YASKAWA

YASKAWA Sigma-7 Library

YMC-LIB_PN | YMC-LIB_Sigma7-PN V2.0 | Manual

HB00 | YMC-LIB_PN | YMC-LIB_Sigma7-PN V2.0 | en | 19-40

Block library - Sigma-7 - YASKAWA Motion Control PROFINET

YASKAWA Europe GmbH
Hauptstr. 185
65760 Eschborn
Tel.: +49 6196 569-300
Fax: +49 6196 569-398
Email: info@yaskawa.eu.com
Internet: www.yaskawa.eu.com

Table of contents

1 General

1.1 Copyright © YASKAWA Europe GmbH

All Rights Reserved

For permission to reproduce or distribute, please contact: YASKAWA Europe GmbH, European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany

Tel.: +49 6196 569 300

Fax.: +49 6196 569 398

Email: info@yaskawa.eu.com

Internet: www.yaskawa.eu.com

Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. Nevertheless, the authors retain the right to modify the information.

This customer document describes all the hardware units and functions known at the present time. Descriptions may be included for units which are not present at the customer site. The exact scope of delivery is described in the respective purchase contract.

EC conformity declaration

Hereby, YASKAWA Europe GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions. Conformity is indicated by the CE marking affixed to the product.

Conformity Information

For more information regarding CE marking and Declaration of Conformity (DoC), please contact your local representative of YASKAWA Europe GmbH.

Trademarks

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S and Commander Compact are registered trademarks of YASKAWA Europe GmbH.

SPEED7 is a registered trademark of YASKAWA Europe GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300, S7-400 and S7-1500 are registered trademarks of Siemens AG.

Microsoft and Windows are registered trademarks of Microsoft Inc., USA.

Portable Document Format (PDF) and Postscript are registered trademarks of Adobe Systems, Inc.

All other trademarks, logos and service or product marks specified herein are owned by their respective companies.

Document support

Contact your local representative of YASKAWA Europe GmbH if you have errors or questions regarding the content of this document. If such a location is not available, you can reach YASKAWA Europe GmbH via the following contact:

YASKAWA Europe GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Fax: +49 9132 744 29 1204

Email: Documentation.HER@yaskawa.eu.com

Technical support

Contact your local representative of YASKAWA Europe GmbH if you encounter problems or have questions regarding the product. If such a location is not available, you can reach the YASKAWA customer service via the following contact:

YASKAWA Europe GmbH,
European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany

Tel.: +49 6196 569 500 (hotline)

Email: support@yaskawa.eu.com

1.2 About this manual

Objective and contents

The manual describes the block library *'Sigma-7 - YASKAWA Motion Control PROFINET'* from YASKAWA:

- It contains a description of the structure, project implementation and usage in several programming systems.
- The manual is targeted at users who have a background in automation technology.
- The manual is available in electronic form as PDF file. This requires Adobe Acrobat Reader.
- The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.
- The following guides are available in the manual:
  - An overall table of contents at the beginning of the manual
  - References with pages numbers

Icons Headings

Important passages in the text are highlighted by following icons and headings:

⚠ **DANGER!**
Immediate or likely danger. Personal injury is possible.

⚠ **CAUTION!**
Damages to property is likely if these warnings are not heeded.

ℹ *Supplementary information and useful tips.*

2 Include library

Block library

The block library can be found in the *'Service/Support'* area on the corresponding Web page for download. The library is available as packed zip files. As soon as you want to use these blocks you have to import them into your project.

The following Web pages give you access to the libraries:

- *https://www.yaskawa.eu.com/en/service/drives-motion-software-download*
- *http://www.vipa.com/en/service-support/downloads/yaskawa-lib*

Please always use the manual associated with your library. As long as there are no description-relevant changes, the version information in the manual can differ from those of the library and its files.

The following block libraries are available

File	Description
YMC-LIB_Sigma7-PN_S7_V0001.zip	■ Block library for Siemens SIMATIC Manager. ■ For use in VIPA CPUs or S7-300 CPUs from Siemens.
YMC-LIB_Sigma7-PN_TIA_V0001.zip	■ Block library for Siemens TIA Portal V15. ■ For use in VIPA CPUs or S7-300 CPUs from Siemens.

2.1 Integration into Siemens SIMATIC Manager

Overview

The integration into the Siemens SIMATIC Manager requires the following steps:

1. Load ZIP file
2. "Retrieve" the library
3. Open library and transfer blocks into the project

Load ZIP file

- Navigate on the web page to the desired ZIP file, load and store it in your work directory.

Retrieve library

1. Start the Siemens SIMATIC Manager with your project.
2. Open the dialog window for ZIP file selection via *'File ➔ Retrieve'*.
3. Select the according ZIP file and click at [Open].
4. Select a destination folder where the blocks are to be stored.
5. Start the extraction with [OK].

Open library and transfer blocks into the project

1. Open the library after the extraction.
2. Open your project and copy the necessary blocks from the library into the directory "blocks" of your project.

   ⇨ Now you have access to the blocks via your user application.

> *Are FCs used instead of SFCs, so they are supported by the System 300S VIPA CPUs starting from firmware 3.6.0.*

2.2 Integration into Siemens TIA Portal

Overview

The integration into the Siemens TIA Portal requires the following steps:

1. Load ZIP file
2. Unzip the Zip file
3. "Retrieve" the library
4. Open library and transfer blocks into the project

Load ZIP file

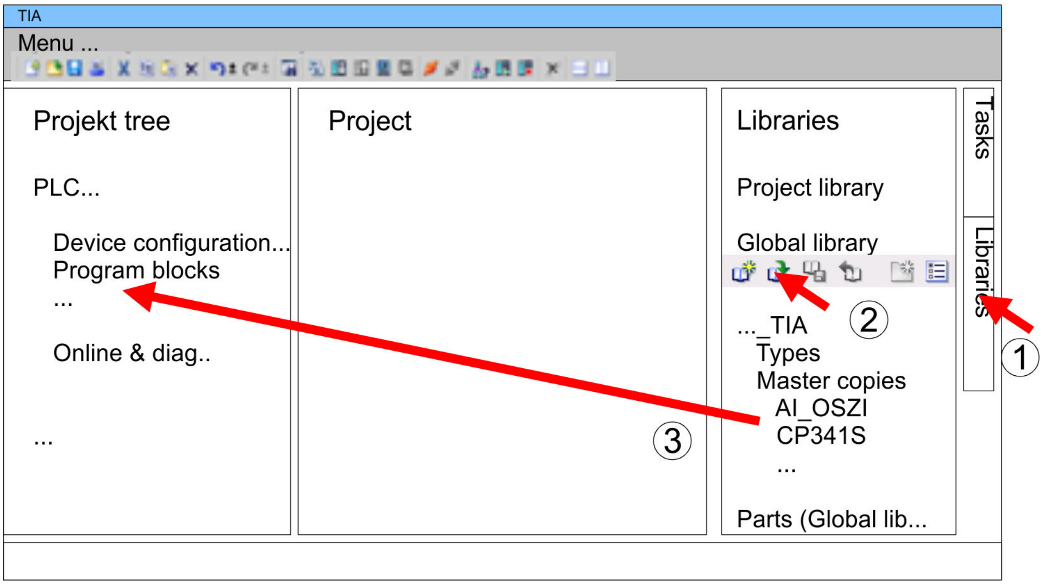
1. Navigate on the web page to the ZIP file, that matches your version of the program.
2. Load and store it in your work directory.

Unzip the Zip file

- Unzip the zip file to a work directory of the Siemens TIA Portal with your unzip application.

Open library and transfer blocks into the project

1. Start the Siemens TIA Portal with your project.
2. Switch to the *Project view*.
3. Choose "Libraries" from the task cards on the right side.
4. Click at "Global libraries".
5. Click at "Open global libraries".
6. Navigate to your work directory and load the file ..._TIA.al1x.



7. Copy the necessary blocks from the library into the "Program blocks" of the *Project tree* of your project. Now you have access to the blocks via your user application.

3 Usage *Sigma-7* PROFINET

3.1 Overview

Precondition

- Siemens SIMATIC Manager from V 5.5, SP2 & *YMC-LIB_Sigma7-PN Library*
  or
  Siemens TIA Portal V 15.1 & *YMC-LIB_Sigma7-PN Library*.
- CPU with PROFINET IO controller such as CPU 017-CEFPR00 with expansion of the working memory to 2MB.
- *Sigma-7* drive with PROFINET connection.

Steps of configuration

1. Setting parameters on the drive ☞ *10*
   - The setting of the parameters happens by means of the software tool *SigmaWin+* respectively FB 834 - Y_WriteParameter ☞ *78*.
2. Hardware configuration in the Siemens SIMATIC Manager ☞ *14*

   or

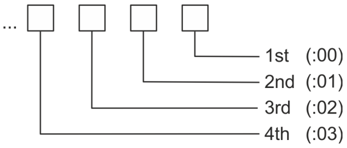
   Hardware configuration in the Siemens TIA Portal ☞ *24*
   - Configuring a CPU with PROFINET IO controller functionality.
   - Configuring a *Sigma-7* PROFINET drive.
   - Configuring a PROFINET connection via the hardware configuration.
3. Programming in the Siemens SIMATIC Manager ☞ *20*

   or

   Programming in the Siemens TIA Portal ☞ *32*
   - Connecting the *Init* block for the configuration of the axis.
   - Connecting the *Kernel* block for communication with the axis.
   - Connecting the blocks for motion sequences.

3.2 Set the parameters on the drive

Parameter digits



⚠ **CAUTION!**
Before the commissioning, you have to adapt your drive to your application! More may be found in the manual of your drive.

The following parameters are to be set by means *SigmaWin+* respectively FB 834 - Y_WriteParameter ☞ *78*:

Sigma-7

SERVOPACK Parameter	Parameter.Digit	Description	Default value
Pn001	Pn001.0	Servo OFF or alarm group 1 stopping method ■ 0: Dynamic break – stops the motor by applying the dynamic brake ■ 1: Dynamic break and release – coast the motor to a stop without the dynamic brake ■ 2: No dynamic break	0
Pn002	Pn002.2	Encoder Usage ■ 0: Use the encoder according to the encoder specifications. ■ 1: Use the encoder as an incremental encoder (currently not supported). ■ 2: Use the encoder as a single-turn absolute encoder (currently not supported).	0
Pn00B	Pn00B.2	Power input selection for three-phase SERVOPACK (Sigma-7 200V) ■ 0: Use a 3-phase power supply input ■ 1: Use a 3-phase power supply input as single-phase power supply input This parameter is relevant for SGD7S-xxxAC0 SERVOPACK only. Do not change in case of SGD7S-xxxDC0 SERVOPACK is used.	0
Pn50A	Pn50A.3	P-OT (forward drive prohibit) signal allocation ■ 0: Enable forward drive when CN1...13 input signal ON (closed) ■ 1: Enable forward drive when CN1...7 input signal ON (closed) ■ 2: Enable forward drive when CN1...8 input signal ON (closed) ■ 3: Enable forward drive when CN1...9 input signal ON (closed) ■ 4: Enable forward drive when CN1...10 input signal ON (closed) ■ 5: Enable forward drive when CN1...11 input signal ON (closed) ■ 6: Enable forward drive when CN1...12 input signal ON (closed) ■ 7: Set the signal to always prohibit forward drive ■ 8: Set the signal to always enable forward drive. ■ 9: Enable forward drive when CN1...13 input signal OFF (open) ■ A: Enable forward drive when CN1...7 input signal OFF (open) ■ B: Enable forward drive when CN1...8 input signal OFF (open) ■ C: Enable forward drive when CN1...9 input signal OFF (open) ■ D: Enable forward drive when CN1...10 input signal OFF (open) ■ E: Enable forward drive when CN1...11 input signal OFF (open) ■ F: Enable forward drive when CN1...12 input signal OFF (open)	1

