identical.

With regard to version requirements, see Overview of distributed synchronous operation with different RT versions (Page 213)

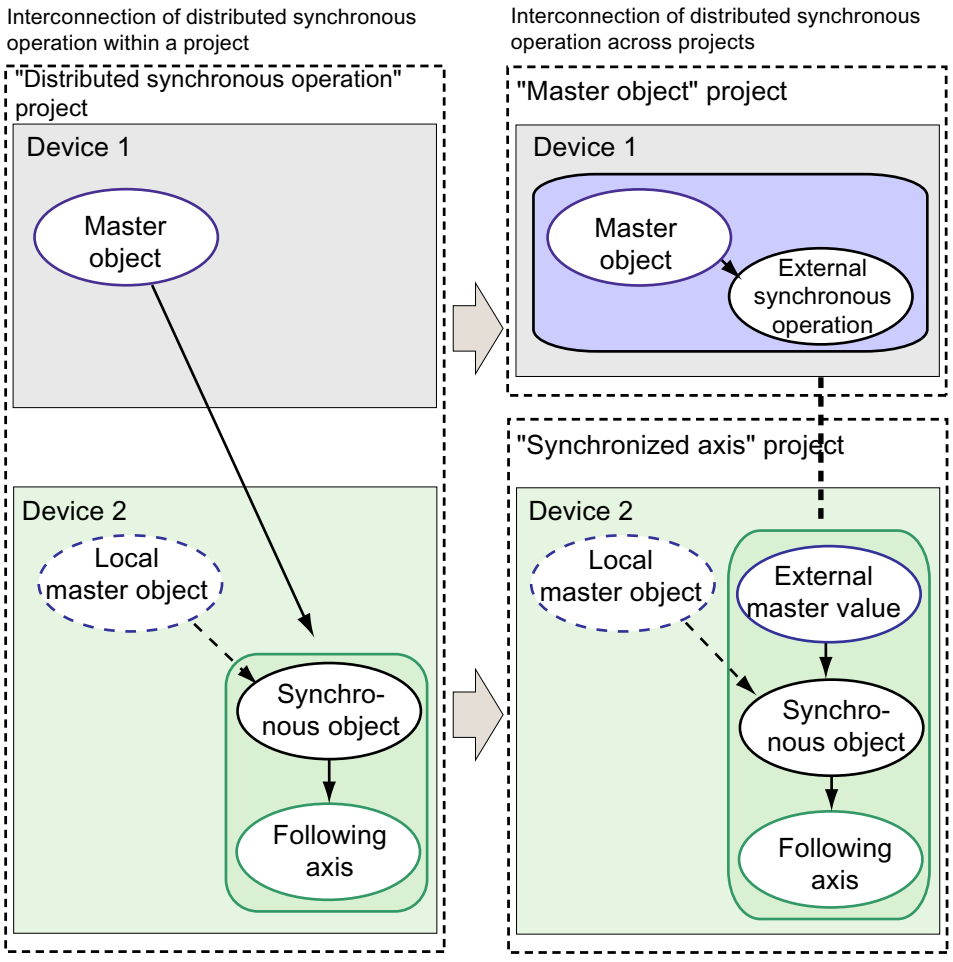


Figure 3-19 Configuration of a cross-project distributed synchronous operation

This figure shows the logical view of a synchronous operation interconnection. If the objects are located in the same project, the master object (master) and the synchronous object (slave) can be directly interconnected (see left-hand side of figure).

If the master object and synchronous object are in different SIMOTION devices that are not located in the same project, they must be interconnected using proxy objects. In each case, these represent the external object (see right-hand side of figure). External synchronous operation reflects the "transmitter master value," whereas the external master value reflects the "receiver master value."

This description uses the following terms:

Table 3-2 Terms used

Master/slave	PROFINET IO	PROFIBUS DP
Master object on	IO controller	DP master
Synchronous object on	I device	I slave

The description below uses an example to explain how communication is configured for cross-project distributed synchronous operation.

Communication configuration for PROFIBUS DP

The following steps need to be carried out for configuration using PROFIBUS DP:

- Create and configure a "synchronized axis" project; see Section Creating and configuring a "synchronized axis" project (Page 180).
- Create and configure a "master object" project; see Section Creating and configuring a "master object" project (Page 187).
- Create proxy objects; see Section Creating proxy objects (Page 205).

Communication configuration for PROFINET

The following steps need to be carried out for configuration using PROFINET:

- Create and configure a "synchronized axis" project; see Section Creating and configuring "Synchronous axis" project on SIMOTION D455-2 (Page 191).
- Create and configure a "master object" project; see Section Creating and configuring "Master Object" project on SIMOTION D455-2 (Page 199).
- Create proxy objects; see Section Creating proxy objects (Page 205).

See also

PROFIBUS communication configuration (Page 179)

Communication configuration via PROFINET IO (Page 190)

Proxy objects (Page 205)

3.5.2 PROFIBUS communication configuration

3.5.2.1 Communication via PROFIBUS DP

This section uses an example to describe how cross-project distributed synchronous operation is configured using PROFIBUS DP. You can adapt the project properties to suit your system requirements. Please note that you must set the entire PROFIBUS configuration. This example only refers to configuring cross-project distributed synchronous operation.

Introduction

If the SIMOTION devices are connected via PROFIBUS, data exchange takes place between a DP master and a DP slave. The SIMOTION device for the "master object" project is configured as the DP master and the SIMOTION device for the "synchronized axis" project is configured as the DP slave. In the "master object" project, in order to configure data exchange with the DP slave in the "synchronized axis" project the DP slave on the PROFIBUS line must be configured with the "GSD file" of the SIMOTION device.

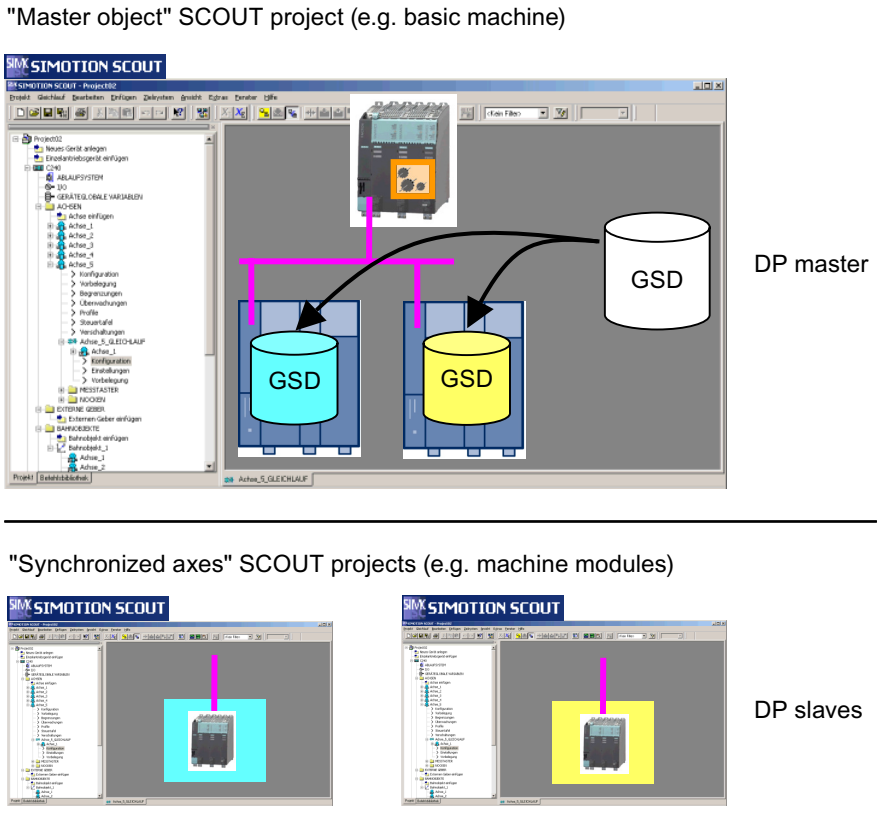


Figure 3-20 PROFIBUS DP: HW configurations of DP slaves via GSD

3.5.2.2 Creating and configuring a "synchronized axis" project

This description is just an example. You can adapt the project properties to suit your system requirements. In this example, communication takes place via PROFIBUS.

Creating a DP slave (project creation in SIMOTION SCOUT)

1. Open SIMOTION SCOUT and create a new project named "synchronous axis".
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".
4. In the "Interface Selection - D455-2" window, select the "PROFIBUS DP1 (X126)" interface by clicking "OK" and click "OK" again to confirm.
   HW Config opens.

Configuration of the interface in HW Config

1. In the rack, double-click the "DP" interface.
   The "Properties - DP - (R0/S2.1)" window opens.
2. Click the "Properties..." button on the "General" tab.
   The "Properties - PROFIBUS interface DP - (R0/S2.1)" window opens. Address 2 is displayed as the default PROFIBUS address. Leave this setting unchanged.
3. Click the "New..." button in the "Subnet" area to create a subnet.
   The "Properties - New PROFIBUS subnet" window opens.
4. On the "General" tab, enter the name "Synchronous operation" at "Name".
5. Click the "Network settings" tab and set the recommended value of 12 MBit/s as the transmission rate.
6. Click "OK" to confirm your settings. You will then be in the "Properties - DP1 - (R0/S2.1)" window.

Setting DP slave

1. Click the "Operating Mode" tab and activate "DP Slave."
   Ensure that the option button "Programming, status/control or other PG functions and non-configured communication connection possible" is not activated, deactivate it if necessary. When the option is activated, no isochronous operation is possible.

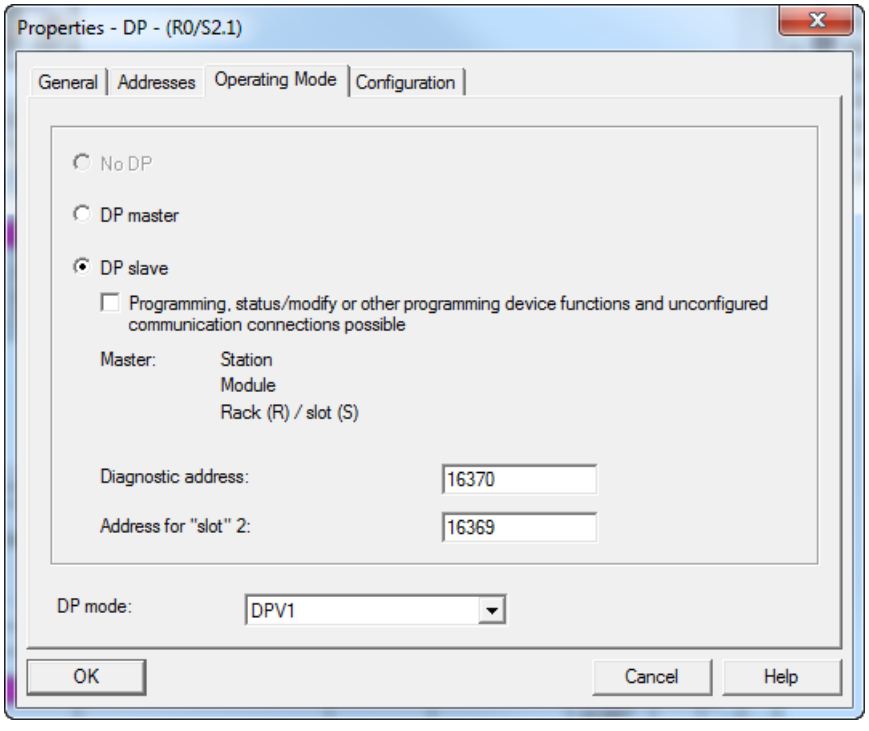


Figure 3-21 Setting the properties for the DP slave

2. Click the "Configuration" tab to create input and output addresses for communication.

3. Click the "New..." button.
   The "Properties - DP - (R0/S2.1) - Configuration - Line 1" window opens.
   Enter the following settings for the input data:
   - Address type: Input
   - Address: 256
   - Length: 12
   - Unit: Word
   - Consistency: Unit

   , and click "OK" to confirm your settings.

   Note

   In this example, you enter 256 as the first available address. If you have already configured other objects in the project, such as a drive, you will need to enter the first potentially available address.

   Please note that you will need the set input and output addresses later when configuring the external master value in SIMOTION SCOUT.

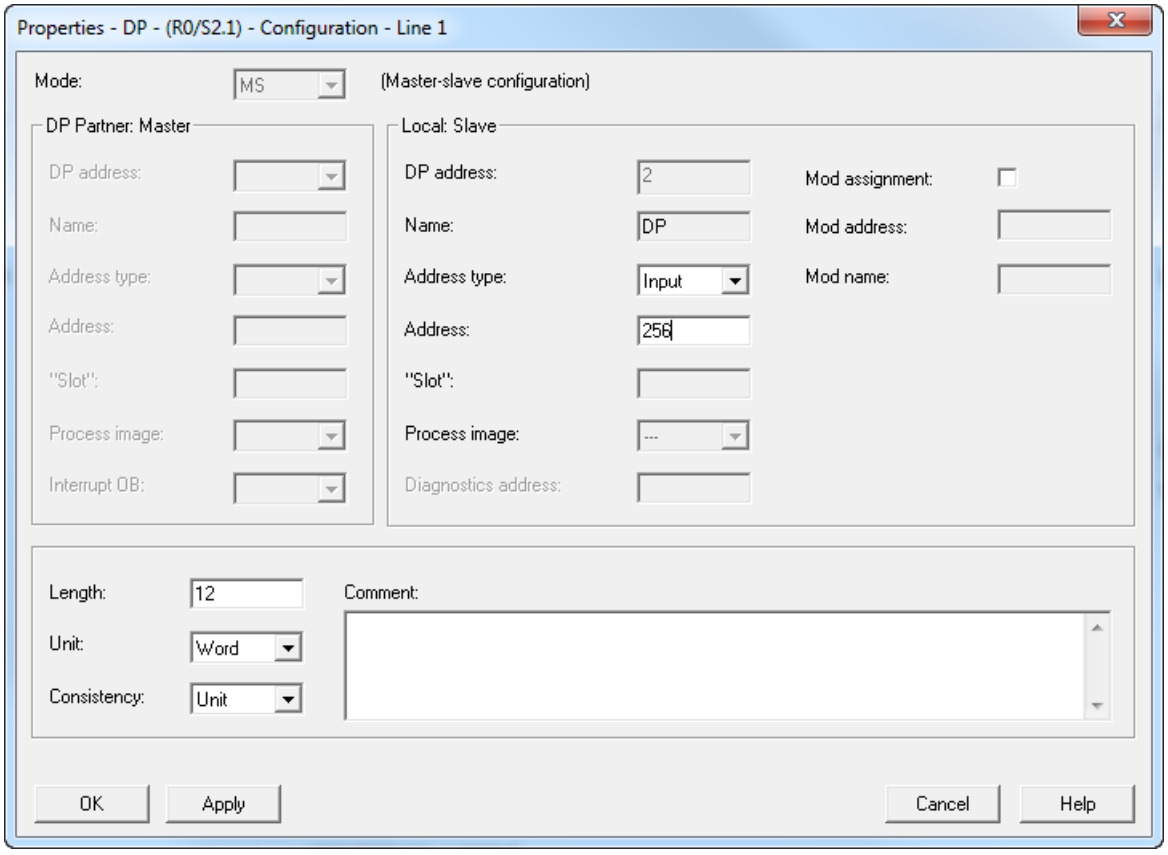


Figure 3-22 Configuring the input address on the slave

4. Click "New...".
   The "Properties - DP1 - (R0/S2.1) - Configuration - Line 2" window opens.
   Enter the following settings for the output data:
   - Address type: Output
   - Address: 256
   - Length: 12
   - Unit: Word
   - Consistency: Unit
   , and click "OK" to confirm your settings.

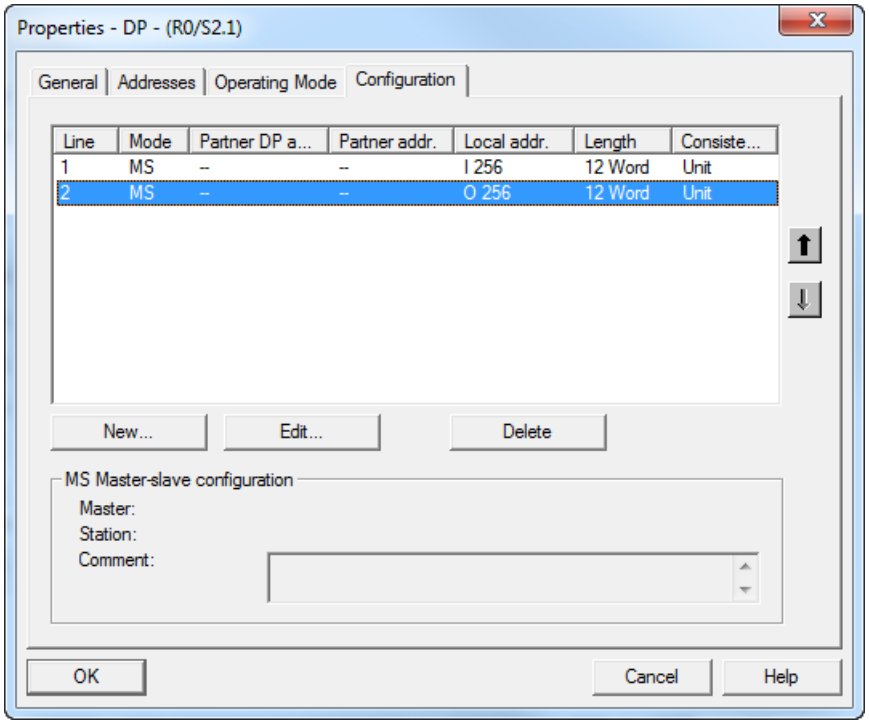


Figure 3-23 Configured input and output addresses on the DP slave

Note

Please also note that, in addition to the data for the synchronous operation, you may also want to transfer user data (communication between master and slave, such as project-specific control and status data for synchronous axes) and will, therefore, need to configure further lines with additional input and output addresses accordingly.

5. Click "OK" to confirm your settings, followed by the "Save" button.

In order to configure an isochronous PROFIBUS line, you must now use HW Config to create and configure a new station ("Dummy master") in the same project. This "dummy master" is only required to act as a proxy and does not have to be physically present.

Configuring the dummy master

It is recommended that you configure an S7 CPU as the master, so that the slave project remains clear and the dummy master is not visible in SIMOTION SCOUT.

1. Open or switch to HW Config of the DP slave.
2. In the HW Config menu bar, click "Station > New...".
   The "New" window opens. Enter "dummy master" as the object name and select the SIMATIC 300 station as the object type.

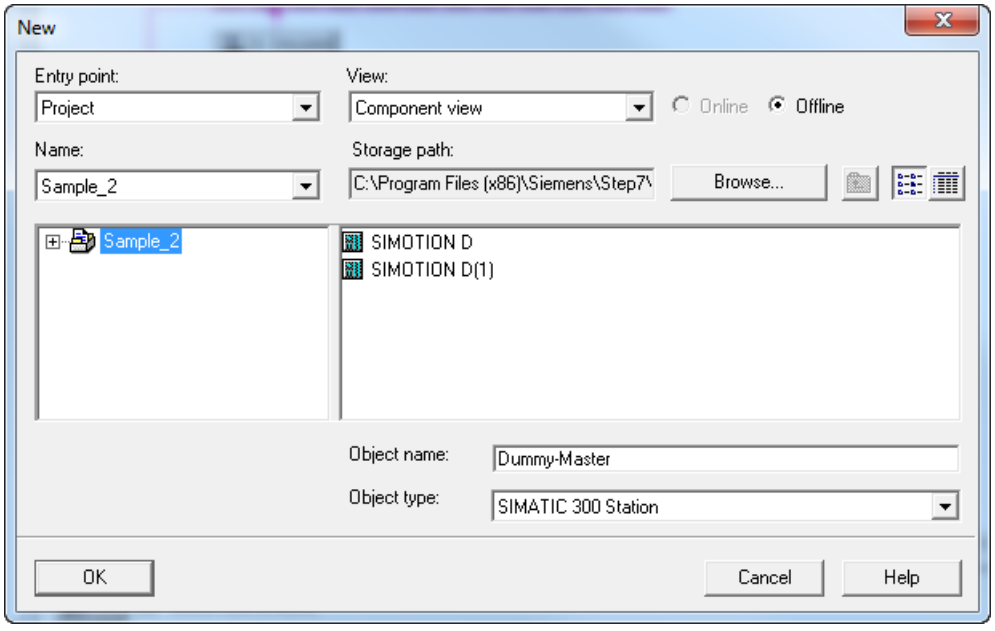


Figure 3-24 Creating a new station ("Dummy master") in HW Config

3. Click "OK" to confirm your settings. The new "Dummy master" station opens

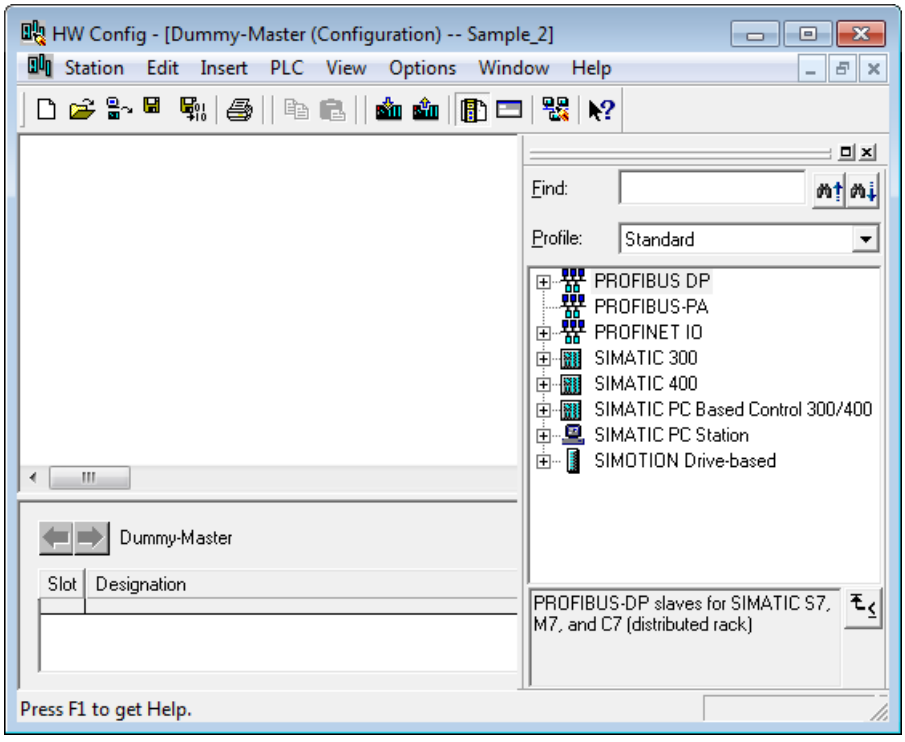


Figure 3-25 New station in HW Config

4. In the hardware catalog, open "SIMATIC 300 > RACK 300". Double-click the "Mounting rail". This is inserted in HW Config.

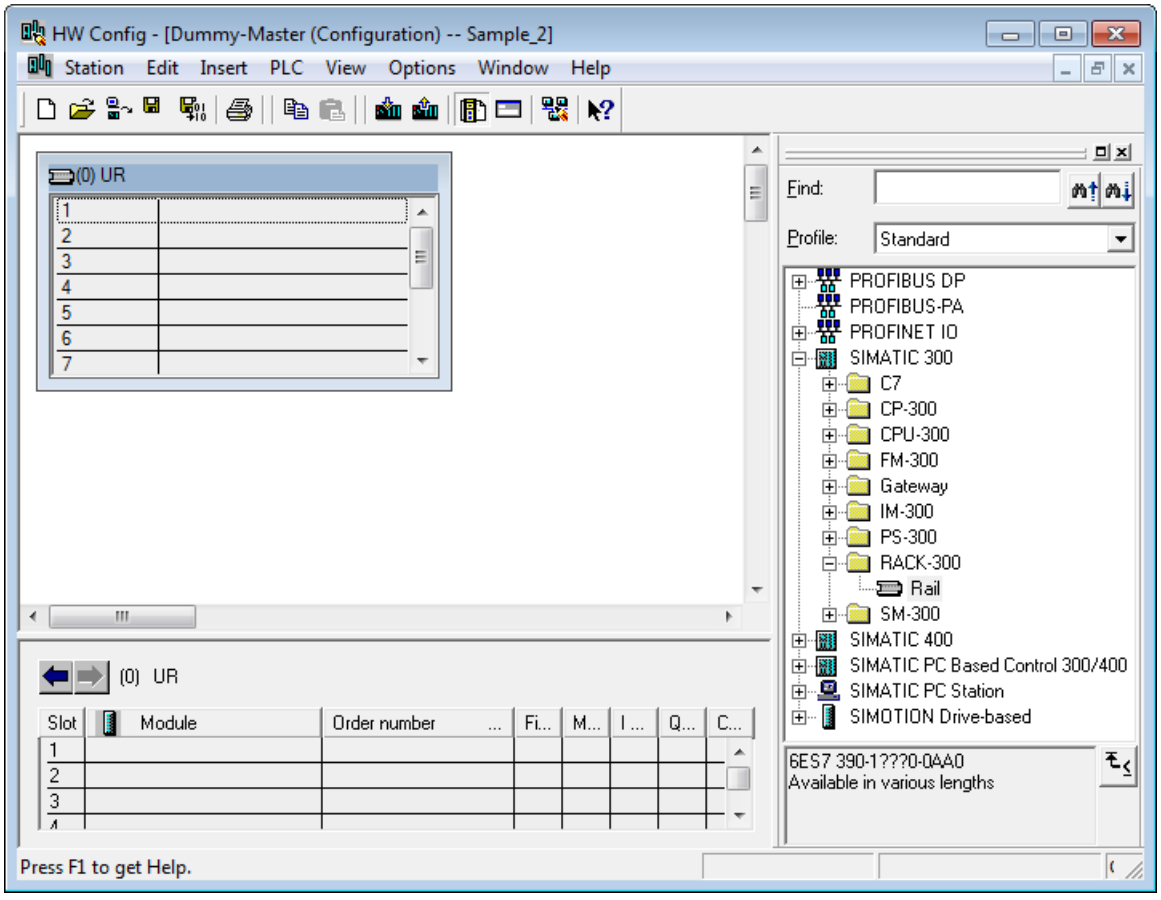


Figure 3-26 Inserting a mounting rail in HW Config

5. In the hardware catalog, open SIMATIC 300 > CPU 300 > CPU 316-2 DP > 6ES7 316-2AG00-0AB0. Select "V1.2" and move the module to its designated slot (highlighted in green) using drag-and-drop.
   The "Properties - PROFIBUS interface DP - (R0/S2.1)" window opens. It is not necessary to set the operating mode for the dummy master.
6. Select PROFIBUS address 3 as the address. The PROFIBUS address of the dummy master must be identical to that of the DP master in the "master object" project. You will create the "master object" project later.
7. Click "New...". The "New" window opens.
8. Click the "Network settings" tab and set the recommended value of 12 MBit/s as the transmission rate.
9. Click the "Options..." button. The "Options" window opens.
10. Activate the "Activate equidistant bus cycle" function and enter 3 ms for the "Equidistant DP cycle." This value must also be set later in the "master object" project.

11. Click "OK" to confirm your settings in the open window, followed by the "Save" button.
12. Switch to SIMOTION SCOUT and save the "synchronous axis" project.
    This concludes configuration of the DP slave communication in HW Config. The next step involves creating the "master object" project in SIMOTION SCOUT, see section Creating and configuring a "master object" project (Page 187).

3.5.2.3 Creating and configuring a "master object" project

This description is just an example. You can adapt the project properties to suit your system requirements. In this example, communication takes place via PROFIBUS.

Creating a "Master object" project in SIMOTION SCOUT

1. Open SIMOTION SCOUT and create a new project named "Master Object."
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".
4. In the "Interface selection - D435" window, select the "PROFIBUS DP1 (X126)" interface by clicking "OK" and click "OK" again to confirm.
   The HW Config opens.

Configuration of the interface in HW Config

1. In the rack, double-click the "DP" interface.
   The "Properties - DP - (R0/S2.1)" window opens.
2. Click the "Properties" button on the "General" tab.
   The "Properties - PROFIBUS interface DP - (R0/S2.1)" window opens.
3. Select the PROFIBUS address by putting 3 in the "Address" field. The address of the DP master must be identical to that of the dummy master in the "synchronous axis" project.
4. Click the "New..." button in the "Subnet" area to create a subnet.
   The "Properties - New PROFIBUS subnet" window opens.
5. On the "General" tab, enter the name "Synchronous operation" at "Name".
6. Click the "Network settings" tab and set the recommended value of 12 Mbit/s as the transmission rate, then click "Options".
   The "Options" window opens.
7. Activate the "Activate equidistant bus cycle" function and enter 3 ms for the "Equidistant DP cycle."
8. Click "OK" to confirm your settings in the open window.

Setting DP Slave

1. In the hardware catalog, open "PROFIBUS DP > Additional field devices > PLC > SIMOTION" and select "SIMOTION D4xx".
2. Using drag-and-drop, move "SIMOTION D4xx" (DP slave) to the "Synchronous operation: DP master system (2)" subnet. The "Properties – PROFIBUS interface SIMOTION D4xx" window opens.
3. Select PROFIBUS address 2 as the address and confirm the settings with "OK". This PROFIBUS address must be identical to that of the DP slave in the "synchronous axis" project.

4. Insert a 12-word module from the hardware catalog for the slave outputs first. Under "SIMOTION D4xx" in the hardware catalog, select the "Master_O Slave_I 12Wo/Unit" module and move it to its designated slot (highlighted in green) using drag-and-drop.

Note

Please note that the the inputs and outputs must always be configured in the opposite direction. An input slot must always go to an output slot (and vice-versa). That means that if the first slot is configured as an input in the "synchronous axis" project, then the first slot in the "master object" project has to be configured as an output. The length of configured inputs and outputs must always be identical.

The addresses for the inputs and outputs must lie above the first 64 bytes of the logic address area. You will need this address again later for configuring the proxy object.

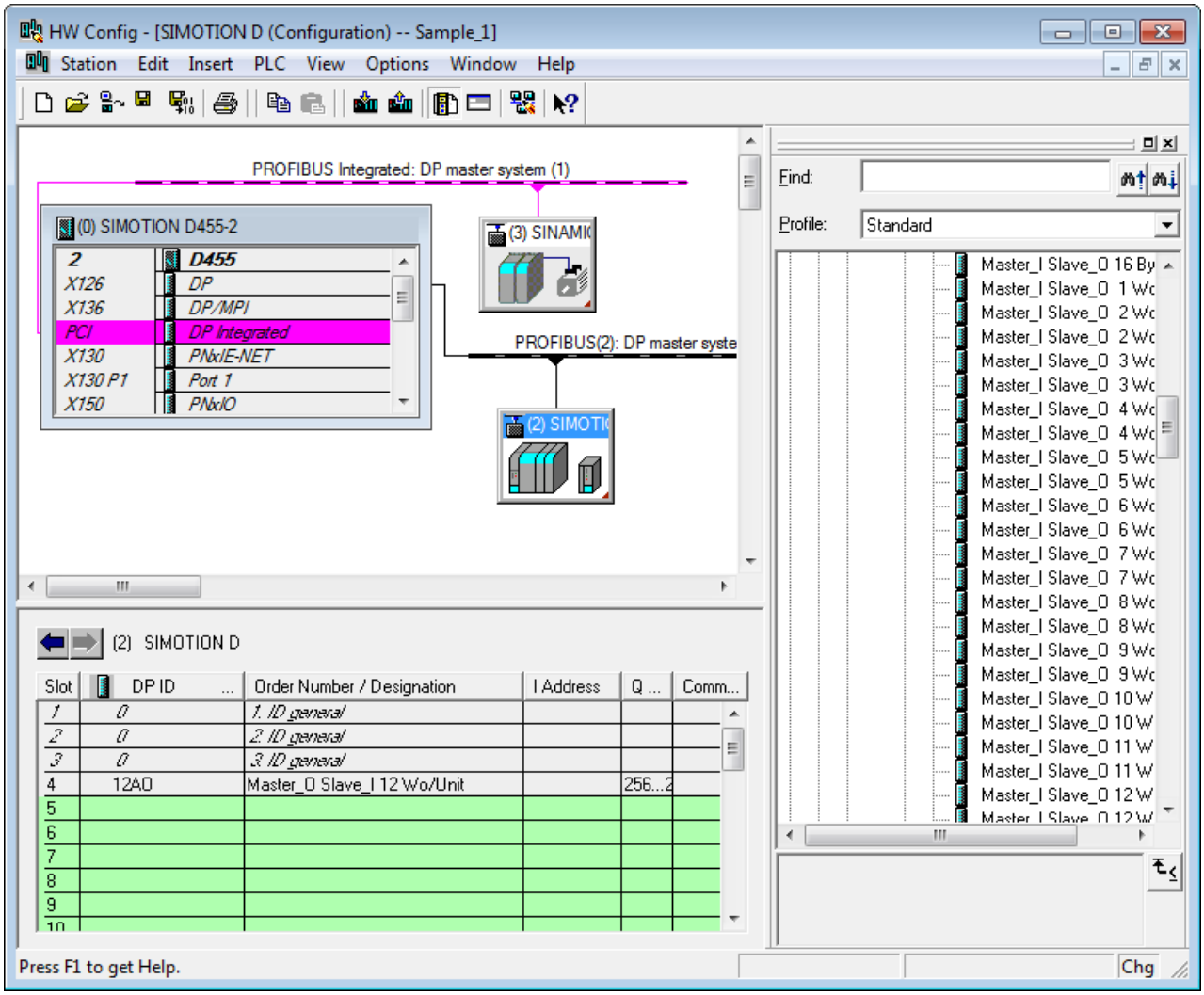


Figure 3-27 Creating an output module on the DP slave in HW Config

5. Under "SIMOTION D4xx" in the hardware catalog, select the "Master_I Slave_O 12Wo/Unit" module and move it to its designated slot (highlighted in green) using drag-and-drop.

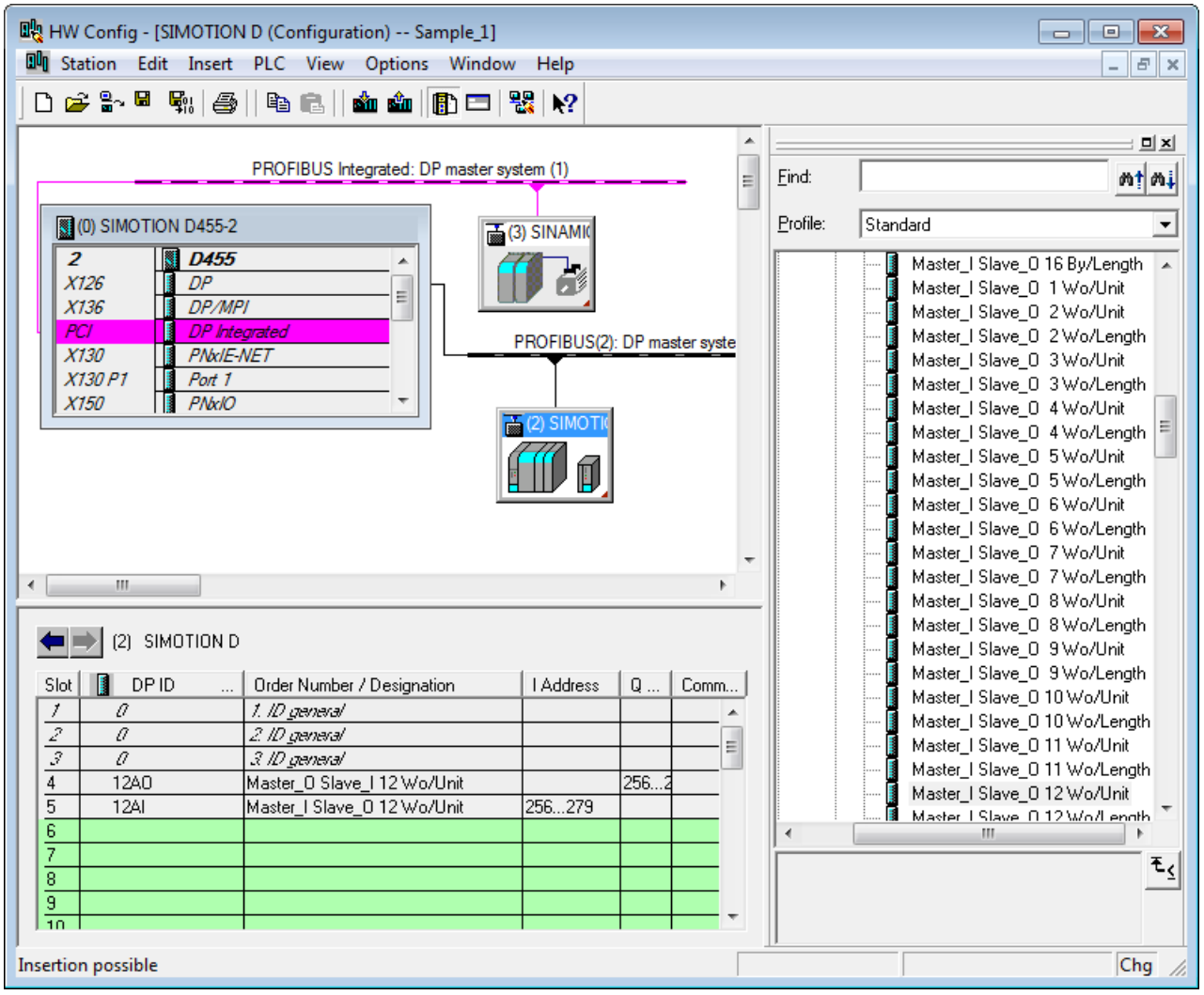


Figure 3-28 Creating an input module on the DP slave in HW Config

6. Click the "Save" button.
7. Switch to SIMOTION SCOUT and save the "master object" project.
   This concludes configuration of the DP master communication in HW Config. The next step involves creating proxy objects in SIMOTION SCOUT, see section Creating proxy objects (Page 205).

3.5.3 Communication configuration via PROFINET IO

3.5.3.1 Communication via PROFINET IO

This section uses an example to describe how cross-project distributed synchronous operation is configured using PROFINET IO. You can adapt the project properties to suit your requirements.

Introduction

If the devices are connected via PROFINET, data exchange takes place between an IO controller and an I device. The I device must be taken into account during the configuration of the I/O controller by means of its GSDML file. The GSDML can be generated in HW Config of the I device.

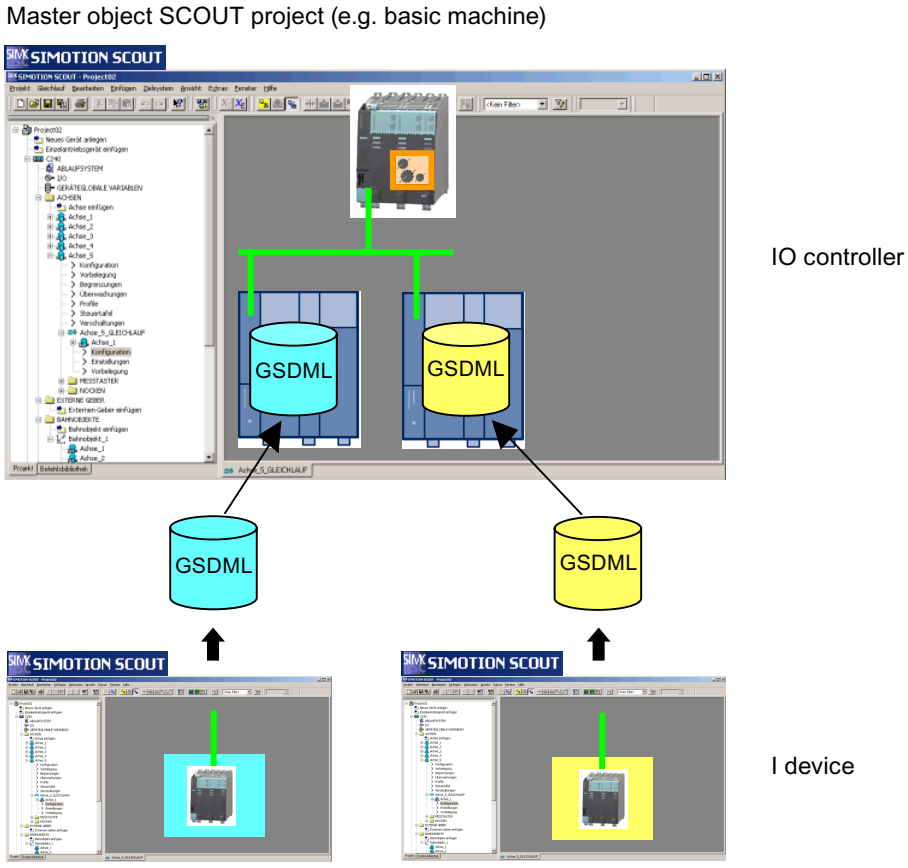


Figure 3-29 PROFINET distribution bus: HW configurations of the basic machine and modules

Note

The I device can be configured in STEP 7 as of V5.4 SP2.

Note

Distributed synchronous operation is also possible via two PN interfaces, see Section *Example configuration for controlled Sync Master* in the *Communication* System Manual.

3.5.3.2 Creating and configuring "Synchronous axis" project on SIMOTION D455-2

This description is just an example. You can adapt the project properties to suit your system requirements. In this example, configuration using PROFINET V2.2 is described. SIMATIC STEP 7 V5.5 SP1 must be installed for this.

The creation and configuration of a "synchronous axis" project on SIMOTION D455-2 consists of the following steps:

Project creation in SIMOTION SCOUT

1. Open SIMOTION SCOUT and create a new project named "synchronous axis".
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".
4. Click on "New..." in the "Properties - Ethernet interface PNxIO" window.
5. In the following window, add a new Industrial Ethernet subnet with the name "Synchronous operation".
6. Make the interface selection for PG/PC connection.
7. Click "OK" to confirm your settings. HW Config opens.

Configuration of I device in HW Config

1. In the rack, double-click the PNxIO interface to change the device name for the PROFINET interface of the I device, and enter the device name "pn1a."
   This is necessary because you are working with 2 projects and the default name is the same for both. retaining these same names would result in consistency problems at a later point.
2. In this same window, click the "PROFINET" tab and set the required send cycle clock. In this example, this is set to 2 ms.

Note

You can set the cycle clocks in two ways:

- You can set the send cycle clock so it is equal to the DP cycle clock on the drive.
- You can set the send cycle clock so that it is lower than the DP cycle clock: By opting for this you will be working with cycle clock scaling. The position control cycle clock (master application cycle) and the DP cycle on the drive must then be of an equal length.

3. In this same window, click the "I device" tab.

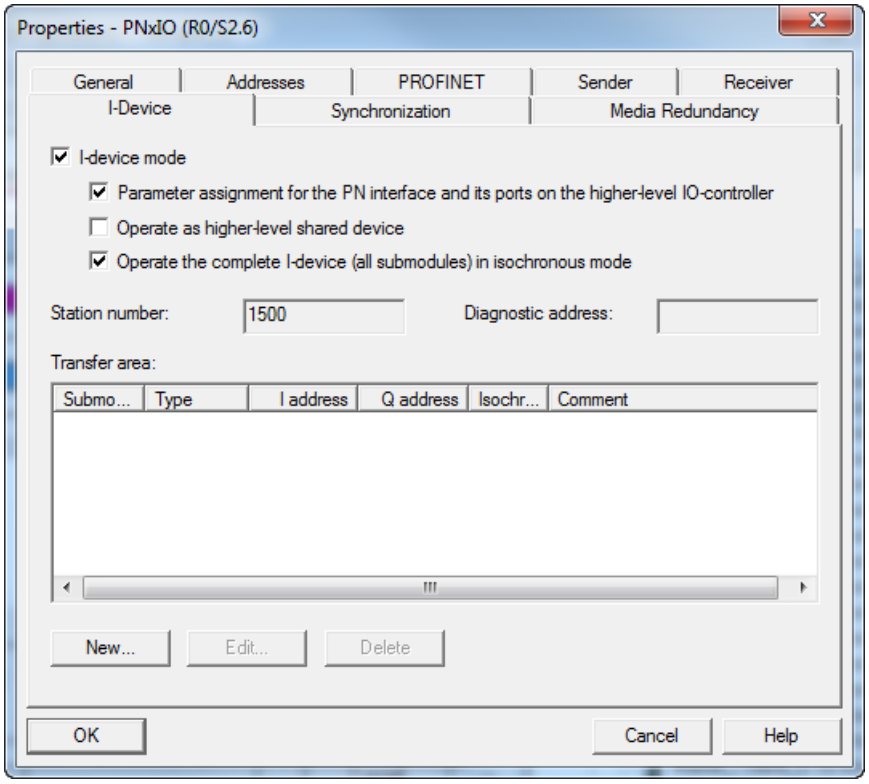


Figure 3-30 Settings on the I device

4. Activate the "I device mode", "Parameterization of the PN interface and its ports on the higher level IO controller" and "Operate complete I device (all submodules) isochronously" functions.
5. In the lower area of this same tab, click "New...".
The "Properties of virtual submodule" window opens.