SERVOPACK Parameter	Parameter.Digit	Description	Default value
Pn50B	Pn50B.0	N-OT (reverse drive prohibit) signal allocation ■ 0: Enable reverse drive when CN1...13 input signal ON (closed) ■ 1: Enable reverse drive when CN1...7 input signal ON (closed) ■ 2: Enable reverse drive when CN1...8 input signal ON (closed) ■ 3: Enable reverse drive when CN1...9 input signal ON (closed) ■ 4: Enable reverse drive when CN1...10 input signal ON (closed) ■ 5: Enable reverse drive when CN1...11 input signal ON (closed) ■ 6: Enable reverse drive when CN1...12 input signal ON (closed) ■ 7: Set the signal to always prohibit reverse drive ■ 8: Set the signal to always prohibit reverse drive ■ 9: Enable reverse drive when CN1...13 input signal OFF (open) ■ A: Enable reverse drive when CN1...7 input signal OFF (open) ■ B: Enable reverse drive when CN1...8 input signal OFF (open) ■ C: Enable reverse drive when CN1...9 input signal OFF (open) ■ D: Enable reverse drive when CN1...10 input signal OFF (open) ■ E: Enable reverse drive when CN1...11 input signal OFF (open) ■ F: Enable reverse drive when CN1...12 input signal OFF (open)	2
Pn511	Pn511.0	DEC (home switch input) signal allocation ■ 0: Active when CN1...13 input signal ON (closed) ■ 1: Active when CN1...7 input signal ON (closed) ■ 2: Active when CN1...8 input signal ON (closed) ■ 3: Active when CN1...9 input signal ON (closed) ■ 4: Active when CN1...10 input signal ON (closed) ■ 5: Active when CN1...11 input signal ON (closed) ■ 6: Active when CN1...12 input signal ON (closed) ■ 7: The signal is always active ■ 8: The signal is always inactive ■ 9: Active when CN1...13 input signal OFF (open) ■ A: Active when CN1...7 input signal OFF (open) ■ B: Active when CN1...8 input signal OFF (open) ■ C: Active when CN1...9 input signal OFF (open) ■ D: Active when CN1...10 input signal OFF (open) ■ E: Active when CN1...11 input signal OFF (open) ■ F: Active when CN1...12 input signal OFF (open)	3

SERVOPACK Parameter	Parameter.Digit	Description	Default value
Pn511	Pn511.1	EXT1 (probe 1 latch input) signal allocation ■ 0 ... 3: The signal is always inactive ■ 4: Active when CN1...10 input signal ON (closed) ■ 5: Active when CN1...11 input signal ON (closed) ■ 6: Active when CN1...12 input signal ON (closed) ■ 7 ... C: The signal is always inactive ■ D: Active when CN1...10 input signal OFF (open) ■ E: Active when CN1...11 input signal OFF (open) ■ F: Active when CN1...12 input signal OFF (open)	4
Pn511	Pn511.2	EXT2 (probe 2 latch input) signal allocation ■ 0 ... 3: The signal is always inactive ■ 4: Active when CN1...10 input signal ON (closed) ■ 5: Active when CN1...11 input signal ON (closed) ■ 6: Active when CN1...12 input signal ON (closed) ■ 7 ... C: The signal is always inactive ■ D: Active when CN1...10 input signal OFF (open) ■ E: Active when CN1...11 input signal OFF (open) ■ F: Active when CN1...12 input signal OFF (open)	5

Please do not change these parameters

When calling the Init block Y_SIG7PN_ServoInit, the following parameters are set. These should not be changed:

- *PnC00 ... PnC0F - Setpoint telegram: PZD 1 ... 16*
- *PnC10 ... PnC1F - Actual value telegram: PZD 1 ... 16*
- *PnC20 - Telegram selection*
- *PnB02 - Position user unit: Numerator*
- *PnB04 - Position user unit: Denominator*
- *PnB06 - Velocity user unit: Numerator*
- *PnB08 - Velocity user unit: Denominator*
- *PnB0A - Acceleration user unit: Numerator*
- *PnB42 - Position range limit (min.)*
- *PnB44 - Position range limit (max.)*
- *PnB48 - Software position limit (min.)*
- *PnB4A - Software position limit (max.)*
- *PnB0C - Acceleration user unit: Denominator*
- *PnB4C (607Fh): Max. profile velocity (Default: In reference to PnBF0 (2312h): Max. motor velocity)*
- *PnB7C (60C5h): Max. acceleration (Default: In reference to PnBF2 (2313h): Max. motor acceleration)*
- *PnB7E (60C6h): Max. deceleration (Default: In reference to PnBF2 (2313h): Max. motor acceleration)*
- *Pn205 - Multiturn limit*

3.3 Usage in Siemens SIMATIC Manager

3.3.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- The blocks can be used with the following CPUs:
  - System SLIO CPU 017-CEFPR00 with expansion of work memory to 2MB
  - System 300S CPU 315-4PN43 with expansion of work memory to 1MB
  - System 300S CPU 315-4PN23
  - System 300S CPU 317-4PN23
- The configuration of the System SLIO CPU happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *‘VIPA SLIO CPU’*. The *‘VIPA SLIO System’* is to be installed in the hardware catalog by means of the GSDML.
- The configuration of the System 300S CPUs 315-4PNxx takes place in the Siemens SIMATIC Manager as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
- The configuration of the System 300S CPU 317-4PN23 takes place in the Siemens SIMATIC Manager as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).
- The configuration of the PROFINET IO controller for the drives is done in the Siemens SIMATIC Manager.

3.3.2 Hardware configuration

3.3.2.1 Hardware configuration System SLIO CPU 017PN

Installing the IO device *‘VIPA SLIO System’*

The configuration of the System SLIO CPU happens by means of a virtual PROFINET IO device *‘VIPA SLIO CPU’*. The installation of the PROFINET IO device *‘VIPA SLIO CPU’* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com.
2. Download the configuration file for your CPU from the download area via *‘Config files ➔ PROFINET’*.
3. Extract the file into your working directory.
4. Start the Siemens hardware configurator.
5. Close all the projects.
6. Select *‘Options ➔ Install new GSD file’*.
7. Navigate to your working directory and install the according GSDML file.

   ⇨ After the installation the according PROFINET IO device can be found at *‘PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA SLIO System’*.

Install GSDML for YASKAWA *Sigma-7* servo drive

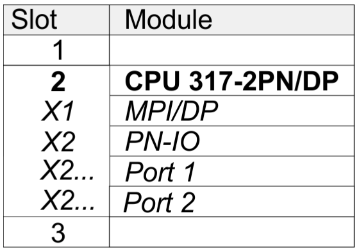
The sample project includes the GSDML file for the Sigma-7 servo drive.

The installation of GSDML for *Sigma-7* servo drives happens in the hardware catalog with the following approach:

1. Extract the GSDML file from your sample project to your working directory.
2. Start the Siemens hardware configurator.
3. Close all the projects.
4. Select *‘Options ➔ Install new GSD file’*.
5. Navigate to your working directory and install the according GSDML file.

   ⇨ After the installation the PROFINET device *‘SGD7...’* can be found at *‘PROFINET IO ➔ Additional field devices ➔ Drives ➔ YASKAWA Drives’*.

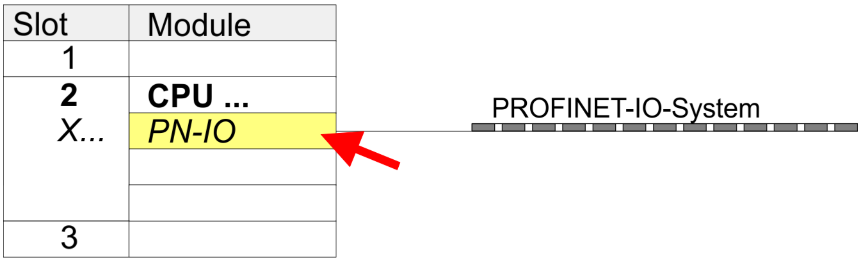
Add CPU in the project



Slot	Module
1	
2	**CPU 317-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

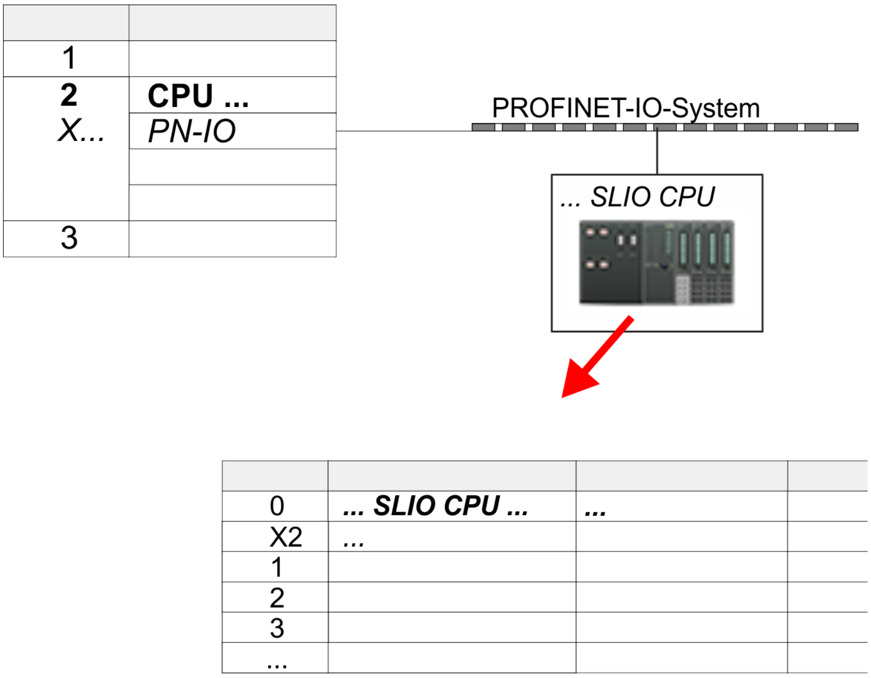
To be compatible with the Siemens SIMATIC Manager the following steps should be executed:

1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Place at *'Slot'* number 2 the CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2).
4. Click at the sub module *'PN-IO'* of the CPU.



Slot	Module
1	
2	**CPU ...**
X...	*PN-IO*
3	

5. Use [New] to create a new subnet and assign valid IP address data for your PROFINET system.
6. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
7. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



1	
2	**CPU ...**
X...	*PN-IO*
3	

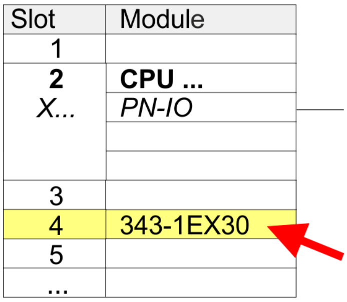
0	***... SLIO CPU ...***	**...**	
X2	...		
1			
2			
3			
...			

8. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA SLIO System'* and connect the IO device, which corresponds to your CPU *'017-CEFPR00 ➔ FW V2.4'*, to your PROFINET system.

   ⇨ In the slot overview of the PROFINET IO device *'VIPA SLIO CPU'* the CPU is already placed at slot 0. From slot 1 you can place your System SLIO modules.

Configuration of Ethernet PG/OP channel

The CPU has an integrated Ethernet PG/OP channel. This channel allows you to program and remote control your CPU.



Slot	Module
1	
2	**CPU ...**
X...	*PN-IO*
3	
4	343-1EX30
5	
...	