6. Enter the following properties for the send and receive data:
   - Address type: Input
   - Start: 256
   - Length: 24
     and click "OK" to confirm your settings.

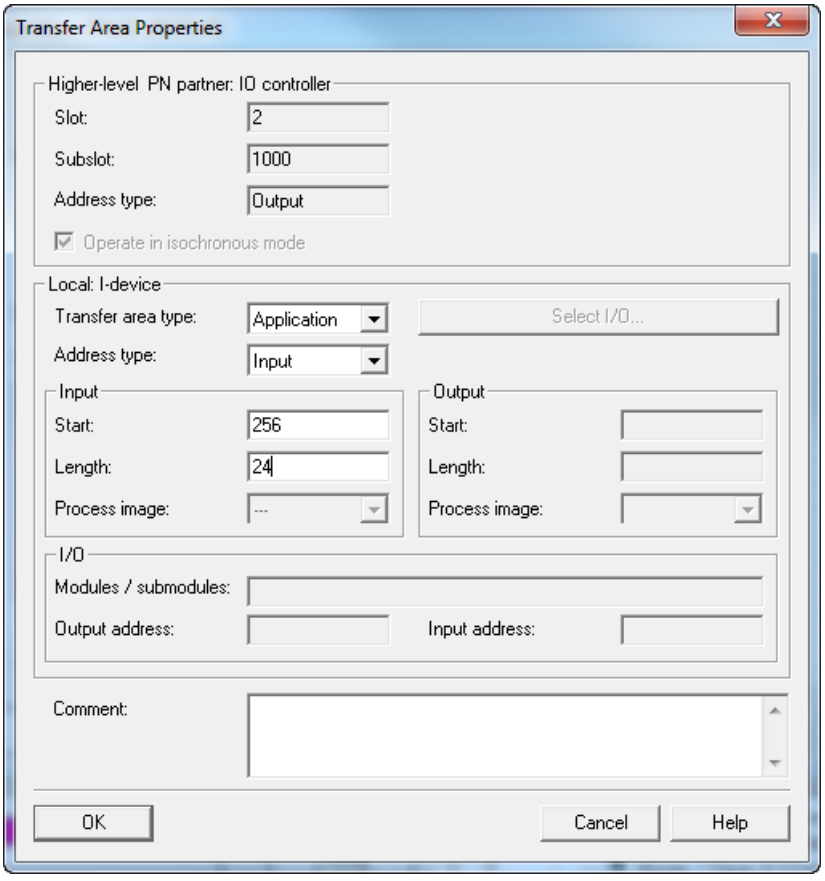


Figure 3-31 Configuring settings on the I device

7. Click the "New..." button again to configure the output address as described below:
   – Address type: Output
   – Start 256

– Length 24
and click "OK" to confirm your settings.
You will need the set input and output addresses later to configure the proxy object in SIMOTION SCOUT.

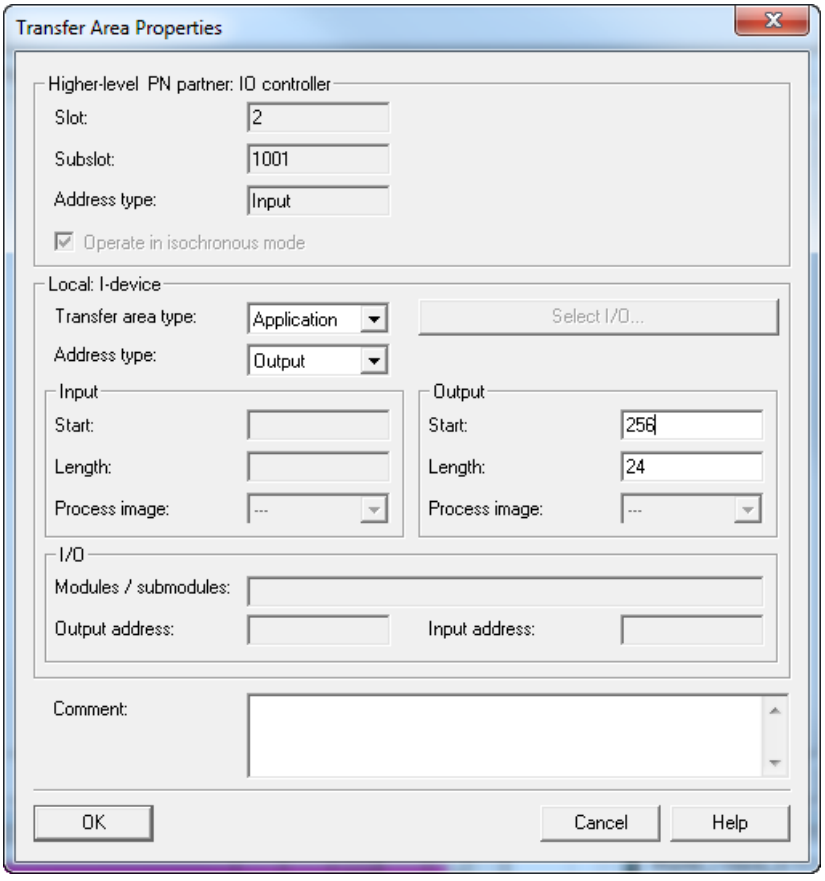


Figure 3-32 Configuring output addresses on the I device

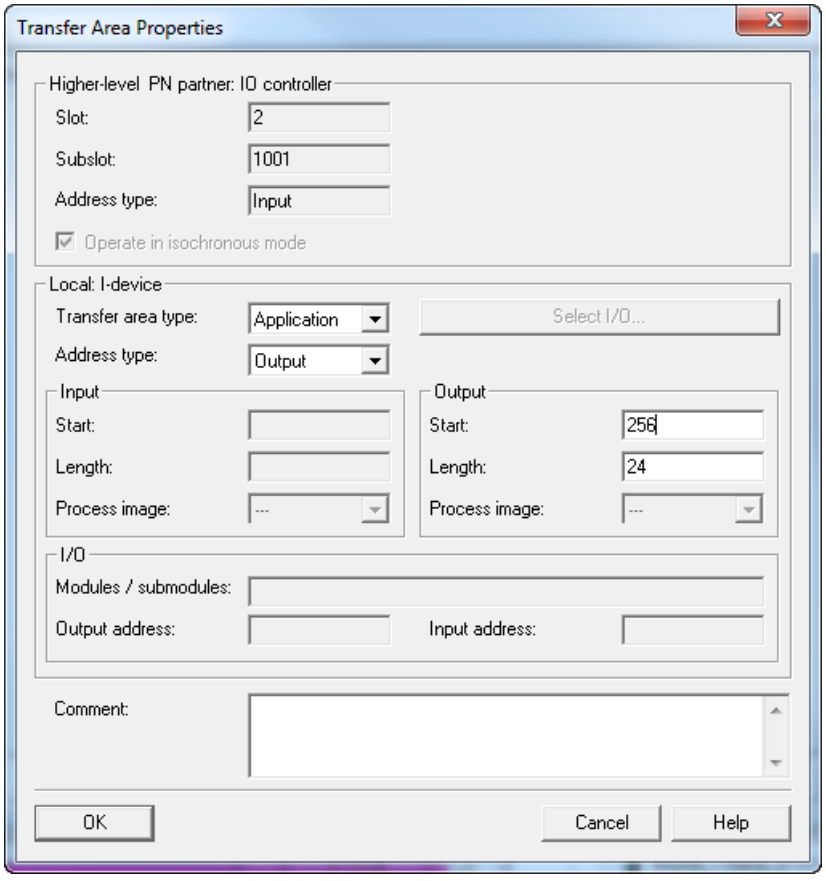


Figure 3-33 Settings on the I device - Addresses

Note

Please note that, in addition to the data for the synchronous operation, you may also want to transfer user data (communication between master and slave, such as project-specific control and status data for synchronous axes) and will, therefore, need to configure additional inputs and outputs accordingly.

8. Confirm your entries with "OK". The I device is created.

Setting DP cycle

1. Double-click on SINAMICS_Integrated.
   The "DP slave properties" window opens.
2. Click the "Isochronous operation" tab and set the "DP cycle" to factor 16. This corresponds to a DP cycle clock of 2 ms.
3. Confirm your entry with "OK".
4. Click the "Save" button.

Creating the GSD file for the I device

1. On the menu bar, select Options > Create GSD for I Device.
   The "Create GSD for I Device" window opens.

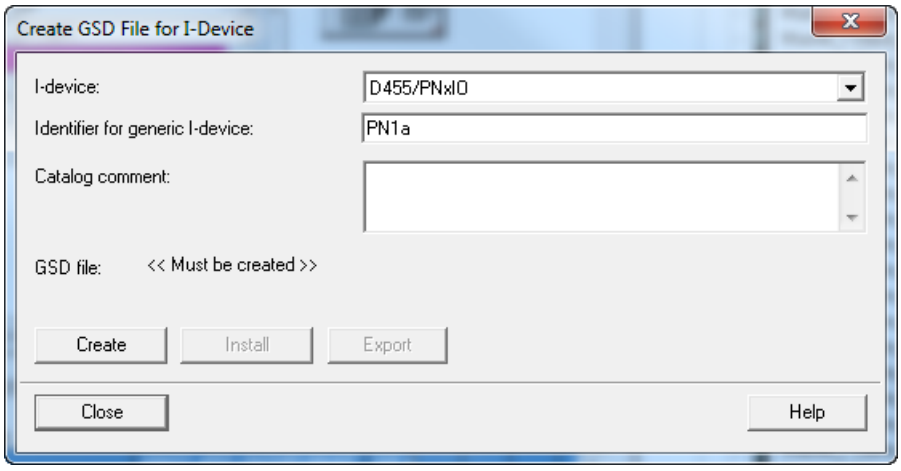


Figure 3-34 Creating the GSDML file

2. Click the "Create" button. This creates a GSDML file for the I device. You will need this file as the proxy for the I device in the "master object" project on the PROFINET IO controller. The name of the GSDML file is shown.
3. Click the "Install" button.

4. Click "OK" to confirm the message that appears, followed by the "Close" button. The GSDML file will now be entered in the HW Config hardware catalog in the PROFINET IO > Preconfigured Stations/D455 structure.

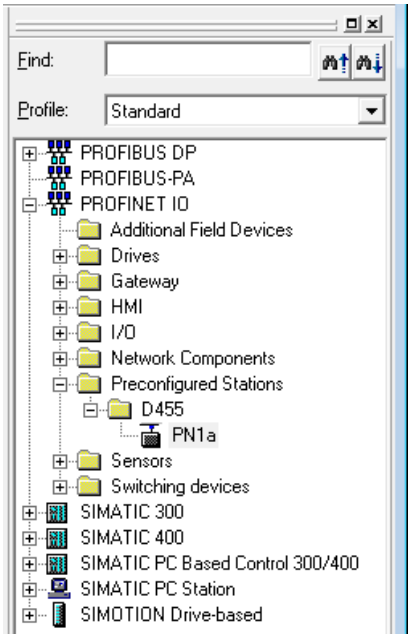


Figure 3-35 View of HW Config after installation of GSDML file

Note

If the project for the PROFINET IO controller is not to be executed with the same engineering system, you must export the GSDML file. The GSDML file must then be imported into HW Config of the other engineering system.

5. Click the "Save" button.
6. Switch to SIMOTION SCOUT and save the "Synchronous axis" project.
This concludes configuration of the I device communication in HW Config. The next step involves creating the "master object" project in SIMOTION SCOUT, see section Creating and configuring "Master Object" project on SIMOTION D455-2 (Page 199).

3.5.3.3 Creating and configuring "Master Object" project on SIMOTION D455-2

This description is just an example. You can adapt the project properties to suit your system requirements.

The creation and configuration of a "master object" project for communication via PROFINET IO consists of the following steps:

Project creation in SIMOTION SCOUT

1. Create a new project in SIMOTION SCOUT and call it "Master object".
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".

4. Click on "New..." in the "Properties - Ethernet interface PNxIO" window.
5. In the following window, add a new Industrial Ethernet subnet with the name "Synchronous operation".
6. Make the interface selection for PG/PC connection.
7. Click "OK" to confirm your settings. HW Config opens.

Setting synchronization type

1. In the rack, double-click on the PNxIO interface.
   The "Properties - PNxIO (R0/S2.10)" window opens.
2. Click the "Synchronization" tab and set Sync-Master as the synchronization type.

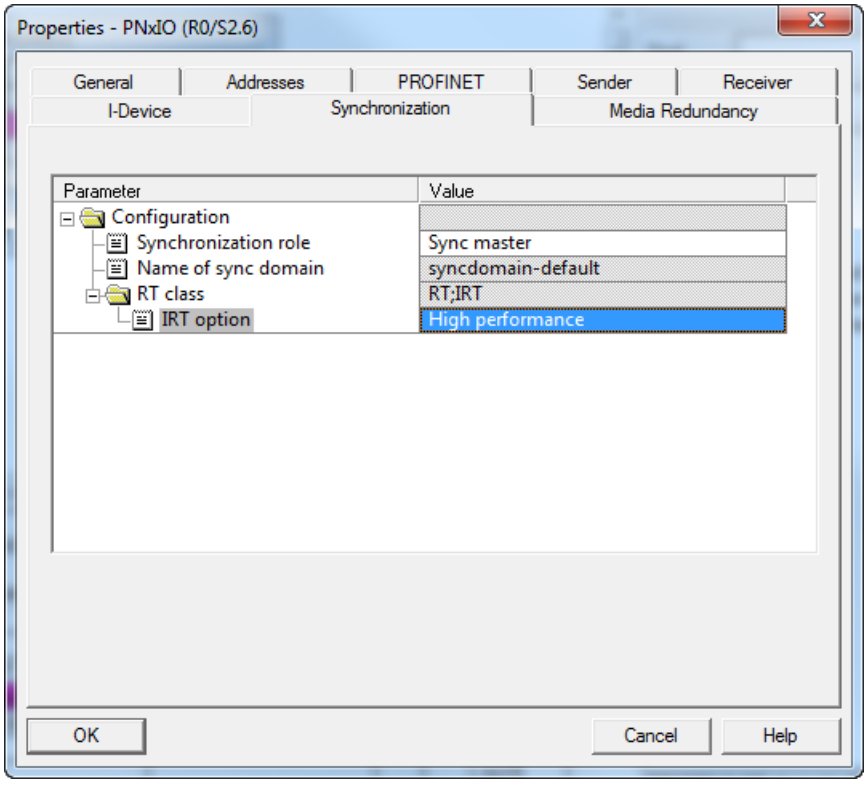


Figure 3-36 Setting Sync-Master on the IO controller

3. Click "OK" to confirm your settings.
4. On the menu bar, select "Edit > PROFINET IO > Domain management..."
   The "Domain management - Synchronous operation" window opens.

5. Set the required send cycle clock: In our example, this is 2 ms.

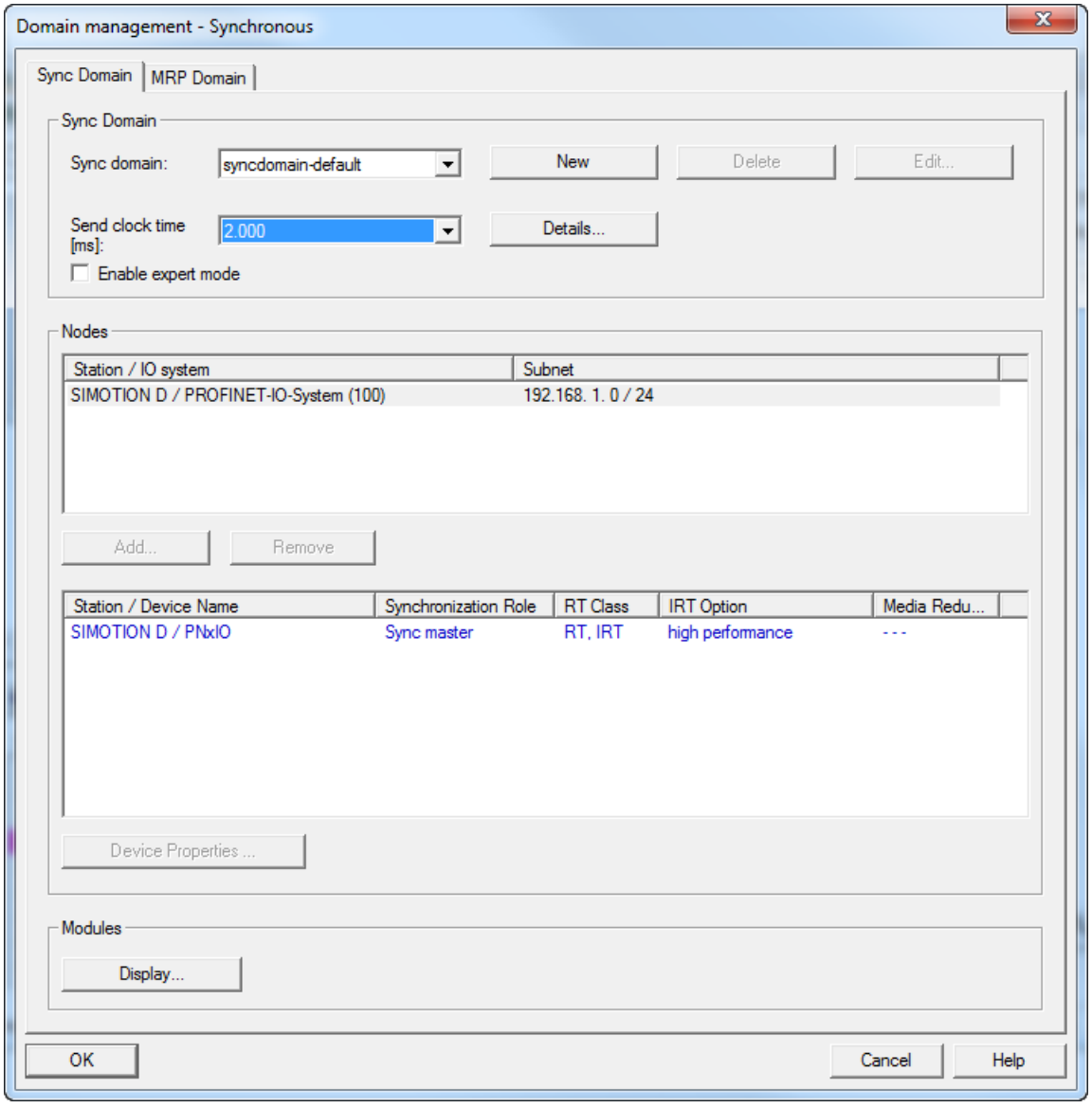


Figure 3-37 Setting the sending cycle on the IO controller

6. Click "OK" to confirm your settings.

Interconnect to PROFINET IO I device

1. Using drag-and-drop, move the I device from the hardware catalog to the "Synchronous operation: PROFINET IO system (100)" subnet. The proxy for the I device is located in the hardware catalog in the PROFINET IO > Preconfigured Stations > D455 structure.

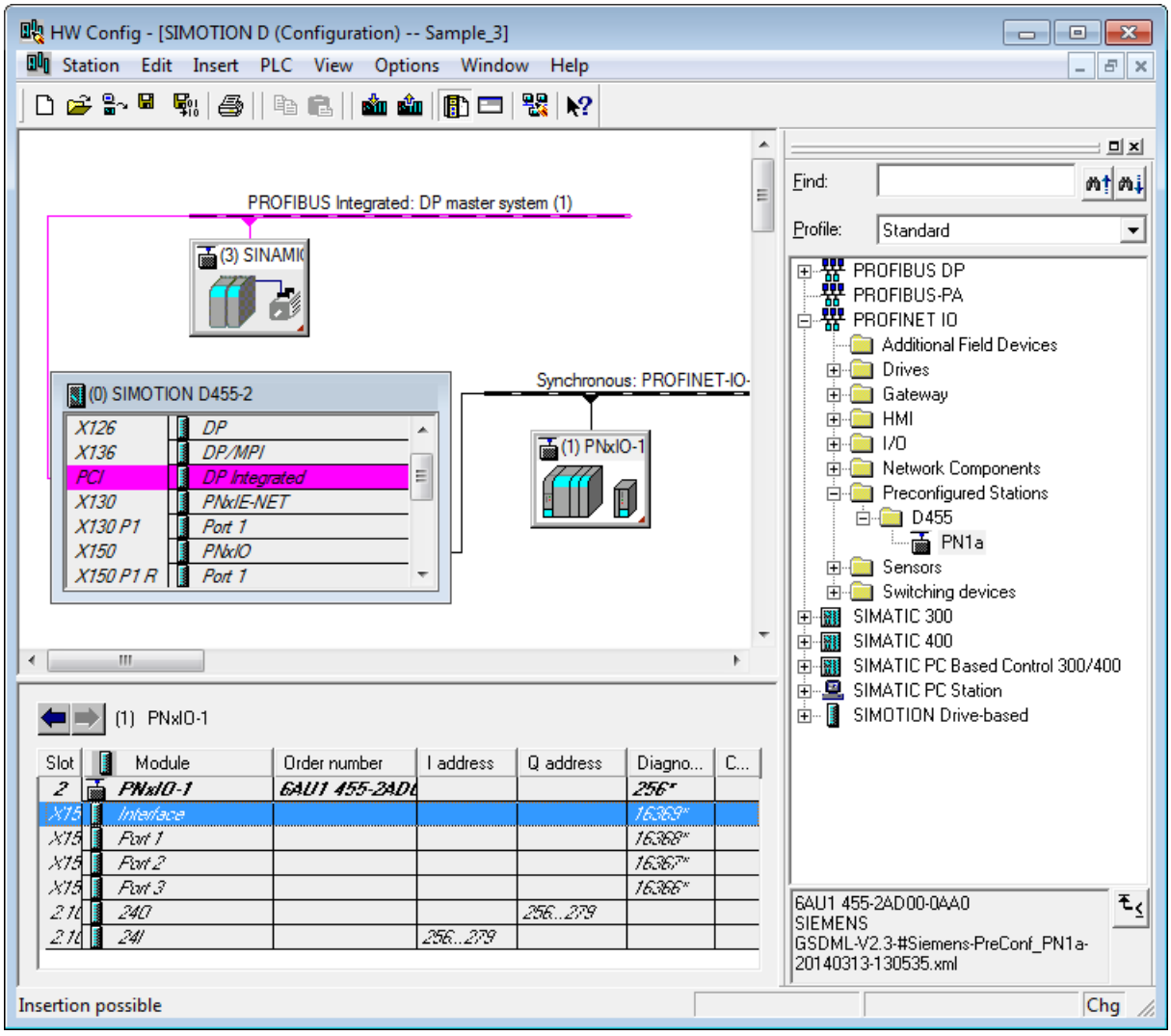


Figure 3-38 View of HW Config after insertion of the I device

Note

When you insert the proxy for the I device, the logic addresses for the cyclic input and output data of HW Config are preassigned. If necessary, correct these addresses before continuing with the configuration of the proxy objects in SIMOTION SCOUT.

2. Select the I device and double-click "Interface" in the detail view.
   The "Properties - Interface (X150)" window opens.
3. Click the "Synchronization" tab and set Sync-Slave as the synchronization type.

4. In this same window, click the "IO Cycle" tab and select the "Servo" entry in the "IO-Device/ assign isochronous application" drop-down list.

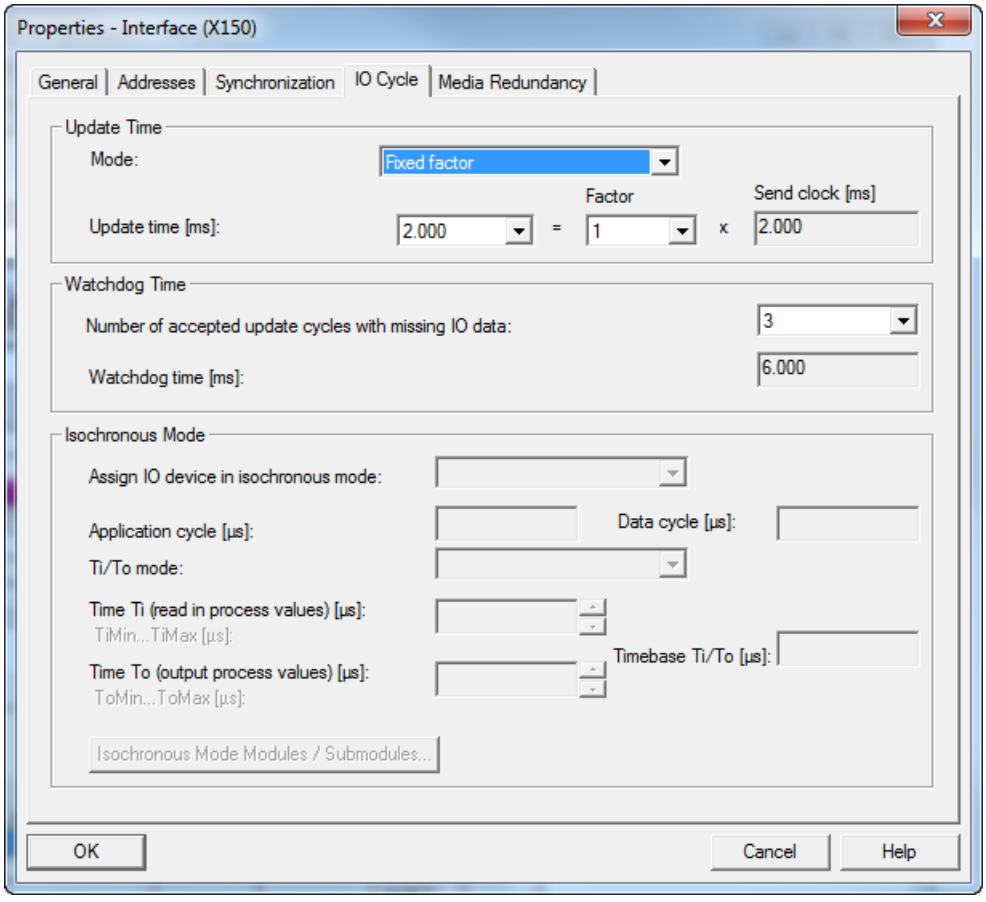


Figure 3-39 Setting isochronous mode

5. Click "OK" to confirm your settings.

Setting DP cycle

1. Double-click the SINAMICS_Integrated.
   The "DP slave properties" window opens.
2. Click the "Isochronous operation" tab and set the "DP cycle" to factor 16. This corresponds to a DP cycle clock of 2 ms.
3. Confirm your entry with "OK".

Interconnecting ports

1. Interconnect the ports by double-clicking on port 1 of the PNxIE net in the rack.
   The "Properties - PN-IO - Port 1 (R0/S2/X1400 P1 R)" window opens.
2. Click the "Topology" tab.

3. At "Partner", select the required port;
   In our example, this is "SIMOTION D\(1)pn1a\Port 1 (X1400 P1)."

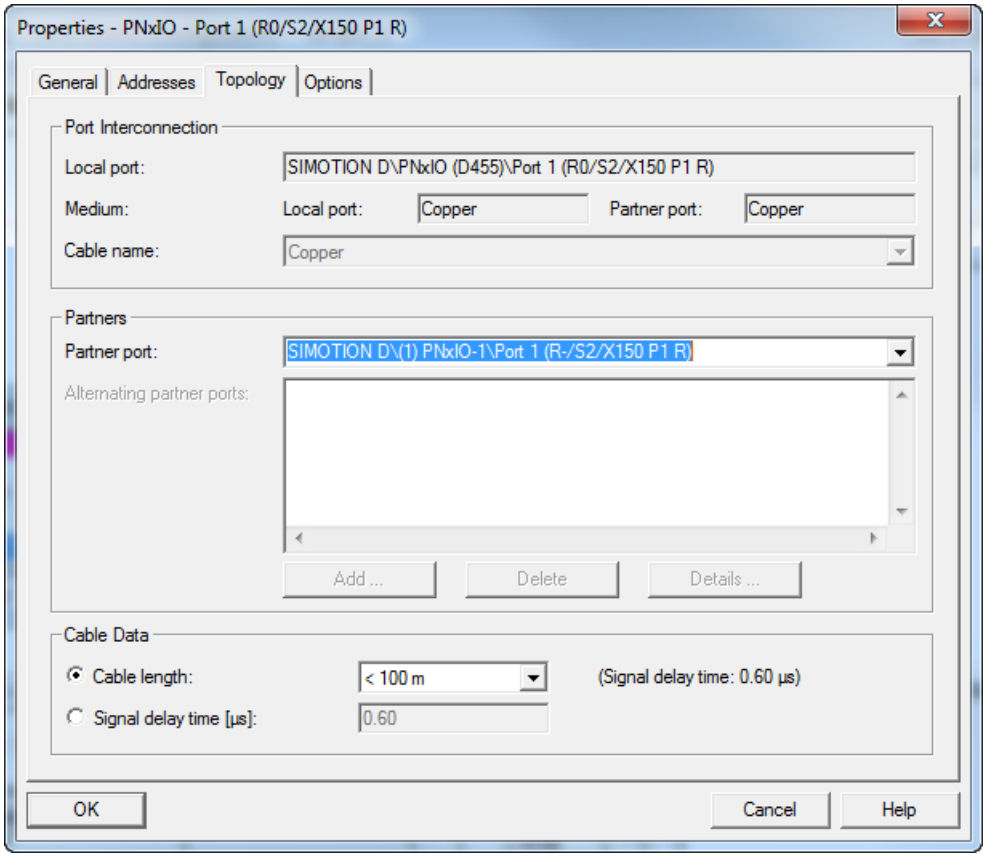


Figure 3-40 Interconnecting ports

4. Confirm your entry with "OK".
5. Click the "Save" button.
6. Switch to SIMOTION SCOUT and save the "master object" project.
   This concludes configuration of the IO controller communication in HW Config. The next step involves creating proxy objects in SIMOTION SCOUT, see section Creating and configuring "Synchronous axis" project on SIMOTION D455-2 (Page 191).

3.5.4 Proxy objects

3.5.4.1 Proxy object types

Proxy object types

There are two different types of proxy objects:

- **External master value** (**ExternalMasterType**): Proxy object for an external master value
  A proxy object for an external master value can only be created under a synchronous object, i.e. interconnected with it.
- **External synchronous operation** (**ExternalFollowingObjectType**): Proxy object for an external synchronous operation
  A proxy object for external synchronous operation can be created under the following technology object types, i.e. interconnected with them.
  - External encoder
  - Synchronous axis
  - Positioning axis
  - Path axis

See also

Creating proxy objects (Page 205)

Configuring proxy objects with SIMOTION scripting (Page 209)

3.5.4.2 Creating proxy objects

In SIMOTION SCOUT, you now need to create the proxy objects. This section uses an example to describe the process involved in this.

Creating TOs with a master value for external synchronous operation

1. In the "master object" project, create a virtual position axis using the axis wizard and call it "master_axis_vir".
2. Select the axis in the project navigator, followed by "Expert > Insert external synchronous operation" (in the shortcut menu).
   An object called "Ext_sync_operation" is created below the selected axis and interconnected with the "Master_axis_vir."

3. In the working area, enter the input and output addresses configured in HW Config. 256 is set for the address in this example.

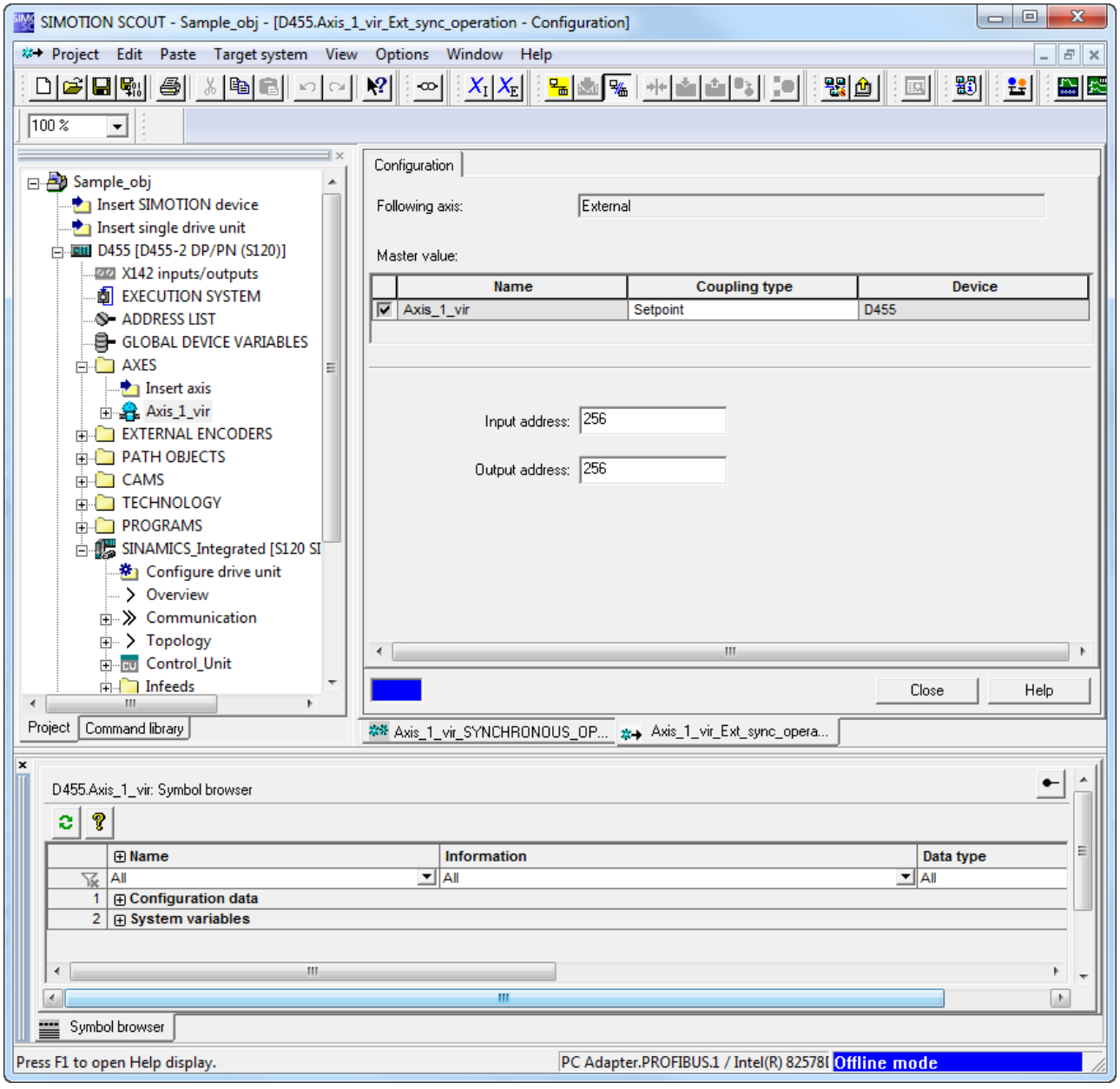


Figure 3-41 Entering input and output addresses

4. Save and close the project.

This concludes configuration of the "Ext_sync_operation" on the master object. The next step involves creating the "Ext_master_value" in the "synchronous axis" project.

Creating the synchronous axis TO with an external master value

1. Open the "synchronous axis" project.
2. Create a synchronous axis and call it "synchronous_axis_slave".

3. An "Ext_master_value" object is created below the selected synchronous object and interconnected with the synchronous object. In the project navigator, select the "synchronous_axis_slave_SYNCHRONOUS_OPERATION" synchronous object below the synchronous axis, followed by "Expert > Insert external master value" (in the shortcut menu). The "Ext_master_value" window opens in the working area.

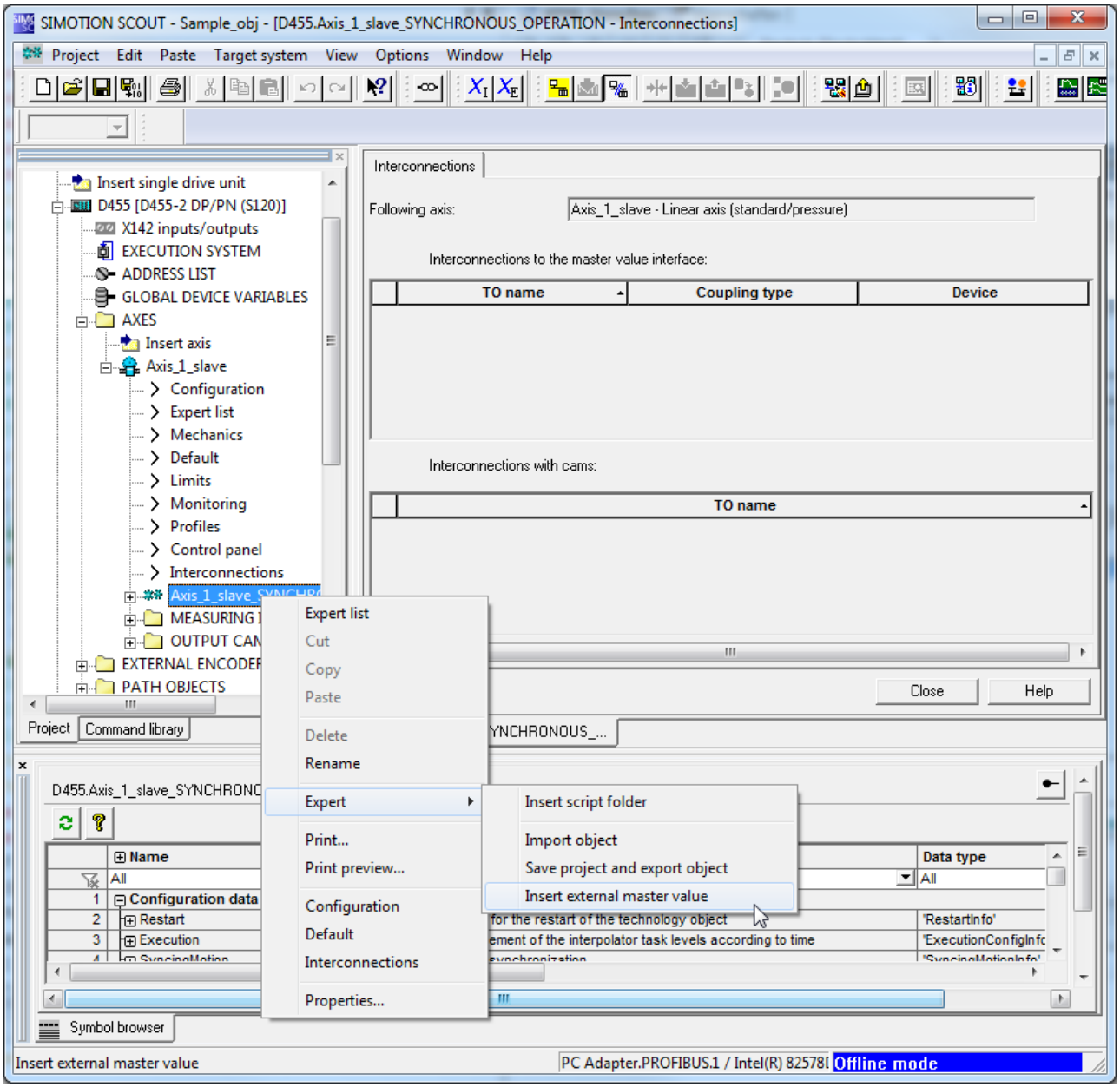


Figure 3-42 Proxy technology object - Setting the external master value

4. In the "Ext_master_value" window within the working area, enter the input and output addresses that you set as input and output addresses in HW Config. 256 is set for the address in this example.

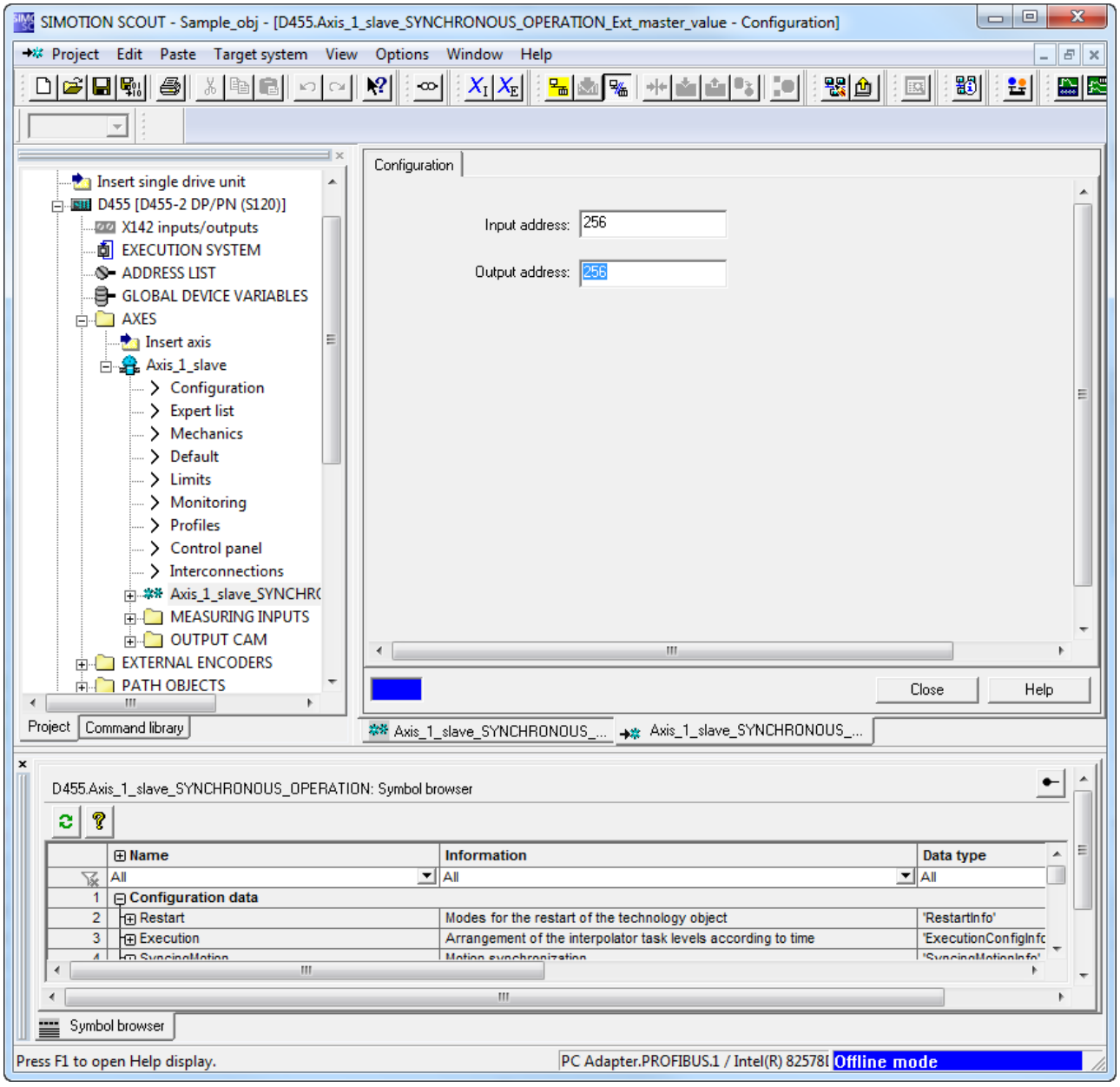


Figure 3-43 Entering input and output addresses on the external master value

This concludes configuration of the communication for cross-project distributed synchronous operation. The steps that follow this are the same as those involved in configuring a distributed synchronous operation, see section Distributed Synchronous Operation Configuration (Page 168).