1. Place for the Ethernet PG/OP channel at slot 4 the Siemens CP 343-1 (SIMATIC 300 \ CP 300 \ Industrial Ethernet \CP 343-1 \ 6GK7 343-1EX30 0XE0 V3.0).

2. Open the properties dialog by clicking on the CP 343-1EX30 and enter for the CP at *'Properties'* the IP address data. You get valid IP address parameters from your system administrator.

3. Assign the CP to a *'Subnet'*. The IP address data are not accepted without assignment!

> *More information about the usage of the Ethernet PG/OP channel can be found in the manual of the CPU.*

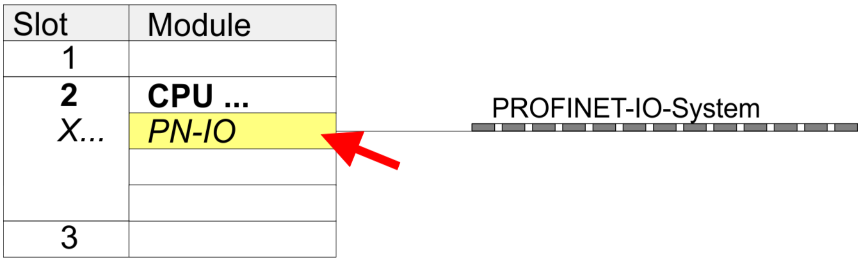
3.3.2.2 Hardware configuration System 300S CPU 315PN ... 317PN

Add CPU in the project

Slot	Module
1	
2	**CPU 315-2 PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
...	...
3	

To be compatible with the Siemens SIMATIC Manager the following steps should be executed:

1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Place at *'Slot'* number 2 for CPU 315PN the Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2) and for CPU 317PN the Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).
4. Click at the sub module *'PN-IO'* of the CPU.



5. Use [New] to create a new subnet and assign valid IP address data for your PROFINET system.
6. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
7. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.

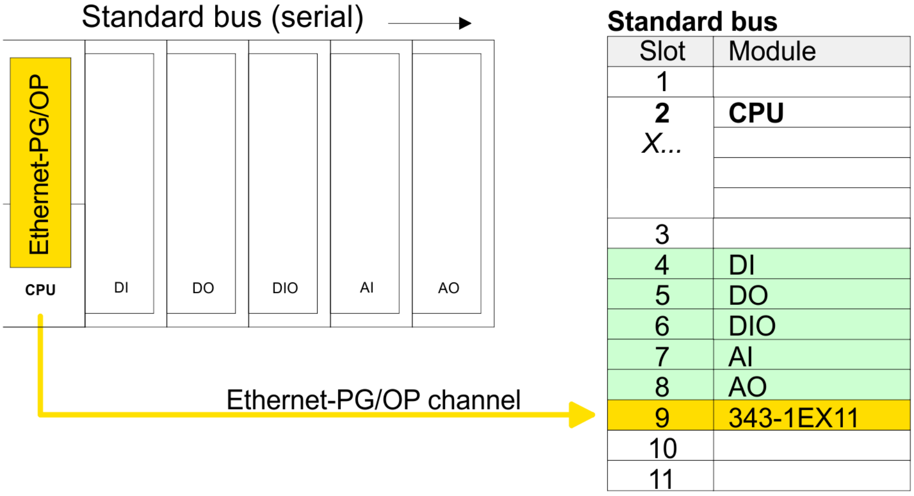
Configuration of Ethernet PG/OP channel

The CPU has an integrated Ethernet PG/OP channel. This channel allows you to program and remote control your CPU.

1. Configure the modules on the standard bus.
2. Place for the internal Ethernet PG/OP channel always below the really plugged modules a Siemens CP 343-1 (SIMATIC 300 \ CP 300 \ Industrial Ethernet \CP 343-1 \ 6GK7 343-1EX11 0XE0).
3. Open the properties dialog by clicking on the CP 343-1EX11 and enter for the CP at *'Properties'* the IP address data from the initialization.
4. Assign the CP to a *'Subnet'*. The IP address data are not accepted without assignment!

5. Transfer your project to your CPU.

   ⇨ The IP address data are stored in your current project.



Standard bus

Slot	Module
1	
2 *X...*	**CPU**
3	
4	DI
5	DO
6	DIO
7	AI
8	AO
9	343-1EX11
10	
11	

More information about the initialization and the usage of the Ethernet PG/OP channel can be found in the manual of the CPU.

3.3.2.3 Insert and configure *Sigma-7* PROFINET drive

***Sigma-7* Insert and configure PROFINET drive**

The drive is configured in the Siemens SIMATIC Manager. Here, a Sigma-7 PROFINET device must be configured for each axis.

1. Select your *Sigma-7* PROFINET drive *'SGD7S-xxx ...'* from the hardware catalog and drag it onto the *'PROFINET-IO-System'*.

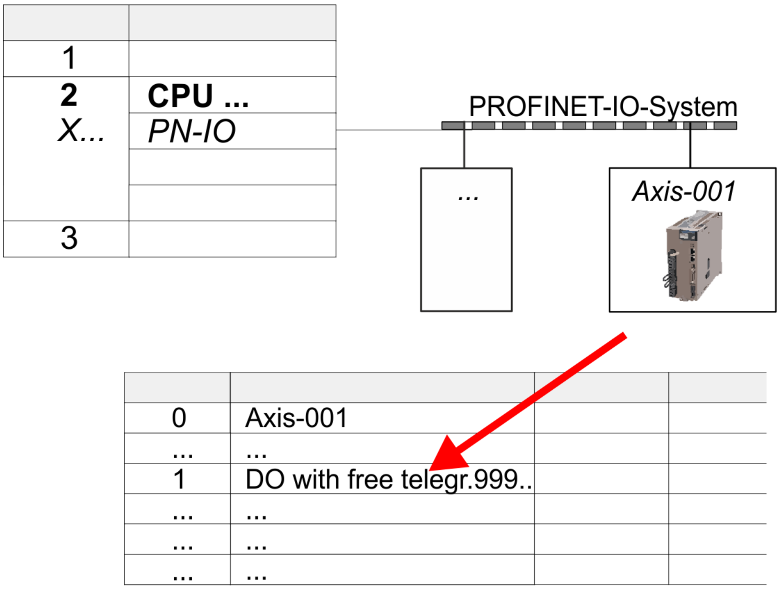
   ⇨ The *Sigma-7* PROFINET drive is connected to the IO controller and can now be configured.

2. Open the object properties of the *Sigma-7* PROFINET drive and assign a suitable *'Device name'* such as Axis-001.

3. Select your *Sigma-7* PROFINET drive *'SGD7S-xxx ...'* from the hardware catalog and drag the element *'DO with YASKAWA telegr.999...'* onto slot 1 of the *Sigma-7* PROFINET drive.

4. Make a note of the diagnostic address of *'DO with free telegr.999...'* in the slot overview.

> ***Please note!***
>
> – *The link between the axes in the hardware configuration and the user program is made via the respective diagnostic address of the sub module 'DO with free telegr.999...'.*
>
> – *The start addresses of the 'I address' and 'O address' of the sub module 'Free telegram PZD-16/16' must be identical.*



1	
2	**CPU ...**
X...	*PN-IO*
3	

0	Axis-001		
...	...		
1	DO with free telegr.999..		
...	...		
...	...		
...	...		

5. Save, compile your configuration and transfer it into your CPU.

6. For the name assignment, *'PLC ➔ Ethernet ➔ Edit Ethernet node'*. Here click at *'Search'*.

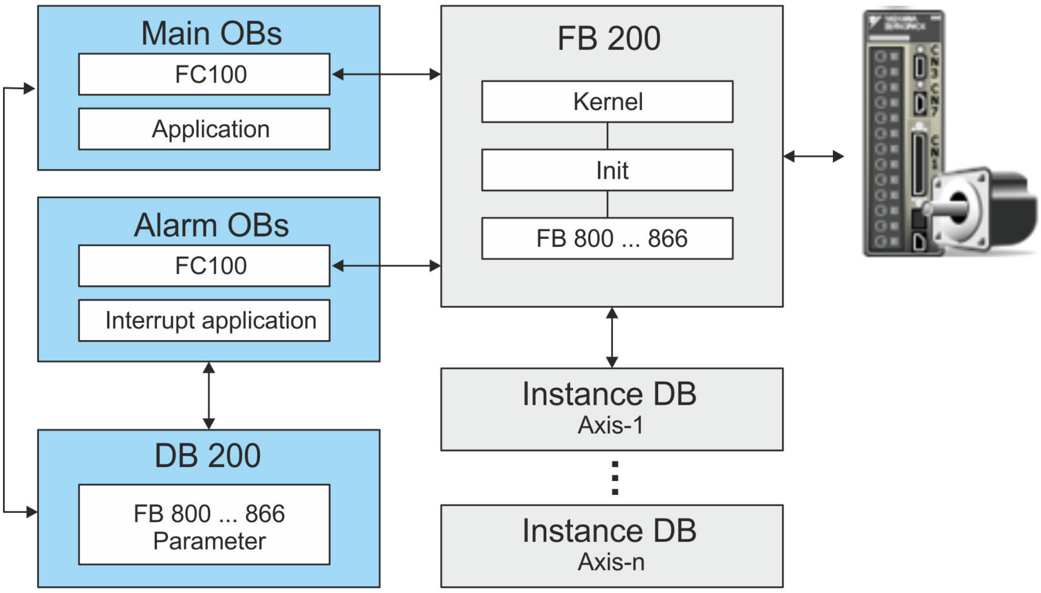
   ⇨ The Ethernet stations are listed.

7. Search the Sigma7 PROFINET drive *'SGM7xxxACxx'* and assign it the name from the hardware configuration.

8. Save, compile your configuration and transfer it into your CPU.

3.3.3 User program

3.3.3.1 Program structure



- *Init*
  - The *Init* blocks are used to configure axes.
  - The *Init* blocks are called cyclically within the FB 200 and executed if necessary.
  - Specific block for *Sigma-7* PROFINET.
  - The configuration for the initialization takes place via the corresponding instance DB.
- *Kernel*
  - The *Kernel* blocks are called cyclically within the FB 200.
  - The *Kernel* blocks communicate with the drive via PROFINET, process the user requests and return status messages.
  - The exchange of data takes place by means of the corresponding instance DB.
- FB 200
  - Within the FB 200, all function blocks of the library are called cyclically and executed if necessary.
  - The DB to be specified during the call serves as instance DB for the corresponding axis.
  - The reference to the axis is the diagnostic address, which must be specified in the hardware configuration of the corresponding axis.
- Instance DB - axis DB
  - For the FB 200, an instance DB must be created for each axis.
  - In the instance DB, the FB 200 handles parameter, status and configuration data for the corresponding axis.
  - In the example DB 101 is for axis 1, DB 102 for axis 2 and so on.
- FB 800 ... FB 866 - *PLCopen*
  - The PLCopen blocks are used to program motion sequences and status queries.
  - All PLCopen blocks are called cyclically within the FB 200 and executed if necessary.
  - In the application, these blocks are used by means of the DB 200.

- DB 200
  - All parameters of all blocks are interconnected in the DB 200.
  - Accessing the DB 200 from your application enables you to control your drive system.
- FC 100
  - In FC 100, the FB 200 calls must be programmed for each axis.
  - To use simple motion control, the FC 100 must be called cyclically in the user program.
  - So that the variables from the simple motion control are available in the corresponding interrupt OB, you must program an FC 100 call for each interrupt OB.