3.5.4.3 Configuring proxy objects with SIMOTION scripting

The logical input and output addresses for the proxy object can be accessed with SIMOTION Scripting. Only the offline data of the object can be accessed. Further information can be found in the online help and on the Utilities & Applications DVD.

Table 3-3 Configuration data for access via scripting

DriverInfo		
	logAddressIn	Base address of the 12-word input data from HW Config
	logAddressOut	Base address of the 12-word output data from HW Config

Changing the logical input and output addresses in the SIMOTION project

An example is described below. The logical input address is changed to address 257 and the output address to address 258.

Note

If the logical input and output addresses are changed with SIMOTION Scripting within the SIMOTION SCOUT project, the project will no longer be consistent, as there is no automatic alignment with HW Config. This means that you must automate the changes to the input and output addresses in HW Config, for example via SIMATIC STEP 7 Scripting.

Proceed as follows to change the logical input and output addresses with SIMOTION Scripting:

1. In SIMOTION SCOUT, create a folder called SCRIPTS below the project. Select the name of the project, open the shortcut menu and click on **Insert script folder**.
2. Create a script.
3. Enter the following text:
   - `PROJ.TOs ("Ext_sync_operation").Symbols("DriverInfo.LogAddressIn") = 257`
   - `PROJ.TOs ("Ext_sync_operation").Symbols("DriverInfo.LogAddressIn") = 258`

4. Save the project.
5. Execute the script by selecting it and clicking on **Accept and execute** in the context menu. The logical input and output addresses are then changed.

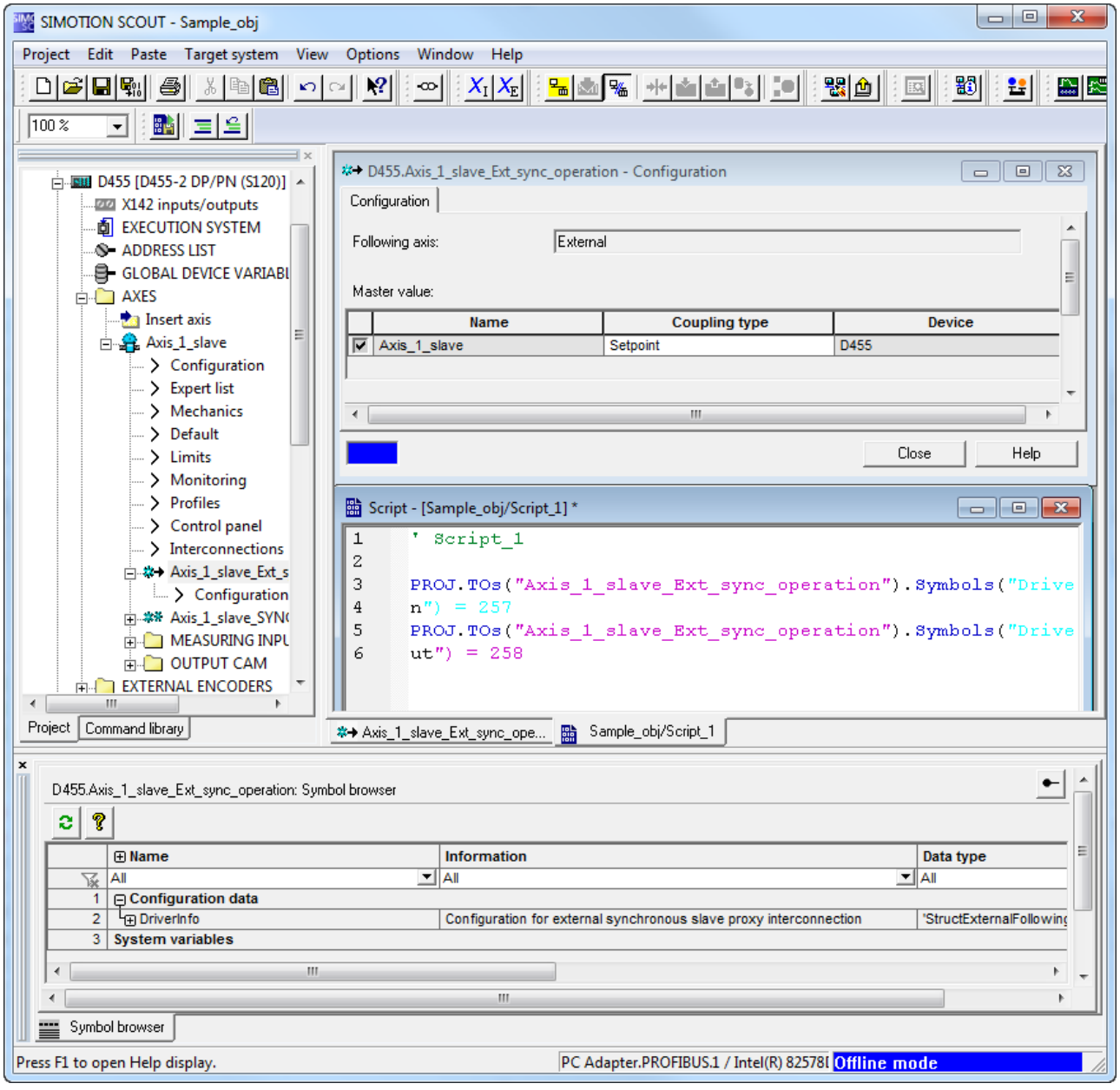


Figure 3-44 SIMOTION Scripting - example

3.5.5 Interconnection possibilities

In general, proxy objects can be interconnected with a maximum of one technology object. The following figure shows the interconnection options.

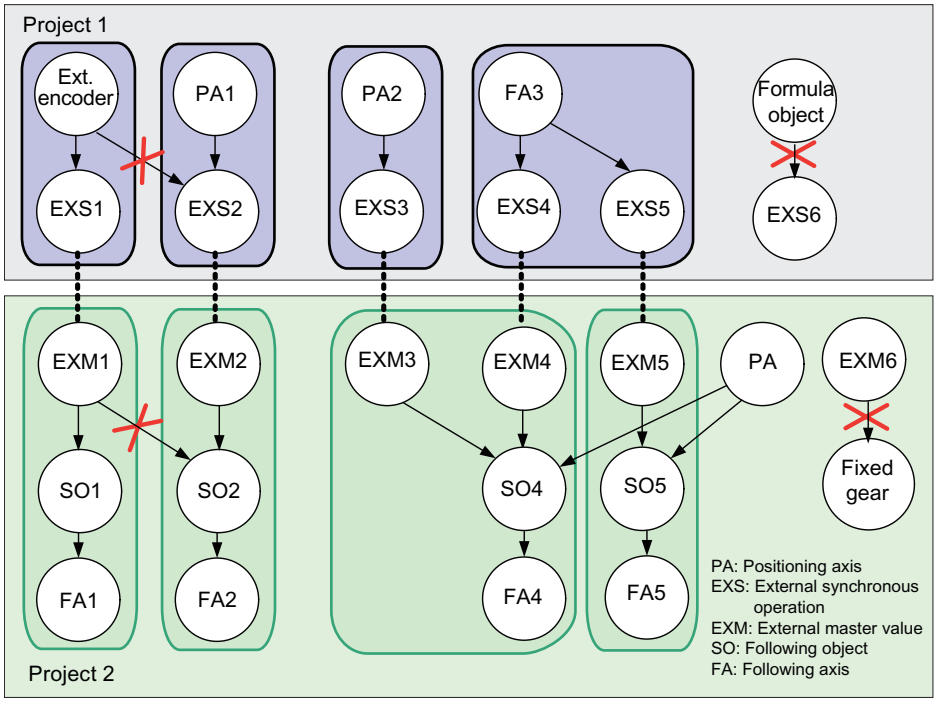


Figure 3-45 Interconnection options for proxy objects

In order to interconnect multiple synchronous objects to one external master value, you must create an additional synchronous operation with a virtual axis, to which the external master value proxy object will be assigned. In this case, the additional virtual axis acts as a master value source for multiple synchronous objects.

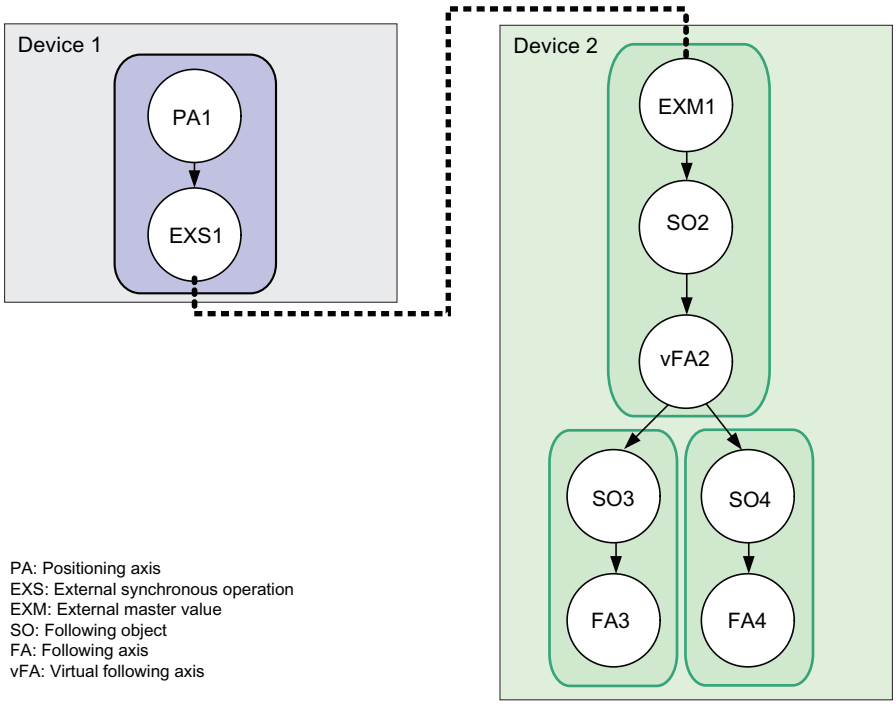


Figure 3-46 Interconnection of multiple synchronous objects to an external master value

The same also applies when different master objects can alternatively be the master value for an external synchronous operation. In this case, an additional synchronous object with a virtual axis must be provided to be interconnected with the master object. The external synchronous operation is assigned to the virtual axis.

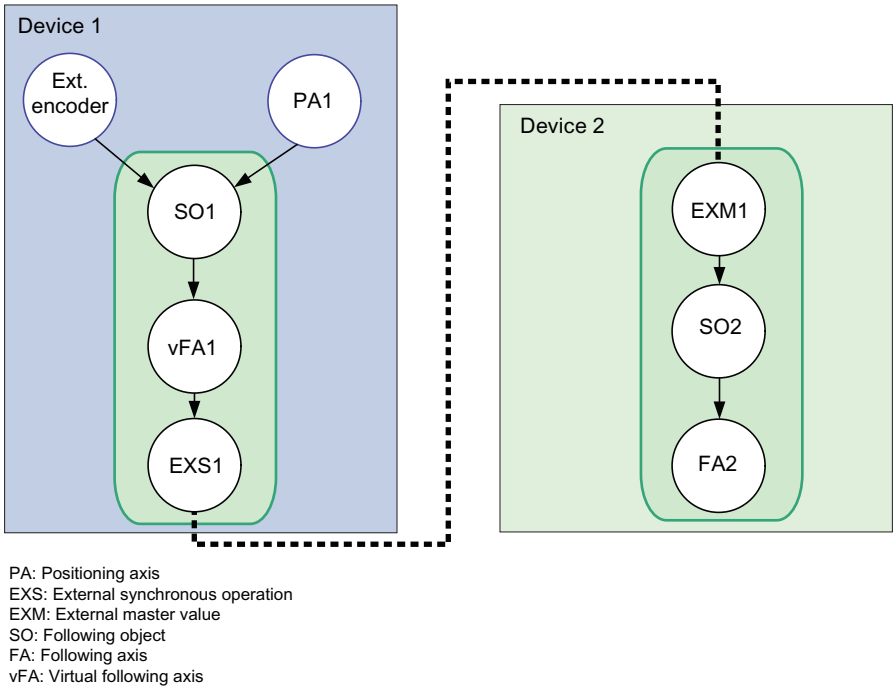


Figure 3-47 Interconnection of multiple master objects to an external synchronous operation

3.5.6 Synchronizing the interface

See Synchronizing the interfaces (Page 175).

3.5.7 Switching over to an external master value source

If more than one master value is assigned to a synchronized axis, the master value source can be selected and switched over on the synchronous object using the **_setMaster** command (see **Fundamentals of synchronous operation**, "Switching over the master value source"). If you want to switch to an external master value source, you must specify the name of the external master value proxy object in the **_setMaster** command.

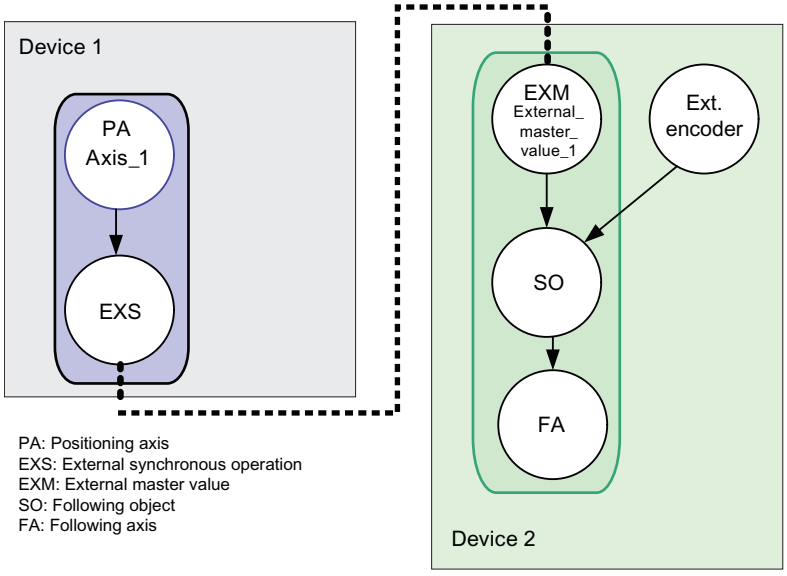


Figure 3-48 Synchronous operation with external and local master value source

This figure shows an example with interconnection to a local and external master value source. In order to switch to the master value of **Axis_1** with the **_setMaster** command, the name of the **Ext_master_value_1** proxy object must be specified in the command.

3.6 Configuration of distributed synchronous operation with different RT versions

3.6.1 Overview of distributed synchronous operation with different RT versions

It is useful to configure a distributed synchronous operation with different RT versions when existing plants/machines are to be extended partly, or functions are required that are not supported in the currently used RT version. This applies for an extension by one or more controllers and for the upgrade of an individual SIMOTION controller in a machine with several SIMOTION controllers.

3.6.2 Configuration of distributed synchronous operation with different RT versions across projects

This is possible as of RT version V4.3. For further information, see Configuring distributed synchronous operation across projects (Page 177).

3.6.3 Configuration of distributed synchronous operation with different RT versions in one project

Prerequisite for the configuration is an engineering system version as of SCOUT V5.1 in conjunction with RT versions as of V4.3.

The following scenarios are considered in more detail:

- Extending an existing project by a SIMOTION controller with a different RT version than that used previously in the project.
- Upgrading a single controller to a different RT version in the project.

Project extension by a SIMOTION controller with a different RT version

To configure an additional controller that has a different RT version for a distributed synchronous operation in an existing project, proceed as follows:

1. Create the new controller with the appropriate RT version in a project.
2. Configure the controller as IRT slave (master axis available in the project) or as IRT master (slave axes available in the project) and integrate it in the PROFINET topology. Further information can be found in Section Distributed Synchronous Operation Configuration (Page 168).
3. Create an external synchronous operation on the master axis. Further information can be found in Section Configuring distributed synchronous operation across projects (Page 177).
4. Create an external master value on the synchronous object on the slave axis.
5. Create the input and output addresses with 24-byte length in the controller-controller direct data exchange. More detailed information can be found in the Communication System Manual, *in Section Configuring direct data exchange (data exchange broadcast) between IO controllers.*
6. Assign input and output addresses on the external synchronous operation (master) and external master value (slave) from item 5. corresponding to the start address in the controller-controller direct data exchange. Note that changes to the addresses in the controller-controller direct data exchange must also be made in the external synchronous operation / external master value.
7. Optionally: Perform the PROFIsafe configuration on the new controller (F-proxy). Note that when using the controller-controller direct data exchange with IRT, the controllers must not be used on the same interfaces as IRT I device. Only RT I device couplings are possible. Further information on F-proxy configuration can be found in the Communication System Manual.

Upgrading a single controller to a different RT version in the project

To upgrade a single controller to a different RT version in the project, proceed as follows:

1. Upgrade the controller in the existing project that requires functions of a later RT version.
2. Remove the existing distributed synchronous operation configuration on the controller.
3. Use the proxy TOs to implement the distributed synchronous operation. If the master axis is on the upgraded CPU, an external synchronous operation must be created. If the slave axis/ axes are on the CPU, create an external master value. Further information can be found in Section Configuring distributed synchronous operation across projects (Page 177).
4. Depending on which axis/axes are on the upgraded CPU, the counterpart of the proxy TO (external synchronous operation / external master value) must be created on the other controllers.
5. Create the input and output addresses with 24-byte length in the controller-controller direct data exchange. More detailed information can be found in the Communication System Manual, *in Section Configuring direct data exchange (data exchange broadcast) between IO controllers*.
6. Assign input and output addresses on the external synchronous operation (master) and external master value (slave) from item 5. corresponding to the start address in the controller-controller direct data exchange. Note that changes to the addresses in the controller-controller direct data exchange must also be made in the external synchronous operation / external master value.
7. Adapt the user programming. Change the master values on the synchronous operation commands on the external master value.

Part III - Synchronous operation across multiple cycles

4

4.1 Overview of multiple cycle synchronous operation

This part describes the function of **synchronous operation across multiple cycles**. A synchronous operation can be configured such that high-priority axes are calculated in the SERVO or IPO processing cycle clock and low-priority axes in the IPO or IPO_2 processing cycle clock (see under processing cycle clock Real and virtual axes Runtime model and processing). It introduces you to the operating principle and technological boundary conditions of synchronous operation with the master object and following axis in different interpolator cycle clocks (IPO, IPO_2, SERVO). You are shown how to create and configure a synchronous operation in different interpolator cycle clocks.

Function overview

The **synchronous operation across multiple cycles** functionality allows you to operate a master value source and a synchronous axis in different IPO cycle clocks (IPO, IPO_2 and SERVO).

Example:

- Axis_1 is the master axis and is assigned to the IPO task.
- Axis_2 is the following axis and is assigned to the IPO_2 task.

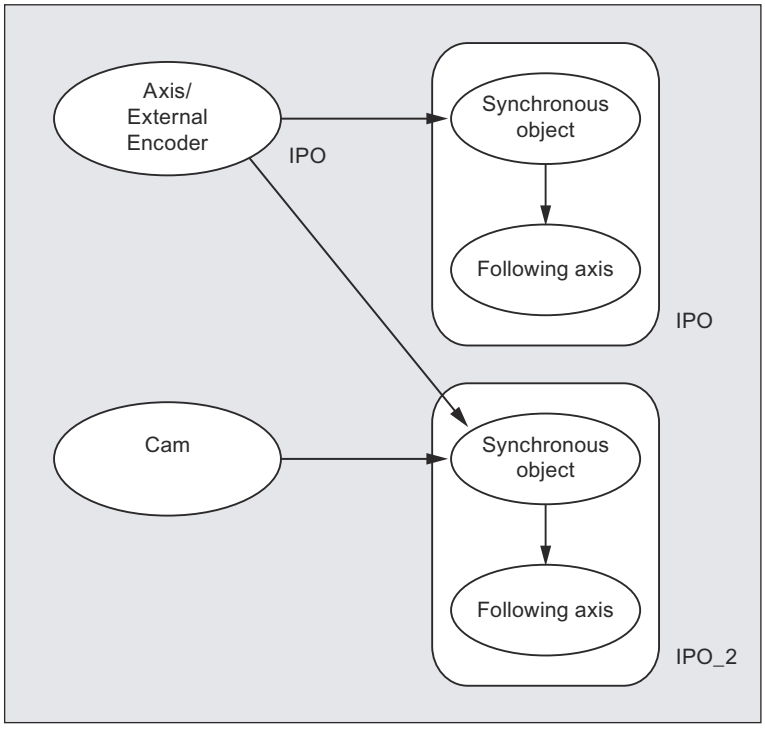


Figure 4-1 Example configuration of synchronous operation between IPO and IPO_2

Application

The **synchronous operation across multiple cycles** function enables the interpolator to be placed in a cyclical system task with a greater cycle time for axes that do not require a high time resolution for calculating the reference value. This reduces the required processor performance.

In this way, it is possible to:

- Operate coupled axes on one device in different IPO cycle clocks
- Operate coupled axes on different devices in different IPO cycle clocks (**synchronous operation across multiple cycles** can be combined with **distributed synchronous operation**).

Operating principle/Compensations

The phase error, which results from processing in a succession of different cycles, can be compensated for in the same way as with distributed synchronous operation:

- Compensation on the master value side by means of setpoint output delay
- Compensation on the slave value side by means of master value extrapolation

For details, see section Running synchronous operation across multiple cycles (Page 220).