> *Please note that you execute a STOP/HALT command after a maximum of 2 different movement tasks. Otherwise, following motion tasks are ignored!*

3.3.3.2 Programming

Include library

1. Go to the service area of the according Web page. The following Web pages give you access to the libraries:
   - *https://www.yaskawa.eu.com/en/service/drives-motion-software-download*
   - *http://www.vipa.com/en/service-support/manuals/yaskawa-lib*
2. Download the from the download area the block library *Sigma-7 - YASKAWA Motion Control PROFINET*.
3. Open the dialog window for ZIP file selection via *'File → Retrieve'*.
4. Select the according ZIP file and click at [Open].
5. Specify a target directory in which the blocks are to be stored and start the unzip process with [OK].

Copy blocks into project

Open the library after unzipping and drag and drop the following blocks into *'Blocks'* of your project:

- *Sigma-7* PROFINET:
  - FB 200 - USER_IF
  - DB 200 - DB_USER_IF
  - FC 100 - USER_CallSMC
  - all UDTs
  - FB 849 - Y_Init
  - FB 862 - Y_SIG7PN_Kernel
  - FB 863 - Y_SIG7PN_DeviceDriver
  - FB 865 - Y_SIG7PN_ServoInit
  - FB 866 - Y_SIG7PN_ServoOrder
  - FC 1 - Y_GCD
  - FC 260 - Y_CheckREAL
- Axis control
  - FB 800 ... FB 847: Blocks for your movement sequences

Create instance DB for axes

An instance DB must be created for each axis.

- In your project, click at *'Blocks'* and choose *'Context menu ➔ Insert new object ➔ Data block'*.

  Specify the following parameters:

  - Name and type
    - The DB number as *'Name'* can freely be chosen, such as DB101 for axis 1.
    - Set *'Shared DB'* of *'FB 200'* as the *'Type'*.
  - Symbolic name
    - Enter "FB200_Axis01_DB".

  Confirm your input with [OK].

  ⇨ The block is created.

FC 100 - USER_CallSMC

Simple motion calling structure

- Open FC 100 and program an FB 200 call for each axis according to the following structure:

```
CALL  FB 200, DB n
 DiagAddress  :=DW#16# [Diagnostics address axis n]
 Axis_IF      :="DB_USER_IF".Axis_IF[n]
```

DiagAddress - Diagnostics address from the hardware configuration of axis n

Axis_IF - Axis data of axis n within DB 200 - DB_USER_IF

Example for 3 axes

```
CALL  FB 200, DB 101                    //Instance DB axis 1
 DiagAddress  :=DW#16#7DC               //from HW config axis 1
 Axis_IF     :="DB_USER_IF".Axis_IF[1]  //Data axis 1 DB 200

CALL  FB 200, DB 102                    //Instance DB axis 2
 DiagAddress  :=DW#16#7DD               //from HW config axis 2
 Axis_IF     :="DB_USER_IF".Axis_IF[2]  //Data axis 2 DB 200

CALL  FB 200, DB 103                    //Instance DB axis 3
 DiagAddress  :=DW#16#7DE               //from HW config axis 3
 Axis_IF     :="DB_USER_IF".Axis_IF[3]  //Data axis 3 DB 200
```

For the motion tasks, you must copy the PLCopen blocks to your project. Within FB 200, these are called cyclically. By means of DB 200 - DB_USER_IF you can supply blocks with parameters and activate their call. More details can be found in the example project.

OB 1

User program

1. Open OB 1 and program an FC 100 call:

   ```
   CALL  FC 100
   ```

2. Program your user application.

Create interrupt OBs

1. In your project, click at *'Blocks'* and choose *'Context menu ➔ Insert new object ➔ Organization block'*.

   ⇨ The dialog *'Properties Organization block'* opens.

2. Add OB 57, OB 82, and OB 86 successively to your project.

So that the variables of the simple motion control are available, in the corresponding interrupt OB, you have to program a FC 100 call:

```
CALL FC 100
```

3.3.3.3 Sequence of operations

Sequence of operations

1. Choose the Siemens SIMATIC Manager and transfer your project into the CPU.

   ⇨ You can take your application into operation now.

   > **CAUTION!**
   > Please always observe the safety instructions for your drive, especially during commissioning!

2. Before an axis can be controlled, it must be initialized. For this in DB 200 set the value *'Axis_IF [x].Init.Execute'* to TRUE.

   ⇨
   - The output *'Axis_IF[x].Init.Busy'* returns TRUE.
   - If there is no error, the output *'Axis_IF[x].Init.Done'* returns TRUE.
   - In the event of an error, the output *'Axis_IF[x].Init.Error'* returns TRUE. Here you will receive the corresponding error code via the output *'Axis_IF[x].Init.ErrorID'*.
   - When changing initialization parameters in DB 200, set the value *'Axis_IF[x].Init.Execute'* to TRUE again.

   Do not continue as long as the Init block reports any errors!

3. Program your application with the corresponding FB 200 call sequences.

3.4 Usage in Siemens TIA Portal

3.4.1 Precondition

Overview

- Please use for configuration the Siemens TIA Portal V15.1 and up.
- The blocks can be used with the following CPUs:
  - System SLIO CPU 017-CEFPR00 with expansion of work memory to 2MB
  - System 300S CPU 315-4PN43 with expansion of work memory to 1MB
  - System 300S CPU 315-4PN23
  - System 300S CPU 317-4PN23
- The configuration of the System SLIO CPU happens in the Siemens TIA Portal by means of a virtual PROFINET IO device *'VIPA SLIO CPU'*. The *'VIPA SLIO System'* is to be installed in the hardware catalog by means of the GSDML.
- The configuration of the System 300S CPUs 315-4PNxx takes place in the Siemens TIA Portal as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
- The configuration of the System 300S CPU 317-4PN23 takes place in the Siemens TIA Portal as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).
- The configuration of the PROFINET master for the drives is done in the Siemens TIA Portal.

3.4.2 Hardware configuration

3.4.2.1 Hardware configuration System SLIO CPU 017PN

Installing the IO device *'VIPA SLIO System'*

The configuration of the System SLIO CPU happens by means of a virtual PROFINET IO device *'VIPA SLIO CPU'*. The installation of the PROFINET IO device *'VIPA SLIO CPU'* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com.
2. Download the configuration file for your CPU from the download area via *'Config files ➔ PROFINET'*.
3. Extract the file into your working directory.
4. Start the Siemens TIA Portal.
5. Close all the projects.
6. Switch to the *Project view*.
7. Select *'Options ➔ Install general station description file (GSD)'*.
8. Navigate to your working directory and install the according GSDML file.

   ⇨ After the installation the hardware catalog is refreshed and the Siemens TIA Portal is closed.

   After restarting the Siemens TIA Portal the according PROFINET IO device can be found at *Other field devices* > *PROFINET* > *IO* > *VIPA GmbH* > ....

Thus, the specific components can be displayed, you have to deactivate the "Filter" of the hardware catalog.

Install GSDML for YASKAWA *Sigma-7* servo drive

The sample project includes the GSDML file for the Sigma-7 servo drive.

The installation of GSDML for *Sigma-7* servo drives happens in the hardware catalog with the following approach:

1. Extract the GSDML file from your sample project to your working directory.
2. Start the Siemens TIA Portal.
3. Close all the projects.
4. Switch to the *Project view*.
5. Select *'Options ➔ Install general station description file (GSD)'*.
6. Navigate to your working directory and install the according GSDML file.

   ⇨ After the installation the hardware catalog is refreshed and the Siemens TIA Portal is closed.

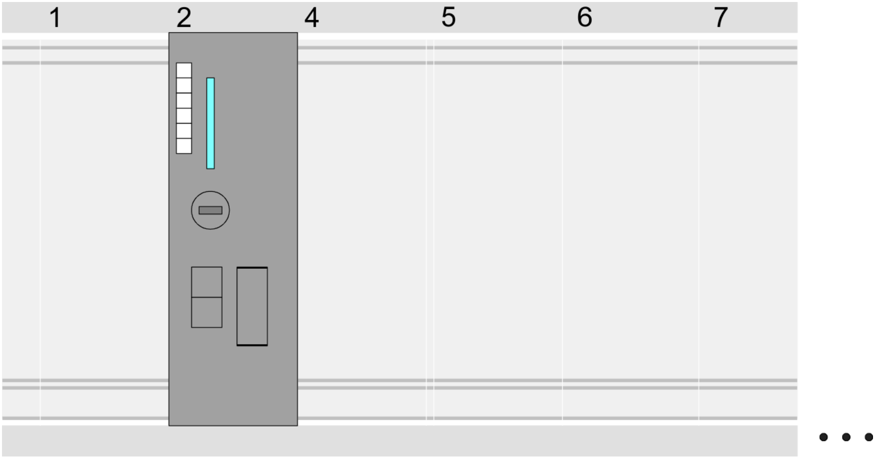
   After restarting the Siemens TIA Portal the PROFINET device *'SGD7...'* can be found at *'PROFINET IO ➔ Additional field devices ➔ Drives ➔ YASKAWA Drives'*.

Add CPU in the project

To be compatible with the Siemens TIA Portal the following steps should be executed:

1. Start the Siemens TIA Portal with a new project.
2. Switch to the *Project view*.
3. Click in the *Project tree* at *'Add new device'*.
4. For the CPU 017-CEFPR00, configure the Siemens CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2).

   ⇨ The CPU is inserted with a profile rail.

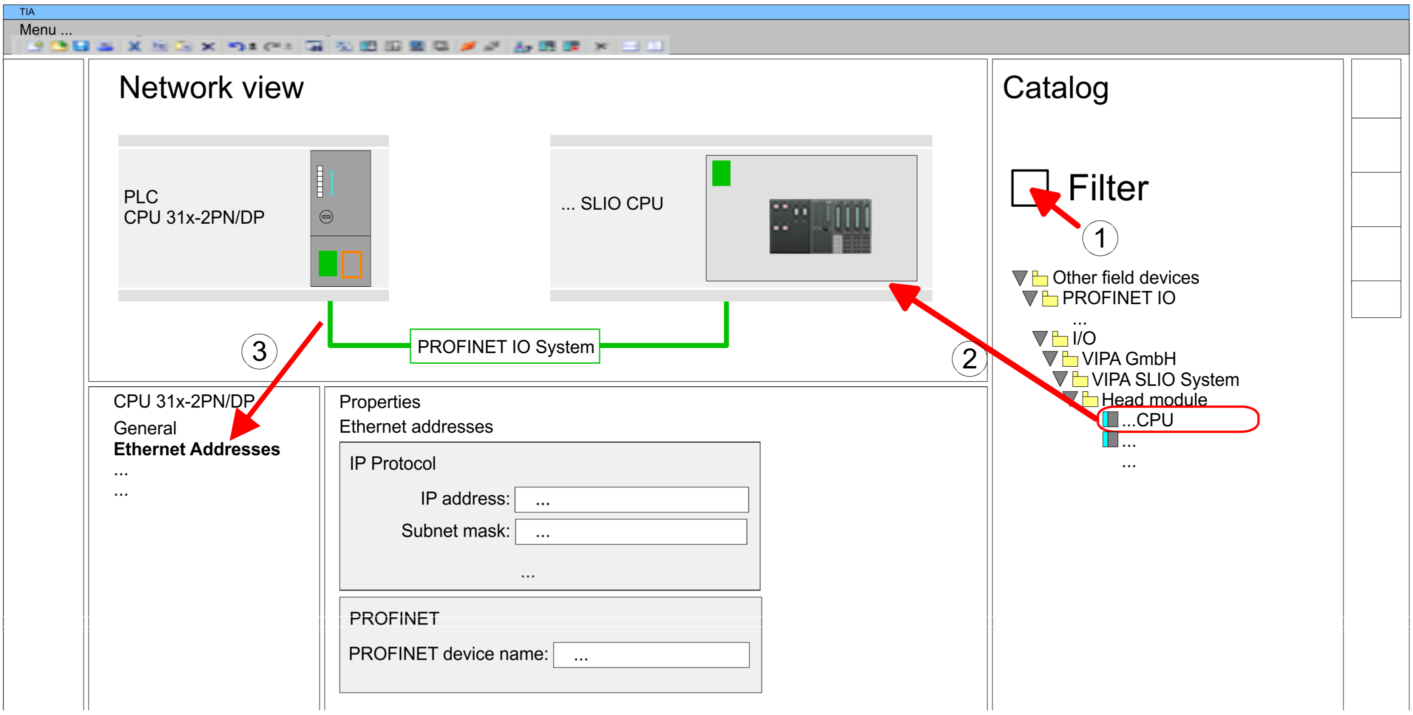


Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 317-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET interface...		2 X2		PROFINET interface	
...					

Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.
2. Navigate in the hardware catalog to *'Other field devices ➔ PROFINET IO ➔ I/O ➔ VIPA GmbH ➔ VIPA SLIO System'* and connect the slave system to the CPU by dragging&dropping it from the hardware catalog to the *Network view* and connecting it via PROFINET to the CPU.
3. Click in the *Network view* at the PROFINET part of the Siemens CPU and enter valid IP address data in *'Properties'* at *'Ethernet address'* in the area *'IP protocol'*.
4. Enter at *'PROFINET'* a *'PROFINET device name'*. The device name must be unique at the Ethernet subnet.



Properties
Ethernet addresses

IP Protocol

IP address: ...

Subnet mask: ...

...

PROFINET

PROFINET device name: ...

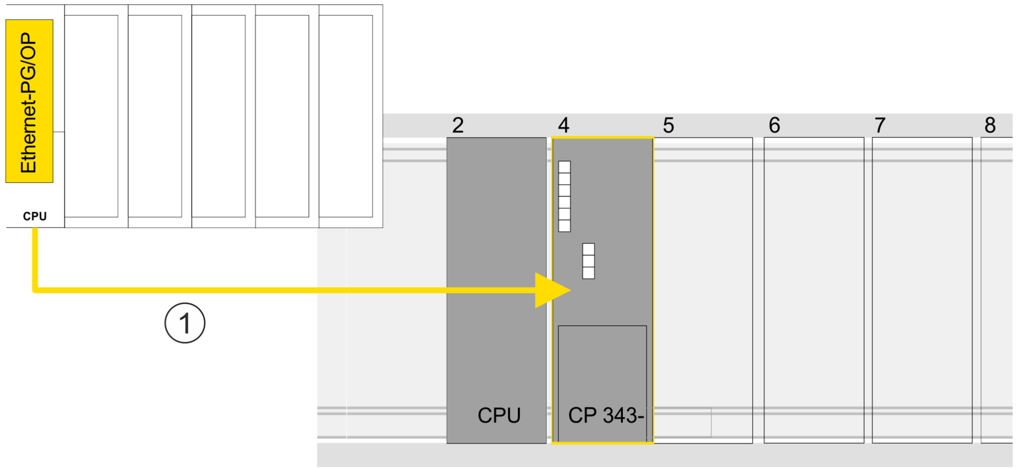
5. Select in the *Network view* the IO device *'VIPA SLIO CPU 017-CEFPR00'* and switch to the *Device overview*.

   ⇨ In the *Device overview* of the PROFINET IO device *'VIPA SLIO CPU 017-CEFPR00'* the CPU is already placed at slot 0. From slot 1 you can place your System SLIO modules.

Configuration of Ethernet PG/OP channel

The CPU has an integrated Ethernet PG/OP channel. This channel allows you to program and remote control your CPU.

1. As Ethernet PG/OP channel place at slot 4 the Siemens CP 343-1 (6GK7 343-1EX30 0XE0 V3.0).
2. Open the "Property" dialog by clicking on the CP 343-1EX30 and enter for the CP at "Properties" at "Ethernet address" the IP address data, which you have assigned before.
3. Transfer your project.



Device overview:

Module	...	Slot	...	Type	...
PLC ...		2		CPU ...	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET interface		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

More information about the usage of the Ethernet PG/OP channel can be found in the manual of the CPU.

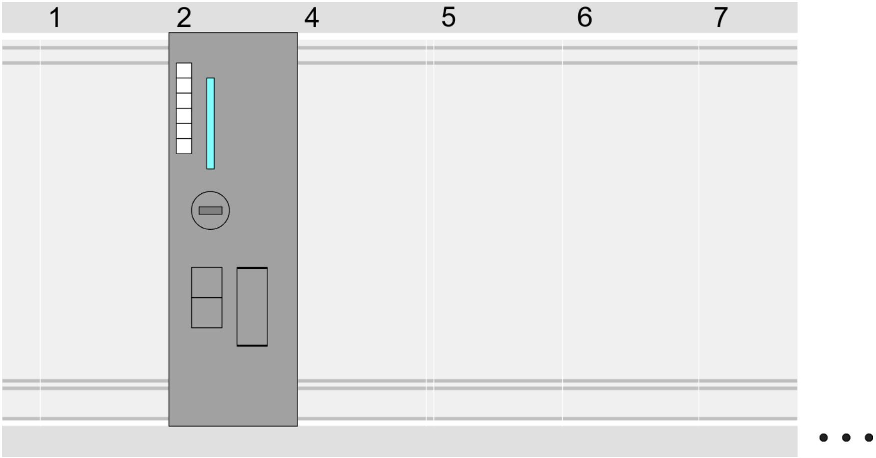
3.4.2.2 Hardware configuration System 300S CPU 315PN ... 317PN

Configuration Siemens CPU

To be compatible with the Siemens TIA Portal the following steps should be executed:

1. Start the Siemens TIA Portal.
2. Create a new project in the *Portal view* with *'Create new project'*.
3. Switch to the *Project view*.
4. Click in the *Project tree* at *'Add new device'*.
5. For the CPU 315PN, select the Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2) and for the CPU 317PN the Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

   ⇨ The CPU is inserted with a profile rail.



Device overview:

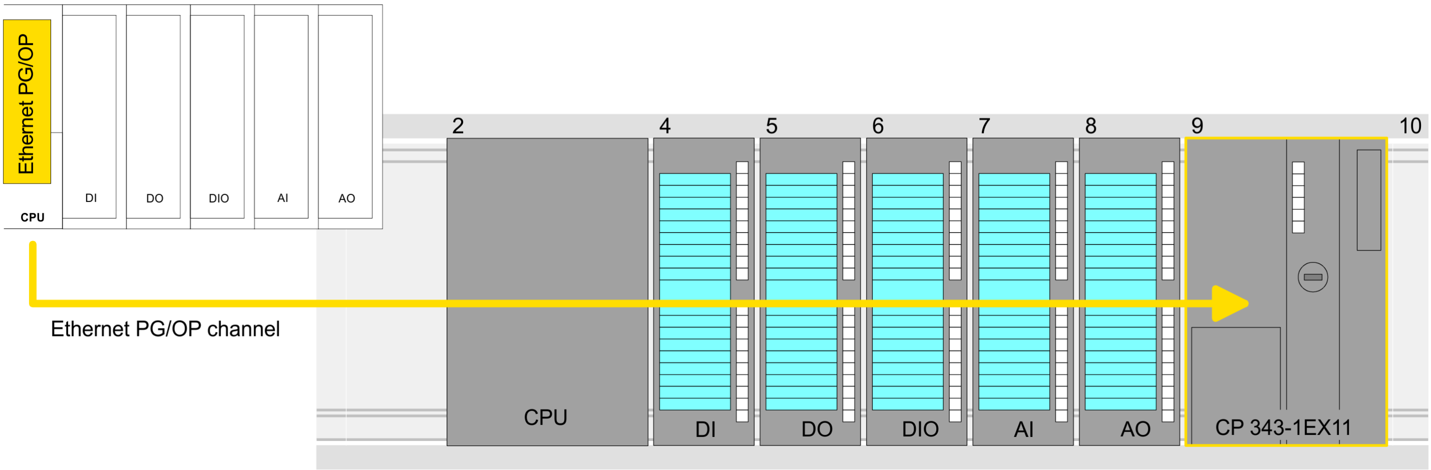
Module	...	Slot	...	Type	...
PLC ...		2		CPU 315-2PN/DP	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET interface		2 X2		PROFINET interface	
...		...		...	

Configuration of Ethernet PG/OP channel

The CPU has an integrated Ethernet PG/OP channel. This channel allows you to program and remote control your CPU.

1. Configure your System 300 modules
2. For the Ethernet PG / OP channel, always configure a Siemens CP 343-1 (6GK7 343-1EX11 0XE0) as the last module after the modules that are actually plugged.
3. Open the properties dialog by double-click the CP 343-1EX11 and enter for the CP at *'Properties'* the IP address data from the initialization.
4. Transfer your project to your CPU.

   ⇨ The IP address data are stored in your current project.



Device overview

Module	...	Slot	...	Type	...
PLC...		2		CPU ...	
...		...		...	
		3			
DI...		4		DI...	
DO...		5		DO...	
DIO...		6		DIO...	
AI...		7		AI...	
AO...		8		AO...	
🟨 CP 343-1		9		CP 343-1	

More information about the initialization and the usage of the Ethernet PG/OP channel can be found in the manual of the CPU.

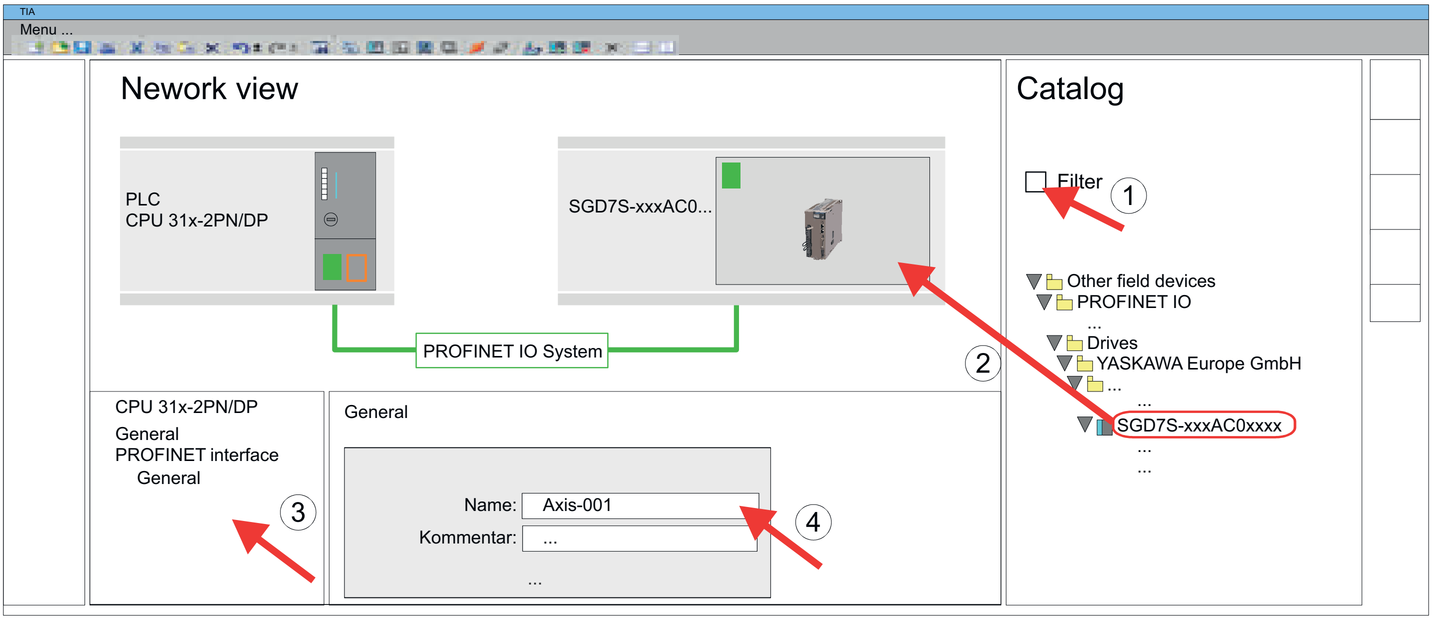
3.4.2.3 Insert and configure *Sigma-7* PROFINET drive

***Sigma-7* Insert and configure PROFINET drive**

The drive is configured in the Siemens TIA Portal. Here, a Sigma-7 PROFINET device must be configured for each axis.