4.2 Boundary conditions

Objects with synchronous operation across multiple cycles cannot be created arbitrarily, but rather must satisfy the following rules:

- In a synchronous operation interconnection of multiple axes, more than one transition of axes is allowed in different IPO cycle clocks. Several compensations are thus required. These can be set independently at the transitions. The total compensation is determined by the system. The effective delay and offset are indicated at each axis and following object.

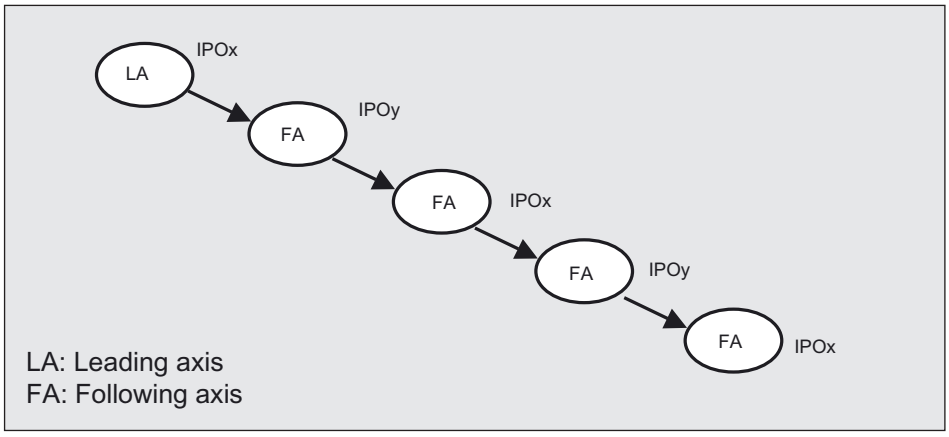


Figure 4-2 Inappropriate configuration with multiple change

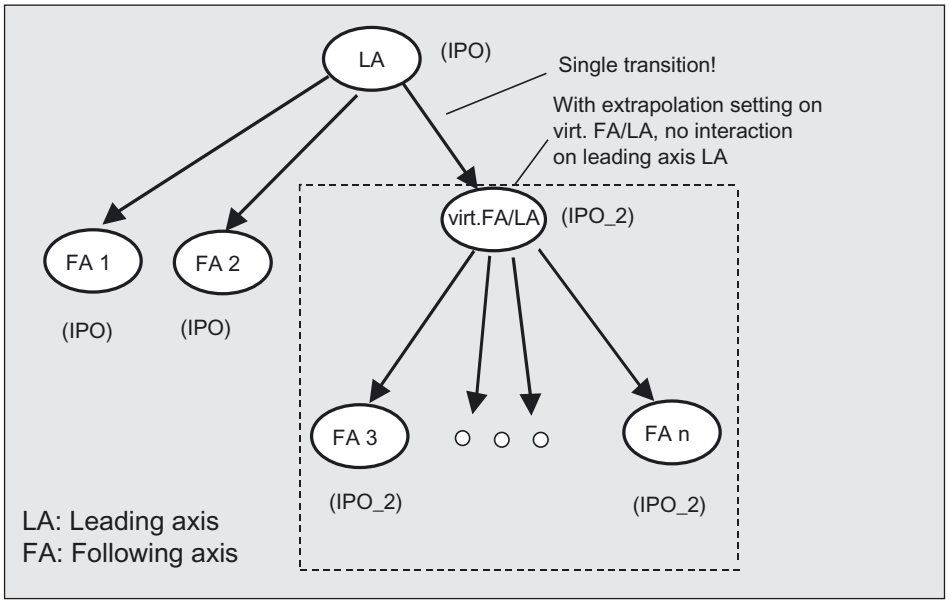


Figure 4-3 Configuration example with axis grouping

Since the calculation of compensation values needs processor resources, it is recommended to keep the number of transitions low and to form axis groups.

- Distributed synchronous operation and synchronous operation across multiple cycles can be combined.
  Synchronous operation across multiple cycles is permitted both in the master system as well as in the slave system.
  For further information, see Rules for the communication/topology for distribution using PROFIBUS. (Page 150)
- A distributed synchronous operation and an IPOx - IPOy transition is only possible between a leading axis (Axis TO or External Encoder TO) and a synchronous object.
  - A Fixed Gear TO, Addition Object TO, Formula Object TO, Closed-loop controller TO cannot be at a transition, neither on the master value side nor on the slave value side.
  - These TOs may only be located in an effective sequence without transitions between distributed synchronous operation or IPO-IPO 2.
- Recursive interconnections which also contain IPOx - IPOy transitions are not prevented.
  Please note, however, that:
  When switching over during motion, the compensations will lead to a jump or a temporarily constant position setpoint, and a loss in velocity as a result. (Although the monitoring functions will remain active.)
  Therefore, the IPOx- IPOy transitions in the recursive synchronous operation interconnection should be switched *during standstill only*.

4.3 Running synchronous operation across multiple cycles

With synchronous operation across multiple cycles, an offset between the master value source and following axis results from the calculation of the synchronous operation in different cycles.

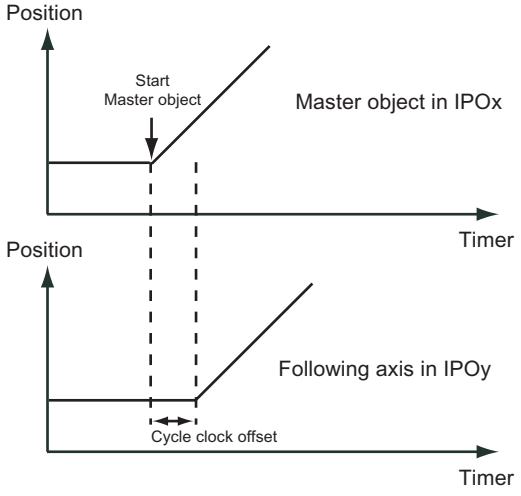


Figure 4-4 Schematic representation of the clock cycle offset due to different interpolator clock cycles

Compensation

The offset can be compensated for in the same way as for distributed synchronous operation:

- Compensation on the master value side by means of setpoint output delay in the IPO cycle clock that provides the master value for the distributed synchronous operation (master CPU).
- Compensation on the slave value side by means of master value extrapolation in the IPO cycle clock containing the slave objects (slave CPU).

The compensations are set and displayed via the system variable **distributedMotion**.

- The output delay is displayed on the leading axis.
- The master value delay is displayed on the synchronous object.
- The cycle clock offset is displayed on the synchronous object.

The details in Compensations for distributed synchronous operation apply accordingly.

Information on compensations

- Compensations can be set differently on different master objects.
- If "Extrapolation" compensation mode is selected, the output delay on the master is zero. In this way, there is no effect on other higher-level master values and their compensation calculation. This allows you to form axis groups.
- It is not possible to specify interpolation on the following axis without master value delay on the leading axis.
- With the "No compensation" setting, the master values are taken over as the exist.
  - Therefore a transition from the slower IPO to the faster IPO is possible but not technically practical.
  - A transition from the faster IPO to the slower IPO is possible, but is associated with a phase offset.

Life-sign monitoring

The life-sign monitoring is also active for synchronous operation across multiple cycles and takes effect, for example, with level overflows.

The details in Operating axes with distributed synchronous operation apply accordingly.

Note

If commands for the master value and the slave value are programmed in the same task, it must be ensured that the commands are effective in the respective processing cycle clock.

If the master value is in a slower execution level than the slave value (e.g. IPO_2 to IPO), it must be noted, for example, that a **_redefinePosition** to the master value for the slave value only becomes effective at the IPO_2 cycle limit. This must be taken into account before issuing an immediately following synchronize command.

4.4 Creating synchronous operation across multiple cycles in SCOUT

This section describes how to create and configure devices and objects with synchronous operation across multiple cycles.

Master object

1. Insert the leading axis or external encoder and configure it.
2. If required, insert cams and configure these.

Following axis

1. Create one or more following axes.
   Make sure that **Synchronous operation** is activated as the technology on the following axes.
2. Configure the synchronous object.
   Interconnect the synchronous object with the master object.

Processing cycle clock

The processing cycle clock is defined in the **executionConfigInfo.executionLevel** configuration data element of the axis and the synchronous object. The setting can also be made on the axis via the configuration mask.

- Set the desired IPO cycle clock for each object.
  When doing so, select the **IPO** or **IPO2** setting for the **Processing cycle clock**.

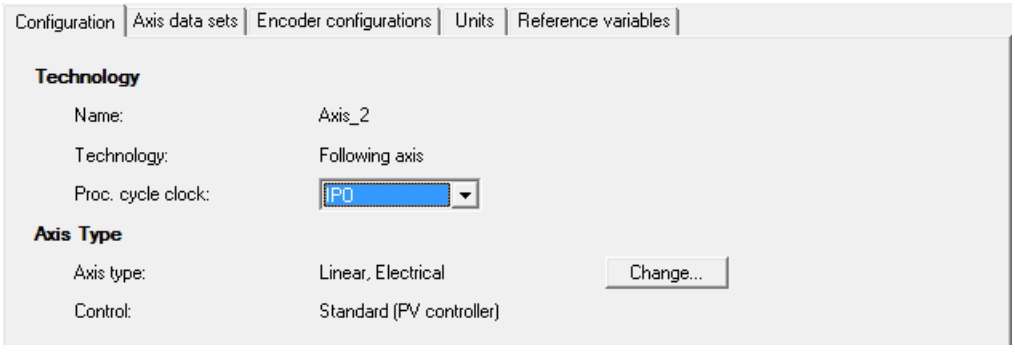


Figure 4-5 Defining the execution cycle in the axis configuration

Configuration

1. In the shortcut menu of the synchronous object, select **Expert > Expert list**; see **Motion Control Basic Functions**, "Expert list."
2. Specify the required configuration (see Operation of synchronous operation across multiple cycles (Page 220) and Compensations of the distributed synchronous operation (Page 157)).

Part IV - Cam

5

5.1 Overview of cam

This part describes the function of the **Cam** technology object. It introduces you to the creation and definition of cams, as well as providing information on supplementary conditions and the operating characteristics of cams.

Function overview

The **Cam technology object** can be used to define a *transmission function* and apply it with other technology objects. A cam describes the dependency of an output variable on an input variable.

- An *input variable* could be the actual position of a master axis, a virtual master value source, or the time.
- An *output variable* could be used as the set position of a following axis, the setpoint profile, or the pressure/force profile.

The TO cam is a *stand-alone* technology object, which can be interconnected with other technology objects.

Application

At present, a **TO cam** can be utilized with the following objects:

- With a **synchronous object** as a transmission function
- With a **TO axis**, e.g.
  - As a velocity, position, or pressure profile
  - As a valve characteristic curve in the hydraulic axis setting

Definition

A **TO cam** describes a function y = f(x) in sections. The sections can be defined by means of *interpolation points* or *segments* (using polynomials). Cams are *dimensionless*. No physical units are used to define them.

Creation and storage

Cams can be created by means of the SIMOTION parameter assignment tool in the engineering system (**CamEdit**) or using the **CamTool** add-on. Cam objects *cannot* be created by the user program.
In order to define a cam in the **user program**, the object must have been created previously in SIMOTION SCOUT.

Cams are stored *by device* and can be assigned to each applicable object of this device. Multi-device cams are not possible.

Exporting/importing cams

With CamEdit, you can export or import the geometry data of the cam.

Scaling and offset

Cams are scalable either in subranges or overall using commands from the user program, even if you have defined the cam using the parameter assignment system. For further information, see "Scaling and offset".

Interpolation

If a curve is defined using segments, gaps in the definition range can be filled by interpolation. If the cam is defined by interpolation points, the characteristic is interpolated. You can select from a variety of interpolation methods. For additional information, see "Interpolation".

Reset

Resetting a cam causes the contents of the cam to be reset. The reset command deletes previously defined interpolation points or segments. The reset command sets the scaling factor to 1 and the offset to 0. If the cam is interpolated, it must be reset before definitions are made in the user program.

Access protection

Only one write action can be performed on the cam at any one time. Any number of read actions can be performed on the cam at any one time. Several write or read actions cannot be performed at the same time.

5.2 Fundamentals of Cam

5.2.1 Definition

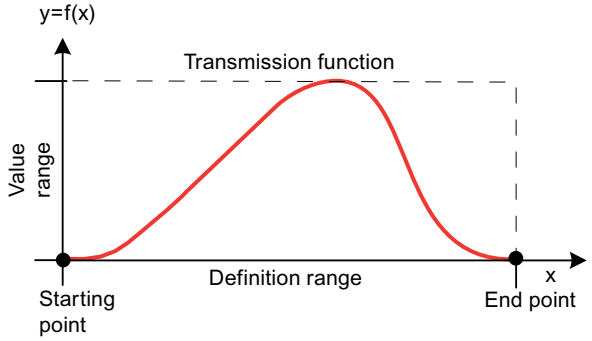


Figure 5-1 Definition of the cam

A cam is defined by the following:

- Definition range
- Starting point and end point of the function in the definition range
- Transmission function
- The value range,
  which is generated from the transmission function

SIMOTION provides the following two options for defining cams. As an option, on the cam in the **interpolatorSettings.dataMode** configuration data element you can set whether only interpolation points, only segments, or both are to be used:

- Definition by means of interpolation points and segments (default).
  Setting **interpolatorSettings.dataMode=SEGMENTS_AND_POINTS**
- Definition by means of segments.
  Setting **interpolatorSettings.dataMode=SEGMENTS_ONLY**
- Definition by means of interpolation points.
  Setting **interpolatorSettings.dataMode=POINTS_ONLY**

A cam can be non-normalized or normalized (with a unit interval of 0.0 ... 1.0) (see Normalization (Page 226)).

Definition based on interpolation points

Interpolation points are represented in the form **P = P(x, y)** in the interpolation point tables. The order by which the value pairs are entered is irrelevant. They are automatically sorted in the definition range in ascending order. SIMOTION interpolates according to the assigned **interpolation type**.

Definition by means of segments

Individual segments are described in accordance with **VDI 2143**, Motion Laws for Cam Mechanisms. For additional information, see Motion laws in accordance with VDI (Page 234).

Polynomials with a maximum polynomial degree of 6 and (as an option) a compound trigonometric function are used for this purpose.

Advantages and disadvantages of the definition modes

Table 5-1 Advantages and disadvantages of defining cams by means of interpolation points or segments

	Definition based on interpolation points	Definition by means of segments
advantages	• Simple definition • Any algorithms can be mapped by interpolation points • Curve creation assisted by teach-in • Simple interface to HMI	• Fewer data used for definition • Standard transitions in accordance with VDI ... • Contour is very precise, transitions are continuous
Disadvantages	• Large number of interpolation points required for the exact representation of the contour	• Complex arithmetic required for calculation of coefficients

Definition by means of interpolation points and segments (mixed)

With V4.2 and higher, on the cam in the **interpolatorSettings.dataMode** configuration data element you have the option of setting whether only interpolation points, only segments, or a mixture of both are to be used.

5.2.2 Normalization

When a cam is defined by means of segments, the individual cam segments can be in normal form (normalized to 1), i.e. both the definition range and value range are contained within the interval [0,1].

Alternatively, the segments can also be entered in the real range.

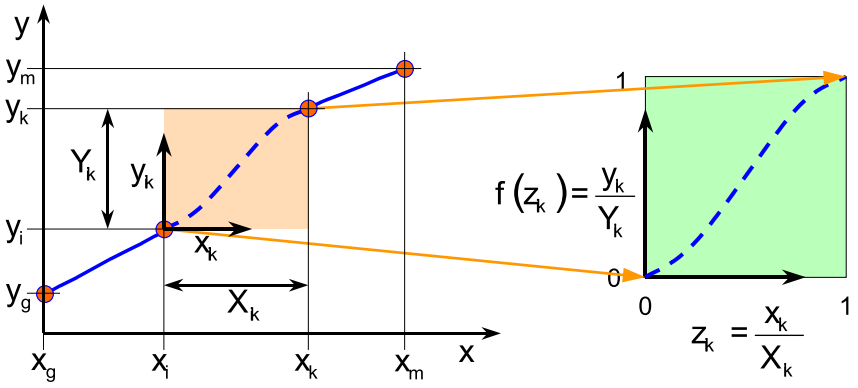


Figure 5-2 Mapping of a real cam segment to the normalized range

A normalization offers the following advantages:

- Motion is clearly defined for similar tasks
- Independent of the real units and value ranges

In addition to the function, the derivatives can also be normalized (normalized transmission function, NTF).

5.2.3 Scaling and offset

The definition range and value range of a cam can be adjusted according to the application, i.e. the function can be offset and stretched/compressed (scaled).

The function value for a scaled and offset cam is generated from the definition using the following formula:

$$y = F_{CAM}\left(\frac{\text{x = Offset of definition range}}{\text{Scaling of definition range}}\right) \bullet \text{Scaling of value range + Offset of value range}$$

Figure 5-3 Formula for scaling and offset of the cam

Scaling

The **_setCamScale** command scales a cam in the value range or definition range. Scaling is applied when the command is successfully issued. Scaling takes place over the complete cam or within a range defined by the starting and end points.

- With *basic scaling*, the entire cam can be scaled and offset.
- With *range scaling*, individual segments of a curve can be scaled and offset.

The zero point of the coordinate axes is used as the scaling point ("pivot point") for basic scaling, whereas the starting point of the specified scaling range is used for range scaling. The starting point of the range scaling can be greater than the end point. In this case, the larger value is the pivot point for scaling (thus the starting point).

The following are possible for the x-axis and y-axis, respectively:

- One complete scaling
- Two range scalings
- One offset

The range scalings can overlap.

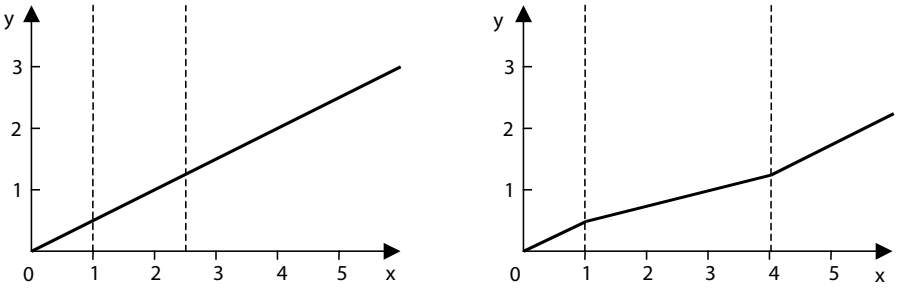


Figure 5-4 Example of scaling of definition range in the range of 1 to 2.5 using a factor of 2

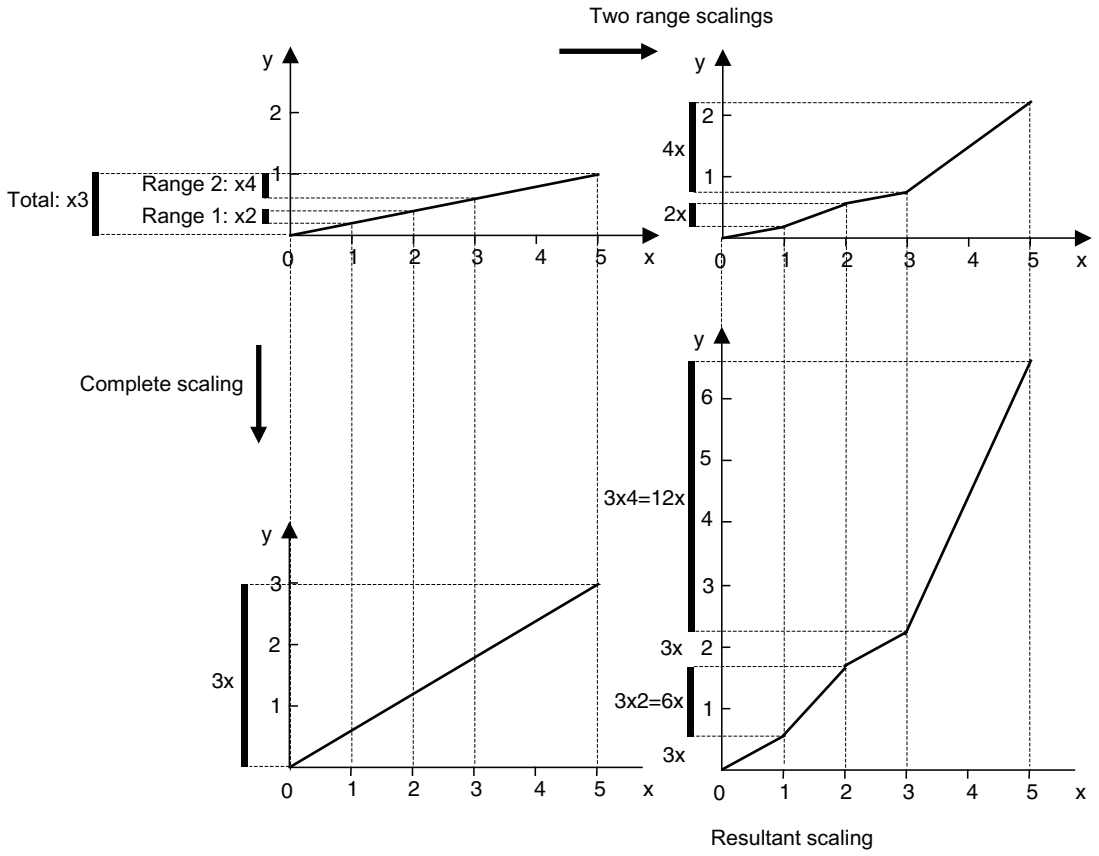


Figure 5-5 Example of two range scalings and a complete scaling in the value range

Scaling can be performed before or after segments and points are inserted or the interpolation is performed. If scaling is performed after interpolation, however, there will be a discontinuity in the first derivative of the cam (even if B-spline or C-spline interpolation is used).

Tip: To prevent this, scaling should begin and end in the dwell ranges.

Offset

The **_setCamOffset** command can be used to separately offset the domain and/or range of a cam.

An offset can be specified as *absolute* or *relative* with respect to the current offset.

- With **ABSOLUTE**, the offset value applies instead of the previous offset value.
- With **RELATIVE**, the offset value is added to the current offset value.

5.2.4 Interpolation

If a curve is defined using segments, gaps in the definition range can be filled by interpolation.

When the cam is defined using **interpolation**, the following checks are performed:

- A plausibility check, i.e., the cam definition is checked (for redundant values within the definition range, for example).
- Missing ranges are added (interpolated).
- Continuity and junction conditions in boundary points are checked.

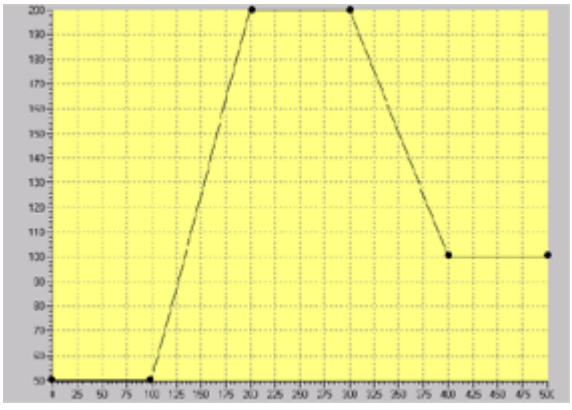
Note

Once interpolation has been performed, new segments or interpolation points can only be inserted after resetting the cam (see Commands for resetting the cam and errors (Page 243)).

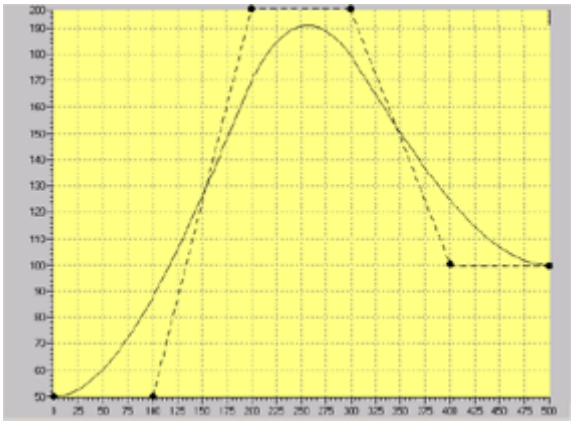
If an attempt is made to insert new segments or interpolation points without first resetting the cam, the attempt is rejected and the return value for the function provides error information. Previously defined interpolation points and segments are deleted when the cam is reset.

Interpolation types

SIMOTION offers the following interpolation types for the **Cam** technology object:

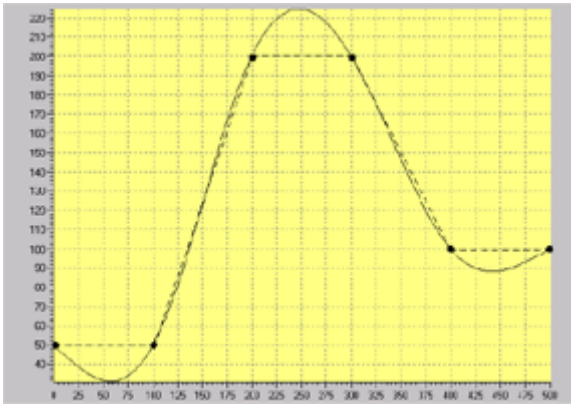
Interpolation	Description	Example
LINEAR	Linear interpolation	

B_SPLINE Approximation using *Bezier splines*, i.e., curve characteristic along the interpolation points



With V4.2 and higher, in the **interpolationSettings.bSplineInterpolation = WITHOUT_APPROXIMATION** configuration data element you have the option of setting whether the 2nd derivation should run continuously.

C_SPLINE Interpolation using *cubic splines*, i.e., curve characteristic through the interpolation points



With V4.2 and higher, in the **interpolationSettings.cSplineInterpolation = ADVANCED** configuration data element you have the option of setting whether the 2nd derivation should run continuously.

Special interpolation settings

- **B spline interpolation**: As an option, you may choose not to set approximation of the B splines in the **interpolationSettings.bSplineInterpolation** configuration data element. This is a good idea if the interpolation points are not at the same distances and at least continuity of the 2nd derivation is required. An accurate display in the ES is not supported in this mode.
- **C spline interpolation**: As an option, you may choose to set a global interpolation in the **interpolationSettings.cSplineInterpolation** configuration data element if problems sometimes arise in the continuity of the 2nd derivation. This results in additional memory and performance requirements.

Continuity check

A function with assigned parameters can be checked for **continuity** in the definition range and value range, and possible points of discontinuity can be corrected. During this process, the points of discontinuity are examined separately for the definition range and value range, and are rated for one of the following corrective actions:

- If the absolute value of the spacing between segments exceeds a maximum value, a correction is made by performing an interpolation between the two segments. This results in insertion of a new segment.

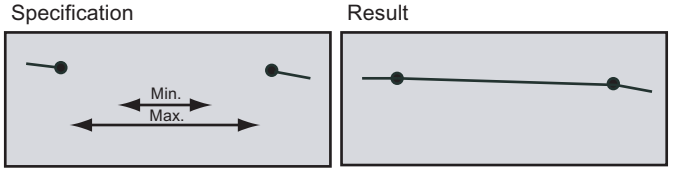


Figure 5-6 Interpolation by insertion a new segment

- If the absolute value of the spacing between segments is greater than the minimum value and less than the maximum value, correction is made by joining the segment end points. The mean value of the spacing of the function is used for the correction. The shape of the segments is affected as a result.

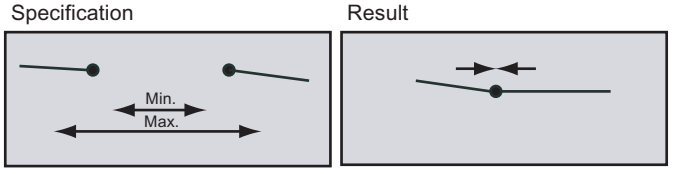


Figure 5-7 Correction by joining segment end points

- If the absolute value of the spacing between segments or interpolation points is less than the minimum value, a correction is not made. The discontinuity point is retained. When this discontinuity point is accessed, the right boundary point is output.

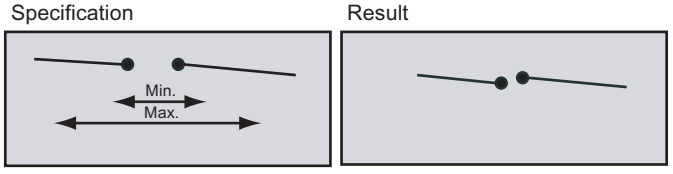


Figure 5-8 Allowing the discontinuity to remain unchanged

The point of discontinuity is corrected according to the evaluation for the definition range and value range.

Table 5-2 Boundary conditions for evaluation in the definition range or value range of the point of discontinuity

Condition	Result
Deviation < minimum	Discontinuity retained
Minimum < deviation < maximum	Join segment end points
Deviation > maximum	Interpolation (new segment)

The correction is controlled (separately for definition range and value range) by specifying the **minimum** and **maximum shape deviation**. This specification can be made on the **_interpolateCam** command for interpolation of the cam.

Depending on the combined evaluation in the definition range and value range, the point of discontinuity is corrected according to the following scheme:

Table 5-3 Combined evaluation for definition range and value range of point of discontinuity

		Correction for definition range		
Discontinuity retained	Join segment end points	Interpolation		
Correction for value range	Retain discontinuity	Retain discontinuity	Join segment end points	New segment
	Join segment end points	Join segment end points	Join segment end points	New segment
	Interpolation	Retain discontinuity	Join segment end points	New segment

- Function continuity can be achieved with **linear interpolation**.
- With **spline interpolation**, continuity is possible in the derivatives.

If the continuity condition cannot be adhered to because of the selected interpolation method or the programmed geometry, a message is provided to that effect.

If an **interpolation boundary point** lies within the programmed geometry, all geometry elements up to the boundary points are rejected. If an interpolation boundary point lies outside the programmed geometry, an end point is extrapolated according to the interpolation method used and taking into account the geometry characteristic.

Continuity at boundary points

When assigning parameters to a cam object, you can make three different settings in the **interpolation** tab.

These settings specify how the runtime system should handle discontinuity at the boundaries of the cam. The appearance of the cam in SIMOTION SCOUT may be different from when it is used later in the runtime system.

- **Non-cyclic**: Not constant at the boundary points
  The runtime system uses the cam as specified, including all discontinuities at the boundaries, even if it is applied cyclically. However, the acceleration limits and moment inertia of the mechanical system / drive are the governing factors.
- **Cyclic absolute**: Position-continuous in the boundary points
  The runtime system converts the cam in such a way that is position-continuous and velocity-continuous at the boundaries during cyclic operation, which can cause changes to occur in the characteristic.
- **Cyclic relative**: Constant velocity in the boundary points
  The runtime system calculates the cam in such a way that is has constant velocity at the boundaries during cyclic operation - within mathematical limits, which can cause changes to occur in the characteristic.

Overlapping segments

In the case of overlapping segments, segment validity can be defined using the options listed below.

- Segments *after* the *segment starting points* are valid
- Segments *up to* the *segment end points* are valid
- Valid segments are determined by the chronological order of insertion.

This behavior is set using the **_resetCam** command.

5.2.5 Inversion

Inverse mapping

For some applications, it is necessary to determine the master value for a defined slave value. The master value can be retrieved using **_getCamLeadingValue**. This inverse mapping is only unique in the case of strictly monotone output functions. In order to supply a master value for output functions that are not strictly monotone, an x-value is specified, and the nearest solution (in both directions) for a y-value is then sought for the specified x-value. If an x-value is not specified, the search will begin from the starting point of the function.

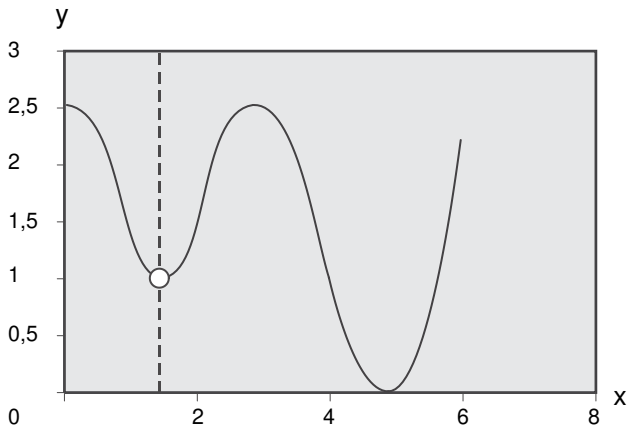


Figure 5-9 Output function

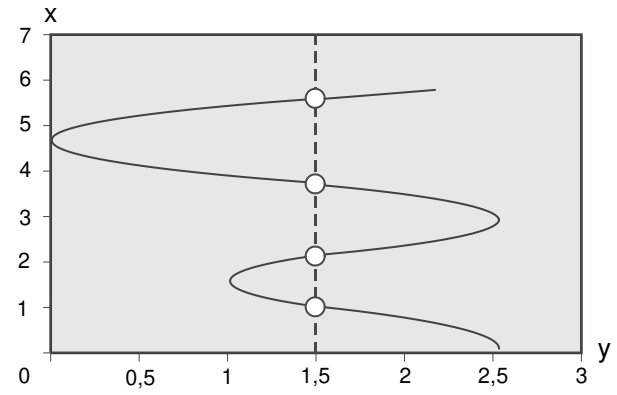


Figure 5-10 Inverse mapping

Inversion during interpolation

The option exists to invert a cam during preparation/interpolation and to store the cam in both the non-inverted and inverted shape (V3.0 and higher). Setting via configuration data **camRepresentation**

Advantage:
Data that has already been inverted can be accessed more quickly when the master value associated with the slave value is read out.

Disadvantage:
More memory is required in the runtime system.

Note

The inverted shape of the cam cannot be represented.

5.2.6 Motion laws in accordance with VDI

The VDI concept of *working ranges and motion transitions* is used to define a cam by means of segments. The **VDI Wizard** can also be used for assistance in the creation of cams.

References

- **VDI Guideline 2143**, 1: Motion Laws for Cam Mechanisms - Basic Theory Düsseldorf: published by VDI-Verlag, 1980
- Volmer, J. (edited.): Mechanism Design - Cam Mechanisms, 2. Ed. Berlin: Published by Technik Verlag, 1989

See also

Motion tasks (Page 235)

Defining a cam for a motion task using segments (Page 237)

5.2.6.1 Motion tasks

The VDI concept differentiates between working ranges and motion transitions.

- *Working ranges* correspond to sequences in a process. The VDI (Association of German Engineers) distinguishes between four different types of working ranges (see below).
- *Motion transitions* are transitions between working ranges, which are not directly relevant to the process but must satisfy certain boundary conditions (such as continuity in velocity and acceleration).

Working ranges in accordance with VDI

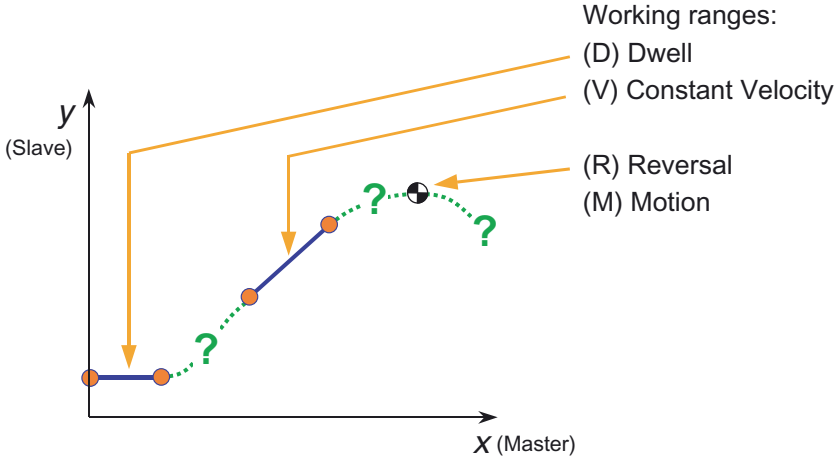


Figure 5-11 Working ranges in accordance with VDI

The VDI concept distinguishes between the following working ranges:

- **R: Dwell** (velocity = 0, acceleration = 0)
- **V: Constant velocity** (velocity <> 0, acceleration = 0)
- **A: Dwell** (velocity = 0, acceleration <> 0)
- **B: Motion** (velocity <> 0, acceleration <> 0)

Example

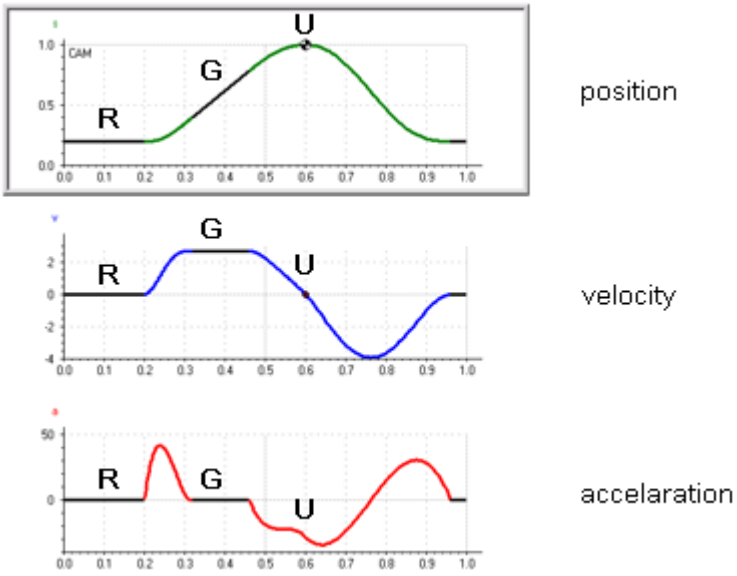


Figure 5-12 Example of a cam with three working ranges

Motion transitions in accordance with VDI

The motion transitions shown in the figure can occur between the individual working ranges.

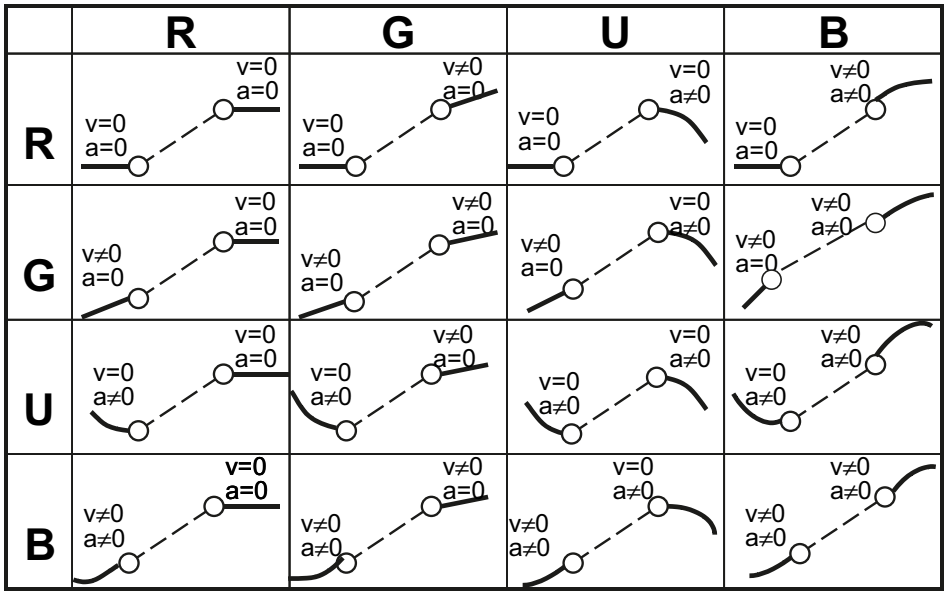


Figure 5-13 Motion transitions in accordance with VDI 2143

5.2.6.2 Defining a cam for a motion task using segments

Definition of working ranges

The **working ranges** of a motion task are usually specified by the process.

Example:

1. A tool waits on a production line for a piece to pass by (dwell).
2. The tool is synchronized to the work piece and performs an action on the work piece (constant velocity).
3. The tool then returns to the waiting position (reversal).

The process starts over from the beginning.

- In order to implement this sequence, the segments of a cam corresponding to the working ranges must first be created.

Creating a motion transition

Now the "only" things left to define are the **motion transitions** that satisfy certain conditions (e.g., jerk-free motion).

- Start by transforming the motion transition to the normalized range.
  For additional information, see Normalization (Page 226).
- The **boundary conditions**, i.e., positions, velocities, and accelerations, must then be taken into account at the boundaries between segments.
- In order to apply a polynomial defined in such a way, it must be transformed back into the **real range**.

5.3 Cam Configuration

Cams can be created with SIMOTION SCOUT or the add-on SIMOTION CamTool add-on. In addition, the curve characteristic can also be defined by a user program during runtime.

You must perform the following tasks when you configure **cams** in SIMOTION SCOUT:

- Create a cam (Page 238)
- Define the cam. (Page 239)
- Exporting/importing cams (Page 238)
- Interconnect the cam.
  Cams are assigned in the respective application. For related information, refer to the descriptions for the corresponding technology objects.

5.3.1 Creating a cam

Follow the steps described below to create a cam:

1. To create a Cam TO in **SCOUT**, double-click **Insert cam** below **CAMS** in the project navigator. You can also copy an existing Cam TO using the clipboard and insert it under another name.
2. Define the cam.

All cams are saved to the **CAMS** folder and are valid for the entire device. The cams can be assigned to all applicable objects of a device (e.g. synchronous objects). This assignment is symbolized in the project navigator, for example, as follows:

- A link to the cam is created below the synchronous object.
- A link to the synchronous object is created below the cam.

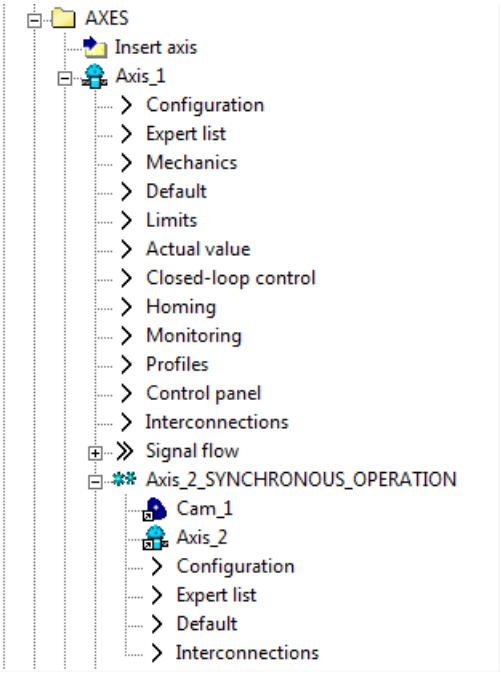


Figure 5-14 Representation of cams in the project navigator

5.3.2 Exporting/importing cams

Description

With CamEdit, you can export or import the geometry data of the cam. For further information, refer to the online help of the "CamEdit cam editor."

5.3.3 Defining and loading cams

Defining with CamEdit

CamEdit can be used to describe cams by means of either interpolation points or segments. These two methods cannot be combined. If the cam is to be created from segments using polynomials, **SIMOTION SCOUT** provides the **VDI Wizard** to assist in creation of the cam.

Defining with CamTool

CamTool is an add-on to the **SIMOTION SCOUT** engineering system providing a graphical option for creation of cams. The add-on is supplied with its own documentation.

Defining by means of program

SIMOTION provides various commands for creation of cams from the application. In ST, cams can be defined by specifying interpolation points, segments, interpolation type, and scaling.

For additional information, see Cam Programming/References (Page 240).

Loading cams

With V4.2 and higher, an active, meshing cam can be loaded with CamEdit or by pressing the **Download** button.

Engaged cams can, therefore, be modified in the engineering system and replaced online. Neither the active camming motions nor active calculation of the valve characteristics are interrupted.

With versions lower than V4.2, it is not possible to download meshing cams to the device.

A cam is only calculated during runtime in the runtime system.

To facilitate diagnostics, the cam can be uploaded again from the runtime system following download so that any changes due to dynamic response adaptations can be detected.

If the cam is modified during runtime and the original cam is going to be downloaded to the runtime system again, the cam must be recompiled in SIMOTION SCOUT. (Otherwise, no change will be detected and the download will be skipped.)