1. Switch to the *'Network view'*.
2. Deactivate *'Filter'* in the hardware catalog.
3. Navigate in the hardware catalog to *'Other field devices ➔ PROFINET IO ➔ Drives ➔ YASKAWA Europe GmbH ➔ YASKAWA Drives ➔ Head module'*. Select your *Sigma-7* PROFINET drive *'SGD7S-xxx ...'* from the hardware catalog and drag it onto the *'PROFINET-IO-System'*.

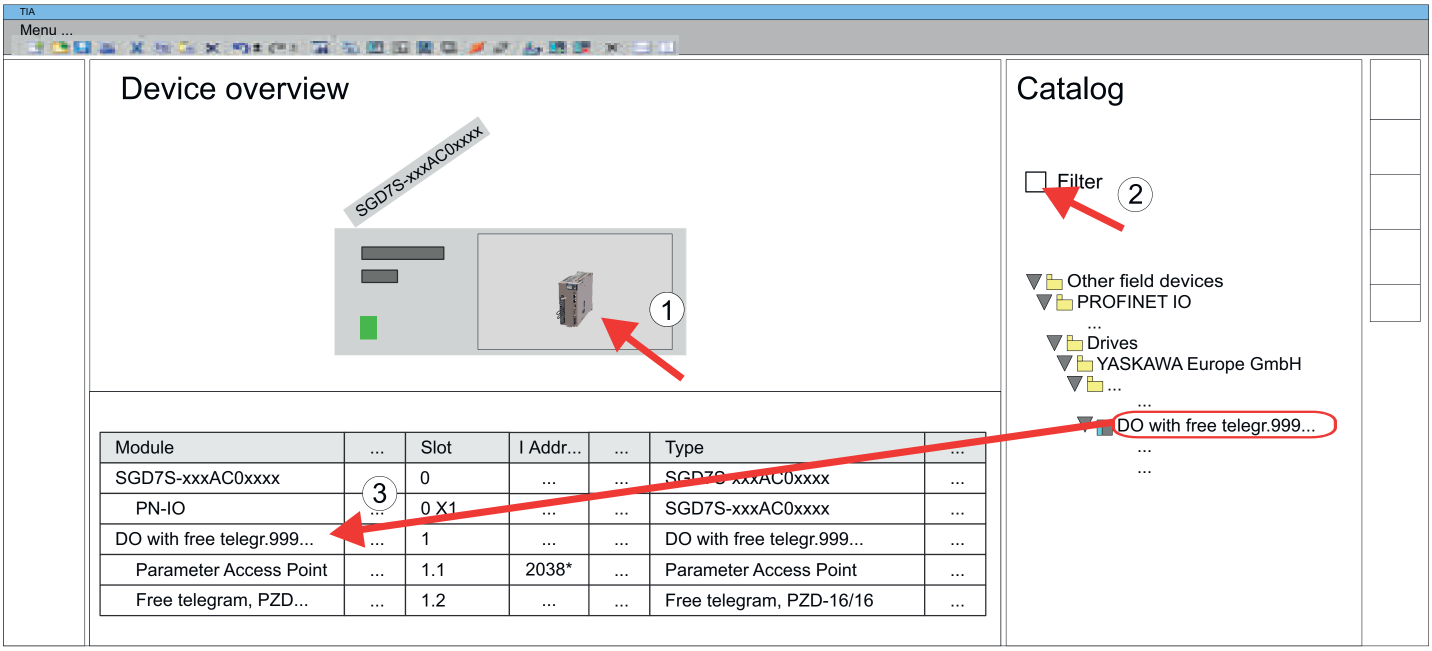
   ⇨ The *Sigma-7* PROFINET drive is connected to the master and can now be configured.
4. Open the *'Properties'* of the *Sigma-7* PROFINET drive and assign a suitable *'Device name'* such as Axis-001.



Name: Axis-001
Kommentar: ...
...

5. Switch to the *'Device view'* of the *Sigma-7* PROFINET drive.

Select your *Sigma-7* PROFINET drive *'SGD7S-xxx ...'* from the hardware catalog and drag the element *'DO with free telegr.999...'* onto slot 1 of the *'Device overview'* of the *Sigma-7* PROFINET drive.



Module	...	Slot	I Addr...	...	Type	...
SGD7S-xxxAC0xxxx		0	...	...	SGD7S-xxxAC0xxxx	...
PN-IO	...	0 X1	...	...	SGD7S-xxxAC0xxxx	...
DO with free telegr.999...	...	1	...	...	DO with free telegr.999...	...
Parameter Access Point	...	1.1	2038*	...	Parameter Access Point	...
Free telegram, PZD...	...	1.2	...	...	Free telegram, PZD-16/16	...

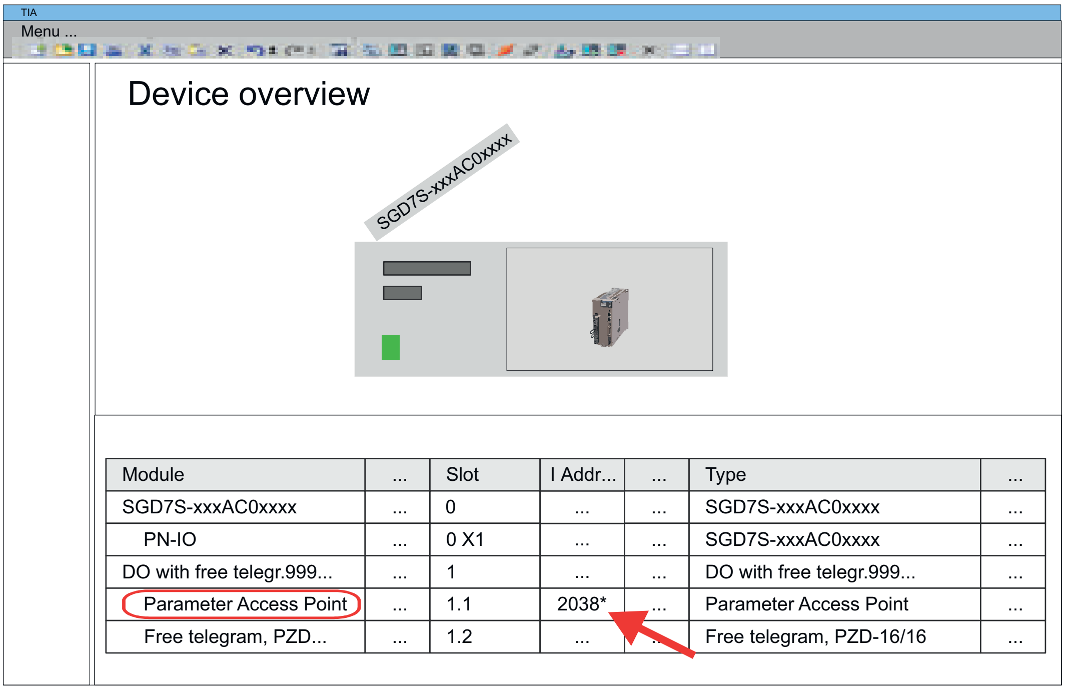
6.

Please note!

– *The link between the axes in the hardware configuration and the user program is made via the I address of 'Parameter Access Point' of the submodule 'DO with free telegr.999...'.*

– *The start addresses of 'I address' and the 'Q address' of the submodule 'Free telegram PZD-16/16' must be identical.*

7. Make a note of the I address of *'Parameter Access Point'* of the submodule *'DO with free telegr.999...'* in the device overview.



Module	...	Slot	I Addr...	...	Type	...
SGD7S-xxxAC0xxxx	...	0	...	...	SGD7S-xxxAC0xxxx	...
PN-IO	...	0 X1	...	...	SGD7S-xxxAC0xxxx	...
DO with free telegr.999...	...	1	...	...	DO with free telegr.999...	...
Parameter Access Point	...	1.1	2038*	...	Parameter Access Point	...
Free telegram, PZD...	...	1.2	...	...	Free telegram, PZD-16/16	...

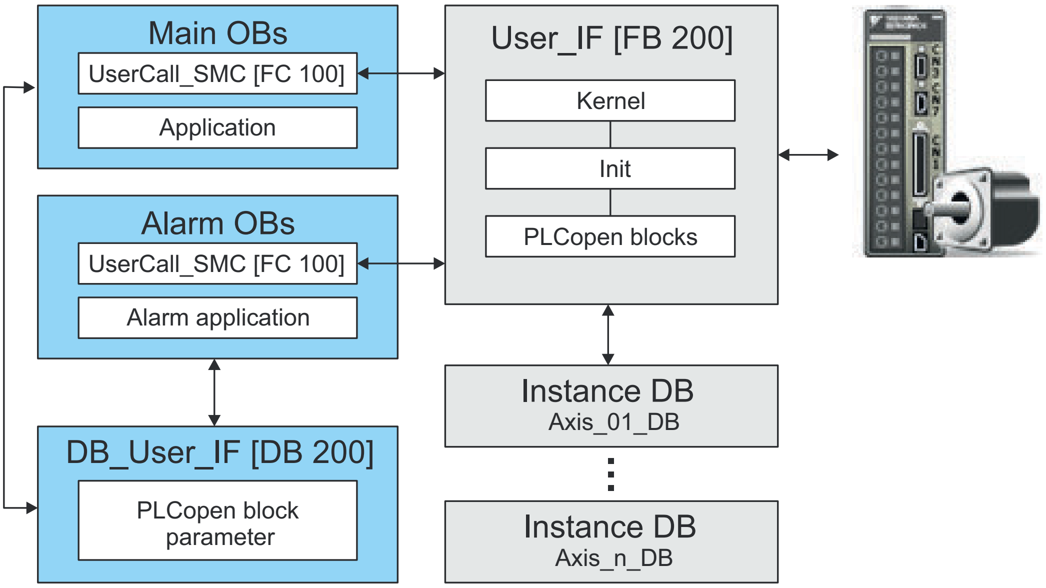
8. Save, compile your configuration and transfer it to the CPU.
9. For the name assignment select *'Online → Accessible devices'*. Select the PROFINET interface and click on *'Start search'*.

   ⇨ The Ethernet stations are listed.
10. Search the Sigma7 PROFINET drive *'SGM7xxxACxx'* and assign it the name from the hardware configuration such as Axis-001.
11. Save, compile your configuration and transfer it to the CPU.

3.4.3 User program

3.4.3.1 Program structure

> *Please note that the block numbers may vary as they are automatically generated in the Siemens TIA Portal.*



- *Init*
  – The *Init* blocks are used to configure axes.
  – The *Init* blocks are called cyclically within the *User_IF [FB 200]* and executed if necessary.
  – Specific block for *Sigma-7* PROFINET.
  – The configuration for the initialization takes place via the corresponding instance DB.
- *Kernel*
  – The *Kernel* blocks are called cyclically within the *User_IF [FB 200]*.
  – The *Kernel* block communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – The exchange of data takes place by means of the corresponding instance DB.