See also

Commands for definition (Page 240)

5.4 Cam Programming/References

Table 5-4 Commands for cam programming

Command	Description
Message functions	Commands for reading out function values (Page 242)
_getCamFollowingValue **_getCamFollowingDerivative** **_getCamFollowingMinMax**	The **_getCamFollowingValue** command returns the cam value, **_getCamFollowingDerivative**, the nth derivation for a specified value in the definition range. The **_getCamFollowingMinMax** command returns (with V4.2 and higher) the minimums and maximums of the cam or of an area of the cam and the associated nth derivation in a specified section of the definition range.
_getCamLeadingValue	**_getCamLeadingValue** returns the value in the definition range for the specified value in the value range
Command tracking	Commands for command tracking (Page 244)
_getStateOfCamCommand	The **_getStateOfCamCommand** command returns a structure with the processing state of a command.
_bufferCamCommandId	With **_bufferCamCommandId**, the command status can be queried after completion or an abort of the command.
_removeBufferedCamCommandId	With **_removeBufferedCamCommandId**, the command should be explicitly removed from the command management of the TO after evaluation is completed.
Geometry	Commands for definition (Page 240)
_addSegmentToCam	The **_addSegmentToCam** command provides you with the option of defining a cam profile in the user program on the basis of polynomial segments f = f(t).
_addPointToCam	The **_addPointToCam** command provides you with the option of defining a cam profile in the user program on the basis of individual interpolation points.
_addPolynomialSegmentToCam	The **_addPolynomialSegmentToCam** command creates a segment f = f(t), consisting of a polynomial with a maximum of 6 degrees.
_interpolateCam	Before a cam is used, it must first be interpolated with the **_interpolateCam** cam.
_setCamScale	The **_setCamScale** command scales a cam in the range or domain. See Scaling and offset (Page 226).
_setCamOffset	The **_setCamOffset** command can be used to separately offset the domain and/or range of a cam. See Scaling and offset (Page 226).
Object and Alarm Handling	Commands for resetting the cam and errors (Page 243)
_getCamErrorNumberState	The **_getCamErrorNumberState** command can be used to fetch the status of an error number.
_resetCam	The **_resetCam** cam resets the cam.
_resetCamError	The **_resetCamError** command provides you with the option of acknowledging a particular error or all pending errors on the cam.

5.4.1 Commands for definition

Interpolation points, segments, or a combination of the two can be used to describe cams.

Commands for the following actions are available for modifying the definition of a cam:

- Adding segments (**_addSegmentToCam**)
- Adding interpolation points (**_addPointToCam**)
- Adding a polynomial segment (**_addPolynomialSegmentToCam**)
- Interpolation (**_interpolateCam**)

Programming a cam

When the cam is created, the order in which the cam is edited plays a role. If the shape of the cam is dependent on parameters and undergoes a change, the cam must be reset before each redefinition using the **_resetCam** command. The cam can also be reset prior to the first calculation without triggering an error message. This command "erases" the cam. This means that the interpolation is cleared and the points and segments are removed. The technology object is retained but is empty.

The interpolation points and/or segments are then placed side by side in the appropriate order. As with CamEdit, the interpolation points can be in any order and will be sorted automatically.

- This is accomplished using the **_addPointToCam** command (addition of an interpolation point), the **_addSegmentToCam** command (addition of a segment) or **_addPolynomialSegmentToCam** (addition of a polynomial segment).
- Use the **_resetCam** command to define cam behavior for overlapping segments/ranges.
- If the shape of the cam is described completely, interpolation is performed using the **_interpolateCam** command.

Subsequent addition of one or more points or segments, or modification of points or segments is not possible. In this case, the cam must be reset and recreated.

Adding segments (_addSegmentToCam)

The **_addSegmentToCam** command provides you with the option of defining a cam profile in the user program on the basis of polynomial segments f = f(t). Individual segments consist of a polynomial with a maximum of 6 degrees and a trigonometric component.

The polynomial parameters, amplitude, period, and phase of a sine function must be entered in standard form. The transformation parameters must be specified in the basic cam representation (without scaling or offset) or in the actual cam representation (with scaling, offset). The definition range values of the cam must always be increasing, i.e. specified in the positive direction.

Adding interpolation points (_addPointToCam)

The **_addPointToCam** command provides you with the option of defining a cam profile in the user program on the basis of individual interpolation points. In so doing, you can choose to specify values either for a scaled and offset range or a range that is not scaled and offset. The definition range values of the cam must always be increasing, i.e. specified in the positive direction.

Adding a polynomial segment (_addPolynomialSegmentToCam)

The **_addPolynomialSegmentToCam** command creates a segment f = f(t), consisting of a polynomial with a maximum of 6 degrees. The polynomial parameters are input in the real range.

Interpolation (_interpolateCam)

Before a cam is used, it must be interpolated. The interpolation defines the connections between points, between segments and between a point and a segment.

For additional information, see Interpolation (Page 228).

5.4.2 Commands for reading out function values

The following commands can be used to read out individual function values from cam characteristics.

- The **_getCamLeadingValue** command supplies the value in the domain (master value) for the specified value in the range (slave value).
  Since this relationship is not always unique, a reference value can be specified. See Inversion (Page 233).
- The **_getCamFollowingValue** command supplies the cam with a specified value in the domain (master value).
- The **_getCamFollowingDerivative** command can be used to obtain the n-th derivative of the function for a specified value in the domain (V4.0 and higher).
  The **derivativeOrder** parameter can be used to select the n-th derivative.
  This allows, for example, the application to replace specific cams and so take account of the velocity, acceleration, etc.

The **leadingPositionMode** parameter can be selected for the commands to specify whether the scaling and offset is to be used (**ACTUAL**) or not (**BASIC**).

Reading out the minimums/maximums of a cam

The **_getCamFollowingMinMax** command returns (with V4.2 and higher) the minimums and maximums of the cam or of an area of the cam and the associated nth derivation in a specified section of the definition range.

The minimums/maximums are also available (in the case of V4.2 and higher) via the **followingRange** system variable.

System variables

The sequence of programmed commands can be read out by means of system variables; see Monitoring the synchronization (Page 59).

The **activeCam** variable indicates the active cam on the synchronous object. The variable is a read-only variable.

The **activeMaster** variable indicates the active master on the synchronous object. The variable is a read-only variable.

5.4.3 Commands for resetting the cam and errors

- The **_resetCam** command has the following effect:
  - The cam is reset to the initial state.
  - Pending errors are deleted.
  - Depending on the command parameters, the geometry and corrections are deleted as applicable (see table below).
  - The system variables are reset according to parameters.
    With versions lower than V4.2, resetting an active cam that is interconnected with a synchronous operation function by means of the **_enableCamming** command always results in an error message that has to be acknowledged. The **_resetCam** command cannot be executed.
  - With V4.2 and higher, the **camModify = WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM** parameter can even be used to reset an active cam. This makes it easy to modify the cam subsequently.
- The **_resetCamError** command provides you with the option of acknowledging a particular error or all pending errors on the cam.
  The command is terminated with a negative acknowledgment if any errors that must not be acknowledged at this point are present.
- Parameter of the **_resetCam** command (for additional information, see TP system functions reference list)
- **insertMode** parameter (optional). Specifies the sequence interpretation with respect to subsequent entry of overlapping segments.
- **userDefaultData** parameter (optional). This parameter controls whether the user default values are reset to the configured values.
- **activateRestart** parameter (optional). Can be used to perform a technology object restart as well.
- **camData** parameter (optional). Specifies how the cam is to be reset.
- **camModify** parameter (optional, V4.2 and higher). This parameter controls the cam status during and after the **_resetCams** in relation to the geometry data.

Table 5-5 Modes of operation for the parameters associated with the **_resetCam** command for modifying a cam via the user program (UP)

	camModify parameter (V4.2 and higher)		
	WITH_INTERPOLATION	**WITHOUT_INTERPOLATION**	**WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM**
camData parameter			
CAM_DATA_RESET	Deletes the currently effective cam. The cam can be reprogrammed. *) (Also applies to < V4.2)	Deletes the currently effective cam. The cam can be reprogrammed. *)	The active cam is retained. The cam can be reprogrammed in the background in parallel. *) **_interpolateCam()** activates the cam programmed in the background.

LOAD_CONFIGURED_DATA	The currently effective cam is loaded and interpolated with the configured interpolation with the data specified by the ES. (Also applies to < V4.2)	The currently effective cam is loaded with the data specified by the ES without interpolation and can be modified via UP before interpolation is then performed. *)	The active cam is retained. The cam is reloaded in parallel in the background without interpolation with the data specified by the ES and can be modified via the UP. *) **_interpolateCam()** activates the cam programmed in the background.
MAINTAIN_PROGRAMMED_DATA (V4.2 and higher)	Regardless of whether it is loaded from the ES or programmed via the UP, the currently effective cam is interpolated with the current interpolation settings.	Regardless of whether it is loaded from the ES or programmed via the UP, the currently effective cam is loaded without interpolation. It can then be modified via the UP and subsequently interpolated. *)	The active cam is retained. Regardless of whether it is loaded from the ES or programmed via the UP, the cam is loaded in parallel in the background without interpolation. It can then be modified via the UP. *) **_interpolateCam()** activates the cam programmed in the background.

*) Alteration of geometry definition, e.g. with **_add**...**ToCam()**

5.4.4 Commands for command tracking

- The **_getStateOfCamCommand** command returns a structure with the processing state of a command.
  - **functionResult** specifies the error code.
  - **commandIdState** returns the current state of the cam.
  - **abortId** specifies the command abort reason. The abort reason is specified for the alarm 30002 "Command aborted (reason: <abortId>, command type ...)".
    See also **_getMotionStateOfAxisCommand** for the **Axis technology object**.
- With **_bufferCamCommandId**, the command status can be queried after completion or an abort of the command.
- With **_removeBufferedCamCommandId** , the command should be explicitly removed from the command management of the technology object after evaluation is complete.

The number of motion commands that the MotionBuffer can accept can be specified using the **camType.DecodingConfigInfo.numberOfMaxbufferedCommandId** configuration data.

Further information is available in the SIMOTION reference lists.

5.4.5 Programming and sequence model

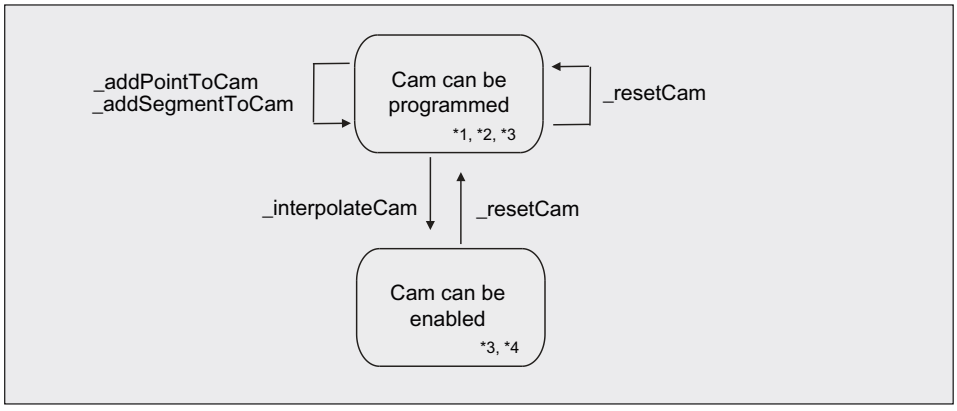


Figure 5-15 Programming and sequence model for a cam technology object

The following commands and functions are active in the particular technology object states:

*1 **_setCamScale**
_setCamOffset

*2 **_addPointToCam**
_addSegmentToCam

*3 **_getCamFollowingValue**, **_getCamFollowingDerivative**
_getCamLeadingValue

*4 **Error**
In general, illegal parameters and commands do not affect the states of the technology object, but they must be acknowledged using **_resetCamError**.

If commands that are not permitted in the technology object state are transmitted to the **Cam** technology object, an error message is triggered, which requires an acknowledgment. The technology object state, such as **programmable** or **cannot be activated**, is retained.

5.4.6 Local alarm response

Local **alarm responses** are set by means of the system.

The following responses are possible:

- **NONE**
  No response
- **DECODE_STOP**
  Command preparation aborted

Following a **_resetCam** or **_resetCamError**, editing of the Cam technology object can be resumed.

5.5 Graphic output

5.5.1 Reading out cams with CamEdit

Cams can be read out from the controller and displayed graphically using CamEdit.

A cam is only calculated during runtime in the runtime system. If it is read out from the controller, the calculated cam can be displayed graphically in CamEdit. If the cam is modified during runtime and the original cam is going to be downloaded to the runtime system again, the cam must be recompiled in SIMOTION SCOUT. (Otherwise, no change will be detected and the download will be skipped.)

By default, the cam is graphically displayed in the standard form in CamEdit. If scaling and/or offsetting of the cam is required, display in scaled form must be explicitly enabled.

If the command **_resetCam** with the parameter **camModify = WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM** has been output in the user program, the active cam disk will be retained. On read-out, the data of the cam disk loaded in parallel in the background and modified via user program are fetched.

Index

A

Absolute camming, 27
Absolute gearing, 21
Actual value, 35
Actual value coupling, 35, 37
    With extrapolation, 35
    with tolerance window, 39
Actual value error, 76
Adapting the synchronization velocity, 107
Axis
    Removing from the synchronous operation interconnection, 106

B

Basic scaling, 226
Basic synchronous operation, 71
Bezier splines, 230
Boundary conditions
    Synchronous operation IPO - IPO_2, 219

C

Cam, 15
    Application, 223
    Assigning, 112
    Configuring, 237
    Creating, 238
    Cyclic application, 31
    Defining with segments, 237
    Definition, 223, 225
    Direction, 32
    Interpolation, 228
    Inversion, 233
    Normalization, 226
    Opposite direction, 32
    Overview, 223
    Programming model, 245
    Same direction, 32
    Scaling and offset, 226
    Self-terminating, 30
Cam coupling, 18
Cam synchronization
    assigning parameters/defaults, 124
Camming, 14, 26
    Absolute, 27
    assigning parameters/defaults, 118
    Cam synchronization, 124
    Cyclic, 30
    Non-cyclic, 30
    Offset, 32
    Scaling, 32
Cascading
    In distributed synchronous operation, 153
Combination
    Synchronous operation compensation, 163
Command buffer
    Synchronous operation, 139
Command execution
    Synchronous operation, 138
Command processing
    In the IPO cycle clock, 140
    Synchronous operation, 143
Command tracking
    Synchronous operation, 136
Command transition conditions
    Synchronous operation, 141
Commands
    Cam, 240, 242, 243, 244
    Resetting errors, 138
    Resetting states, 138
    Synchronous operation, 135
Compensation, 150
    Combination, 163
    Distributed synchronous operation, 157
    Master value side, 160
    On the slave value side, 162
    Superimposed distributed synchronous operation, 74
    Synchronous operation IPO - IPO_2, 220
Compensation on the master value side, 160
Compensation on the slave value side, 162
    Calculating, 163
Configuring
    Cam, 237
    Distributed synchronous operation, 168, 169
    Synchronous operation, 110
    Synchronous operation IPO - IPO_2, 222
Constant velocity, 235
Continuity at boundary points
    Cam, 232
Continuity check, 231
Coupling
    Cam, 18

Cross-project distributed synchronous operation
Overview, 177
Cubic splines, 230
Cycle clock offset, 150
Calculating, 161, 163, 164
Distributed synchronous operation, 157
Cycle clock scaling
Distributed synchronous operation, 155
Cyclic
Camming, 30
Cyclic cam application, 31

D

Default setting
Cam synchronization, 124
Camming, 118
Dynamic response, 127
Gear synchronization, 120
Gearing, 116
Master dynamic response, 129
Velocity gearing, 117
Definition
Commands for cam, 241
Delay time
Synchronous operation, 162
Desynchronization, 15, 64
Desynchronization position, 65
Master value distance, 65
Synchronous operation, 64, 123, 126
Via dynamic response parameters, 65
Desynchronization position
Synchronous operation, 65
Direction
Gearing, 23
Distributed synchronous operation
Cascading, 153
Compensation, 157
Configuring, 169
Master-slave relationship, 151
Operating states, 166
Overview, 149, 168
Rules for topology, 150
Synchronization interfaces, 175
With cycle clock scaling, 155
DP master
Creating, 187
DP slave
Creating, 187
Dummy master
Creating, 183
Dwell, 235
Dynamic response
assigning parameters/defaults, 127
Synchronous operation influence, 67
Dynamic response parameters
Desynchronization, 65
Synchronization, 52, 53

E

Error message
Synchronous operation, 174
Error reaction
For cam, 245
Synchronous operation, 143
Example
Cross-project distributed synchronous operation, 187, 191, 199
Programming, 98, 101
External master value
Creating, 205
External master value source
Switching, 213

F

Filtering
for actual value coupling, 37
Following axis, 15

G

Gear ratio, 20
Gearing, 23
Velocity gearing, 25
Gear synchronization
assigning parameters/defaults, 120
Gearing, 14, 20
Absolute, 21
assigning parameters/defaults, 116
Direction, 23
Relative, 22

I

I device
Creating, 191, 199
In the opposite direction
Synchronous operation, 32
In the same direction
Synchronous operation, 32

Interconnection, 18
Interfaces
Synchronization, 175
Interpolation
Cam, 228
Interpolation point table, 225
Interpolation points, 225
Interpolation type, 225
Interpolation types, 229
Inverse mapping, 233
Inversion
Cam, 233
IO controller
Creating, 191, 199
IPO - IPO_2
Synchronous operation, 217
IPO cycle clock
Command processing, 140

L

Life-sign
Monitoring, 164
Life-sign monitoring
Synchronous operation IPO - IPO_2, 220
Linear interpolation, 232
Local alarm response
during synchronous operation, 245
For cam, 245

M

Master, 15
Master dynamic response
assigning parameters/defaults, 129
Master object, 15
Creating, 205
Master value, 16
Assigning, 112
Creating, 205
Switching, 213
Master value distance
Synchronization, 49
Master value position, 43, 44, 45
Master value source
Switching, 68
Master value switchover
Synchronous operation, 69, 70
With dynamic response, 69
Without dynamic response, 69
Master-slave relationship
In distributed synchronous operation, 151
Monitoring
Life-sign, 164
Synchronous operation, 132
Motion, 235
Motion laws in accordance with VDI, 234
Motion tasks
In accordance with VDI, 235
Motion transition
Creating, 237
Motion transitions
In accordance with VDI, 235, 236

N

Non-cyclic
Camming, 30
Normalization
Cam, 226
Normalized transmission function, 226

O

Offset
Cam, 228
Camming, 32
Changing, 23, 33
Distributed synchronous operation, 157
Effectiveness, 33
Superimposing, 24
Synchronous operation, 21
Operating states
In distributed synchronous operation, 166
Overlapping segments
Cam, 233
Overshoot factor
Synchronization, 51
Overview
Cam, 223

P

Parallel effective commands, 139
Polynomials, 223, 225
Position reference
Synchronizing synchronous operation, 122
Processing cycle clock
Synchronous object, 17
Programming model
Cam, 245

Protective doors
Opening and closing, 106
Proxy
Cross-project distributed synchronous operation, 205
Types, 205
Proxy object
Creating, 205
Interconnecting, 211

R

Range scaling, 226
Reading out function values
Synchronous operation, 135
Readout of function values
Commands for cam, 242
Recursive synchronous operation interconnection, 17
References, 3
Relative gearing, 22
Resetting of states and errors
Commands for cam, 243
Reversal, 235
Reversing function, 233
Rules
Synchronous operation interconnection, 18

S

Scaling
Cam, 226
Camming, 32
Changing, 33
Effectiveness, 33
Segments, 225
Setpoint, 35
Setpoint value coupling, 35
Settings
on the synchronous object, 131
Shape deviation, 231
Simulation mode
Synchronous operation, 78
Slave, 15
Spline interpolation, 232
Standstill
of the master value, 49
Status
of synchronization, 60
Synchronization, 63
Substituting
Synchronous operation, 66
Superimposed synchronous operation, 19, 71
Superimposition
Offset, 24
Switching
Master value source, 68
To external master value source, 213
Synchronization, 15, 49
Cam, 125
Camming cycle, 46
Display, 62
Following axis position, 46
Interfaces, 175
Master value position, 43, 44, 45
Overshoot, 51
Status, 60, 63
Synchronization direction, 47
Synchronization profile, 50
Synchronization profile type, 49
Synchronization range, 48
Synchronous operation, 40, 125
via dynamic response parameters, 53
Via dynamic response parameters, 52
Via master value distance, 49
Synchronization criterion, 42
Synchronization direction, 47
Synchronization profile, 50
Synchronized group, 15
Synchronous axis, 15
Creating, 111
Synchronous commands, 139
Synchronous object, 15
Settings, 131
Synchronous operation, 13
Adapting the synchronization velocity, 107
Assigning, 112
assigning parameters/defaults, 114
Camming, 26
Command buffer, 139
Command execution, 138
Command processing, 143
Command tracking, 136
Command transition conditions, 141
Commands, 138
Configuring, 110
Context menu, 147
Desynchronization, 65, 123, 126
Desynchronization position, 65
Dynamic response, 127
Dynamic response influence, 67
Master dynamic response, 129
Master value switchover, 69
MCC commands, 135

Menu, 146
Monitoring, 75, 132
PLCopen commands, 135
Reading out function values, 135
Recursive, 17
Simulation mode, 78
ST commands, 135
Substituting, 66
Superimposed, 71
Synchronizing the position reference, 122
Synchronous operation across multiple cycles, 217
Synchronous operation configuration
Specifying, 112
Synchronous operation IPO - IPO_2
Boundary conditions, 219
Compensations, 220
Configuring, 222
Life-sign monitoring, 220
Synchronous operation relationship, 15

T

Tolerance window
for actual value coupling, 39
Topology
Distributed synchronous operation via PROFIBUS, 150

V

VDI
Motion laws, 234
VDI Guideline 2143, 234
Velocity gearing, 14, 25
assigning parameters/defaults, 117

W

Working ranges
In accordance with VDI, 235
Specifying, 237