- *User_IF [FB 200]*
  - Within the *User_IF [FB 200]*, all function blocks of the library are called cyclically and executed if necessary.
  - The DB to be specified during the call serves as instance DB for the corresponding axis.
  - The reference to the axis is the diagnostic address, which must be specified in the hardware configuration of the corresponding axis.
- Instance DB - axis DB
  - For the *User_IF [FB 200]*, an instance DB must be created for each axis.
  - In the instance DB, the *User_IF [FB 200]* handles parameter, status and configuration data for the corresponding axis.
  - In the example, *Axis_01_DB* is for axis 1, *Axis02_DB* for axis 2 etc.
- *PLCopen blocks*
  - The PLCopen blocks are used to program motion sequences and status queries.
  - All PLCopen blocks are called cyclically within the *User_IF [FB 200]* and executed if necessary.
  - In the application, these blocks are used by means of the *DB_User_IF [DB 200]*.
- DB_User_IF [DB 200]
  - All parameters of all blocks are interconnected in the *DB_User_IF [DB 200]*.
  - Accessing the *DB_User_IF [DB 200]* from your application enables you to control your drive system.
- UserCall_SMC [FC 100]
  - In *UserCall_SMC [FC 100]*, the *User_IF [FB 200]* calls must be programmed for each axis.
  - To use simple motion control, the *UserCall_SMC [FC 100]* must be called cyclically in the user program.
  - So that the variables from the simple motion control are available in the corresponding interrupt OB, you must program an *User_CallSMC [FC 100]* call for each interrupt OB.

> *Please note that you execute a STOP command after a maximum of 2 movement tasks. Otherwise, following motion tasks are ignored!*

3.4.3.2 Programming

Include library

1. Go to the service area of the according Web page. The following Web pages give you access to the libraries:
   - *https://www.yaskawa.eu.com/en/service/drives-motion-software-download*
   - *http://www.vipa.com/en/service-support/manuals/yaskawa-lib*
2. Load from the download area the block library *Sigma-7 - YASKAWA Motion Control PROFINET*.
3. Switch to the *Project view* of the Siemens TIA Portal.
4. Choose "Libraries" from the task cards on the right side.
5. Click at "Global library".
6. Click on the free area inside the *'Global Library'* and select *'Context menu ➔ Retrieve library'*.
7. Navigate to your work directory and load the ZIP file.
8. Select the according ZIP file and click at [Open].

Copy blocks into project

Open the library after unzipping and drag and drop the following blocks into *'Blocks'* of your project:

- *Sigma-7* PROFINET:
  - User_IF [FB 200]
  - DB_User_IF [DB 200]
  - User_CallSMC [FC 100]
  - all UDTs
  - Y_Init [FB 849]
  - Y_SIG7PN_Kernel [FB 862]
  - Y_SIG7PN_DeviceDriver [FB 863]
  - Y_SIG7PN_ServoInit [FB 865]
  - Y_SIG7PN_ServoOrder [FB 866]
  - Y_GCD [FC 1]
  - Y_CheckREAL [FC 260]
- Axis Control
  - PLCopen blocks: Blocks for your movement sequences

Create instance DB for axes

An instance DB must be created for each axis.

1. Click at *'Project tree ➔ ...CPU... ➔ Program blocks ➔ Add new block'*.

   ⇨ The dialog *'Add block'* is opened.
2. Select the block type *'DB block'* and specify the following parameters:
   - Name
     - Specify "Axis01_DB".
   - Type
     - Set *'Block ... FB 200'* as the *'Type'*.
   - Number
     - Leave *'Automatic'* enabled.
3. Confirm your input with [OK].

   ⇨ The block is created.

User_CallSMC [FC 100]

Simple motion call structure

➔ Open the *User_CallSMC [FC 100]* and program a *User_IF [FB 200]* call according to the following structure for each axis:

```
CALL  User_IF, Axis n_DB
 DiagAddress   :=DW#16# [Diagnoseadresse Achse n]
 Axis_IF       :="DB_USER_IF".Axis_IF[n]
```

DiagAddress - Enter the previously noted I address of *'Parameter Access Point'* of the submodule *'DO with free telegr.999 ...'* from the hardware configuration of axis n. ✎ *Chap. 3.3.2.3 'Insert and configure Sigma-7 PROFINET drive' page 18*

Axis_IF - Axis data of axis n within *DB_USER_IF [DB 200]*

Example call for 3 axes

```
CALL  User_IF, Axis01_DB                        //Instance DB axis 1
 DiagAddress   :=DW#16#7DC                      //from HW config axis 1
 Axis_IF      :="DB_USER_IF".Axis_IF[1]         //Data axis 1 DB 200

CALL  User_IF, Axis02_DB                        //Instance DB axis 2
 DiagAddress   :=DW#16#7DD                      //from HW config axis 2
 Axis_IF      :="DB_USER_IF".Axis_IF[2]         //Data axis 2 DB 200

CALL  User_IF, Axis03_DB                        //Instance DB axis 3
 DiagAddress   :=DW#16#7DE                      //from HW config axis 3
 Axis_IF      :="DB_USER_IF".Axis_IF[3]         //Data axis 3 DB 200
```

For the motion tasks, you must copy the PLCopen blocks to your project. Within User_IF [FB 200], these are called cyclically. By means of DB_USER_IF [DB 200] you can supply blocks with parameters and activate their call. More details can be found in the example project.

OB 1

User program

1. Open OB 1 and program an *User_CallSMC [FC 100]* call:

```
CALL  User_CallSMC
```

2. Program your user application.

Create interrupt OBs

1. Click at *'Project tree ➔ ...CPU... ➔ Program blocks ➔ Add new block'*.

   ⇨ The dialog *'Add block'* is opened.

2. Enter OB 57 and confirm with [OK].

   ⇨ The OB 57 is created.

3. Successively add OB 82 and OB 86 to your project.

So that the variables from the simple motion control are available in the corresponding interrupt OB, you must program an User_CallSMC [FC 100] call for each interrupt OB.

```
CALL User_CallSMC
```

3.4.3.3 Sequence of operations

Sequence of operations

1. Select *'Edit ➔ Compile'* and transfer the project into your CPU. You can find more information on the transfer of your project in the online help of the Siemens TIA Portal.

   ⇨ You can take your application into operation now.

   > ⚠ **CAUTION!**
   > Please always observe the safety instructions for your drive, especially during commissioning!

2. Before an axis can be controlled, it must be initialized. For this in *DB_User_IF [DB 200]* set the value *'Axis_IF [x].Init.Execute'* to TRUE.

   ⇨
   - The output *'Axis_IF[x].Init.Busy'* returns TRUE.
   - If there is no error, the output *'Axis_IF[x].Init.Done'* returns TRUE.
   - In the event of an error, the output *'Axis_IF[x].Init.Error'* returns TRUE. Here you will receive the corresponding error code via the output *'Axis_IF[x].Init.ErrorID'*.
   - When changing initialization parameters in *DB_User_IF [DB 200]*, set the value *'Axis_IF[x].Init.Execute'* to TRUE again.

   > *Do not continue as long as the Init block reports any errors!*

3. Program your application with the corresponding *User_IF [FB 200]* call sequences.

4 Blocks for axis control

4.1 Overview

Please note that the block numbers may vary as they are automatically generated in the Siemens TIA Portal.

Drive specific blocks

Blocks	Page
UDT 862 - Y_SIG7PN_AXIS_CFG - Axis configuration data	➜ *38*
UDT 850 ... UDT 859 - internally used data structures	➜ *38*
FB 841 - Y_ServoFunction - system functions	➜ *39*
FB 847 - Y_ReadSafeState - read safety status	➜ *40*
FB 849 - Y_Init - axis configuration	➜ *41*
FB 862 - Y_SIG7PN_Kernel - Kernel	➜ *45*
FB 863 - Y_SIG7PN_DeviceDriver - internal diagnostics	➜ *45*
FB 865 - Y_SIG7PN_ServoInit - internal initialization	➜ *45*
FB 866 - Y_SIG7PN_ServoOrder - internal job initialization	➜ *45*

Complex motion tasks - PLCopen blocks

Blocks	Page
UDT 860 - Y_SIG7PN_AXIS_REF - Data structure axis data	➜ *46*
UDT 861 - MC_TRIGGER_REF - Data structure trigger signal	➜ *46*
FB 800 - MC_Power - enable/disable axis	➜ *46*
FB 801 - MC_Home - home axis	➜ *48*
FB 802 - MC_Stop - stop axis	➜ *50*
FB 803 - MC_Halt - holding axis	➜ *53*
FB 804 - MC_MoveRelative - move axis relative	➜ *55*
FB 805 - MC_MoveVelocity - drive axis with constant velocity	➜ *57*
FB 808 - MC_MoveAbsolute - move axis to absolute position	➜ *59*
FB 811 - MC_Reset - reset axis	➜ *61*
FB 812 - MC_ReadStatus - read status axis	➜ *63*
FB 813 - MC_ReadAxisError - read axis error	➜ *65*
FB 816 - MC_ReadActualPosition - reading current axis position	➜ *67*
FB 817 - MC_ReadActualVelocity - read axis velocity	➜ *68*
FB 818 - MC_ReadAxisInfo - read additional axis information	➜ *69*
FB 819 - MC_ReadMotionState - read status motion job	➜ *71*
FB 823 - MC_TouchProbe - record axis position	➜ *73*
FB 824 - MC_AbortTrigger - abort recording axis position	➜ *75*

Blocks	Page
FB 833 - Y_ReadParameter - read drive parameter	⇨ *76*
FB 834 - Y_WriteParameter - write drive parameter	⇨ *78*
FB 835 - Y_HomeInit_LimitSwitch - initialisation of homing on limit switch	⇨ *80*
FB 836 - Y_HomeInit_HomeSwitch - initialisation of homing on home switch	⇨ *82*
FB 837 - Y_HomeInit_ZeroPulse - initialisation of homing on zero pulse	⇨ *84*
FB 838 - Y_HomeInit_SetPosition - initialisation of homing mode set position	⇨ *86*
FB 839 - MC_TorqueControl - Move axis with constant torque	⇨ *87*
FB 840 - MC_ReadActualTorque - read actual torque	⇨ *89*

4.2 Drive specific blocks

> *The PLCopen blocks for axis control can be found here: ⇨ Chap. 4 ‘Blocks for axis control’ page 37*

4.2.1 UDT 862 - Y_SIG7PN_AXIS_CFG - Axis configuration data

This is a user defined data structure, that contains configuration data of the axis.

4.2.2 UDT 850 ... UDT 859 - internally used data structures

These are user-defined data structures that are used internally in data structures or blocks. To use the block library, you must copy these UDTs to your project.

4.2.3 FB 841 - Y_ServoFunction - system functions

Description

Here you can specify how the drive parameters should be saved.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ TRUE: Resets all internal axis-related errors.
CmdType	INPUT	INT	■ Temporary input variable to test functions – 2: Load default parameter – 3: Save to non-volatile memory – 4: Software reset – 5: Reset absolute encoder
Mode	INPUT	INT	Mode ■ CmdType: 2: Load default parameters – 1: Load default SERVOPACK and PROFINET parameters – 2: Load default PROFINET parameters – 3: Load default SERVOPACK parameters – ■ CmdType: 3: Save to non-volatile memory – 1: Save SERVOPACK and PROFINET parameters to non-volatile memory – 2: Save PROFINET parameters to non-volatile memory – 3: Save SERVOPACK parameters to non-volatile memory ■ CmdType: 4 and 5: 0 (fix)
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done
Busy	OUTPUT	BOOL	■ Status – TRUE: The FB is not finished and new output values are to be expected.
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information ↬ *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis.

4.2.4 FB 847 - Y_ReadSafeState - read safety status

Description

When using a Yaskawa safety option card on the Sigma-7 drive, you can use this block to get the safety status.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ TRUE: Get the value of the parameter continuously while enabled.
Valid	OUTPUT	BOOL	■ TRUE: A valid set of output values is available in the FB.
Busy	OUTPUT	BOOL	■ TRUE: The FB is not finished and new output values are to be expected.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
SRI_A1_In	OUTPUT	BOOL	■ SRI-A1 input signal – 0: Safety request input signal is ON.
SRI_A2_In	OUTPUT	BOOL	■ SRI-A2 input signal – 0: Safety request input signal is ON.
SRI_B1_In	OUTPUT	BOOL	■ SRI-B1 input signal – 0: Safety request input signal is ON.
SRI_B2_In	OUTPUT	BOOL	■ SRI-B2 input signal – 0: Safety request input signal is ON.
EDM_A_Out	OUTPUT	BOOL	■ EDM-A output signal – 0: External device monitor output signal is OFF.
EDM_B_Out	OUTPUT	BOOL	■ EDM-A output signal – 0: External device monitor output signal is OFF.
Monitoring	OUTPUT	BOOL	■ Safety function - monitoring – 1 = monitoring is in progress.
Safe	OUTPUT	BOOL	■ Safety function - safe state – 1 = safe state
HWBB	OUTPUT	BOOL	■ Safety function - HWBB – 1 = HWBB function is operating
ActiveMode	OUTPUT	BOOL	■ Active mode state – 0 = standby or not selected – 1 = operating
Axis	IN_OUT	STRUCT	Reference to the axis.

4.2.5 FB 849 - Y_Init - axis configuration

Description

This block is used to configure the axis. The block is specially adapted to the use of a Sigma-7 drive, which is connected via PROFINET.

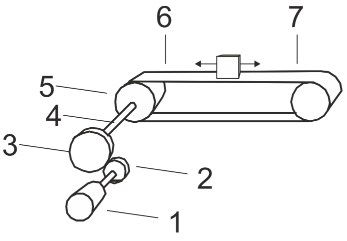
Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Initialize drive
Resolution	INPUT	DINT	Number of increments in [user units]
GearRatioNum	INPUT	DINT	*Gear Ratio* numerator
GearRatioDenom	INPUT	DINT	*Gear Ratio* denominator
FeedConstantNum	INPUT	DINT	Feed per motor revolution *Feed Constant* numerator
FeedConstantDenom	INPUT	DINT	Feed per load revolution *Feed Constant* denominator
AxisType	INPUT	BOOL	■ Configuration of the axis type – FALSE: limited (e.g. conveyor belt with limitation) – TRUE: endless (e.g. rotary table without stop)
MinPosition	INPUT	REAL	SW switch for min. traversing range ■ *Axis type*: limited – Minimum position that can be reached. ■ *Axis type*: endless – Position from which the maximum position value is jumped to.
MaxPosition	INPUT	REAL	SW switch for max. traversing range ■ *Axis type*: limited – Maximum position that can be reached. ■ *Axis type*: endless – Position from which the minimum position value is jumped to.
MinUserPos	OUTPUT	REAL	Minimum user position based on the configured user units [user units].
MaxUserPos	OUTPUT	REAL	Maximum user position based on the configured user units [user units].
MaxUserVelocity	OUTPUT	REAL	Maximum velocity based on the configured user units [user units/s].
MaxUserAcceleration	OUTPUT	REAL	Maximum acceleration based on the configured user units [user units/s^2].
MaxUserTorque	OUTPUT	REAL	Maximum torque/force [1.0% Rated Motor Torque] on motor side, regardless of the configured mechanics.
Done	OUTPUT	BOOL	■ Status – TRUE: Drive is initialized
Busy	OUTPUT	BOOL	■ Status – TRUE: The FB is not ready and new output values are to be expected
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis

Parameter	Declaration	Data type	Description
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found im Parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

Block allocation - conveyor belt

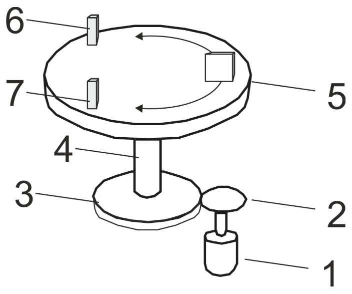
Parameter	Declaration	Data type	Example value	Pos.
Resolution	INPUT	DINT	10 [increments per user unit]	-
GearRatioNum	INPUT	DINT	30 revolutions (motor)	1
GearRatioDenom	INPUT	DINT	12 revolutions (load)	1
FeedConstantNum	INPUT	DINT	314 [user unit]	2
FeedConstantDenom	INPUT	DINT	10 revolutions (load)	3
AxisType	INPUT	BOOL	FALSE (traverse is limited)	-
MinPosition	INPUT	REAL	0.0 [user unit]	6
MaxPosition	INPUT	REAL	1080.0 [user unit]	7



1 Motor
2 Driving cogwheel (12 teeth)
3 Cogwheel driven (30 teeth)
4 Load shaft
5 Conveyor belt (circumference 31.4 mm)
6 Minimum position
7 Maximum position

Block allocation - rotary table with stop

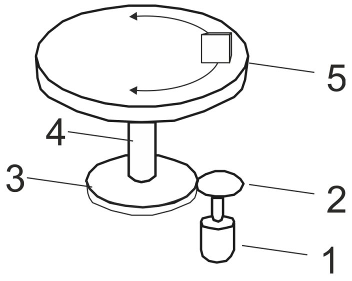
Parameter	Declaration	Data type	Example value	Pos.
Resolution	INPUT	DINT	10 [increments per user unit]	-
GearRatioNum	INPUT	DINT	360 revolutions (motor)	1
GearRatioDenom	INPUT	DINT	24 revolutions (load)	1
FeedConstantNum	INPUT	DINT	360 [user unit]	2
FeedConstantDenom	INPUT	DINT	1 revolution (load)	3
AxisType	INPUT	BOOL	FALSE (traverse is limited)	-
MinPosition	INPUT	REAL	0.0 [user unit]	6
MaxPosition	INPUT	REAL	270.0 [user unit]	7



1 Motor
2 Driving cogwheel (24 teeth)
3 Cogwheel driven (360 teeth]
4 Load shaft
5 Rotary table
6 Minimum position
7 Maximum position

Block allocation - rotary table without stop

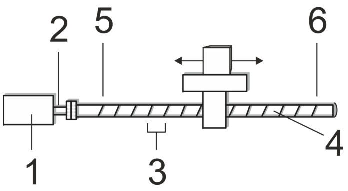
Parameter	Declaration	Data type	Example value	Pos.
Resolution	INPUT	DINT	10 [increments per user unit]	-
GearRatioNum	INPUT	DINT	360 revolutions (motor)	1
GearRatioDenom	INPUT	DINT	24 revolutions (load)	1
FeedConstantNum	INPUT	DINT	360 [user unit]	2
FeedConstantDenom	INPUT	DINT	1 revolution (load)	3
AxisType	INPUT	BOOL	TRUE (traverse is endless)	-
MinPosition	INPUT	REAL	0.0 [user unit]	-
MaxPosition	INPUT	REAL	270.0 [user unit]	-



1 Motor
2 Driving cogwheel (24 teeth)
3 Cogwheel driven (360 teeth]
4 Load shaft
5 Rotary table

Block allocation - ball screw lead

Parameter	Declaration	Data type	Example value	Pos.
Resolution	INPUT	DINT	100 [mm per user unit]	-
GearRatioNum	INPUT	DINT	3 revolutions (motor)	1
GearRatioDenom	INPUT	DINT	1 revolution (load)	1
FeedConstantNum	INPUT	DINT	25 [user unit]	3
FeedConstantDenom	INPUT	DINT	10 revolutions (load)	3
AxisType	INPUT	BOOL	FALSE (traverse is limited)	-
MinPosition	INPUT	REAL	0 [user unit]	5
MaxPosition	INPUT	REAL	1000 [user unit]	6

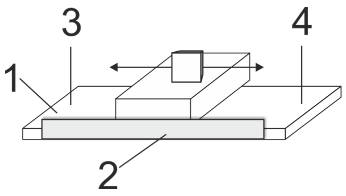


1 Motor with gearbox (3:1)
2 Load shaft
3 Spindle pitch (2.5mm)
4 Ball screw lead
5 Minimum position
6 Maximum position

Block allocation - linear axis

Parameter	Declaration	Data type	Example value	Pos.
Resolution	INPUT	DINT	10 [increments per user unit]	-
GearRatioNum	INPUT	DINT	1 revolution (motor)	see note
GearRatioDenom	INPUT	DINT	1 revolution (load)	
FeedConstantNum	INPUT	DINT	20 [user unit]	see note
FeedConstantDenom	INPUT	DINT	1000 revolutions (load)	
AxisType	INPUT	BOOL	FALSE (traverse is limited)	-
MinPosition	INPUT	REAL	0.0 [user unit]	3
MaxPosition	INPUT	REAL	500.0 [user unit]	4

- *The linear motor requires a gear ratio of 1:1.*
- *In this example application, you have to set the parameter Pn20A ‘Number of External Encoder Pitches’ to 20 [Scale Pitch per revolution] using SigmaWin+ respectively FB 834 - Y_WriteParameter ↪ 78.*



1 Linear motor
2 Length measuring system
3 Minimum position
4 Maximum position

4.2.6 FB 862 - Y_SIG7PN_Kernel - Kernel

Description This block converts the drive commands for a Sigma-7 axis via PROFINET. The block is called cyclically within FB 200.

Parameter

Parameter	Declaration	Data type	Description
DPV1_ID	INPUT	DWORD	DPV1 DIAG
DIAG_BUF	INPUT	BOOL	Diagnostic Buffer ■ TRUE = CPU Diagnostic buffer entry on – relevant for service personnel
Axis	IN_OUT	STRUCT	Reference to the axis

4.2.7 FB 863 - Y_SIG7PN_DeviceDriver - internal diagnostics

Description This block is used internally.

4.2.8 FB 865 - Y_SIG7PN_ServoInit - internal initialization

Description This block is used internally.

4.2.9 FB 866 - Y_SIG7PN_ServoOrder - internal job initialization

Description This block is used internally.

4.3 Complex motion tasks - PLCopen blocks

4.3.1 UDT 860 - Y_SIG7PN_AXIS_REF - Data structure axis data

This is a user-defined data structure that contains status information of the axis.

4.3.2 UDT 861 - MC_TRIGGER_REF - Data structure trigger signal

This is a user defined data structure, that contains information of the trigger signal.

4.3.3 FB 800 - MC_Power - enable/disable axis

Description

With MC_Power an axis can be enabled or disabled.

> *Please note that as long as the drive has not yet been initialised, in particular during commissioning, the FB 800 - MC_Power can trigger error message 0x8103. Calling FB 849 - Y_Init this error is automatically reset. Resetting the error with FB 811 - MC_Reset is not possible.*

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Enable/disable axis – TRUE: The axis is enabled – FALSE: The axis is disabled
Status	OUTPUT	BOOL	■ Status axis – TRUE: The axis is ready to execute motion control jobs – FALSE: The axis is not ready to execute motion control jobs
Valid	OUTPUT	BOOL	Always FALSE
Error	OUTPUT	BOOL	■ Error – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*. The axis is disabled.
ErrorID	OUTPUT	WORD	Additional error information ↳ *Chap. 6 ‘ErrorID - Additional error information’ page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

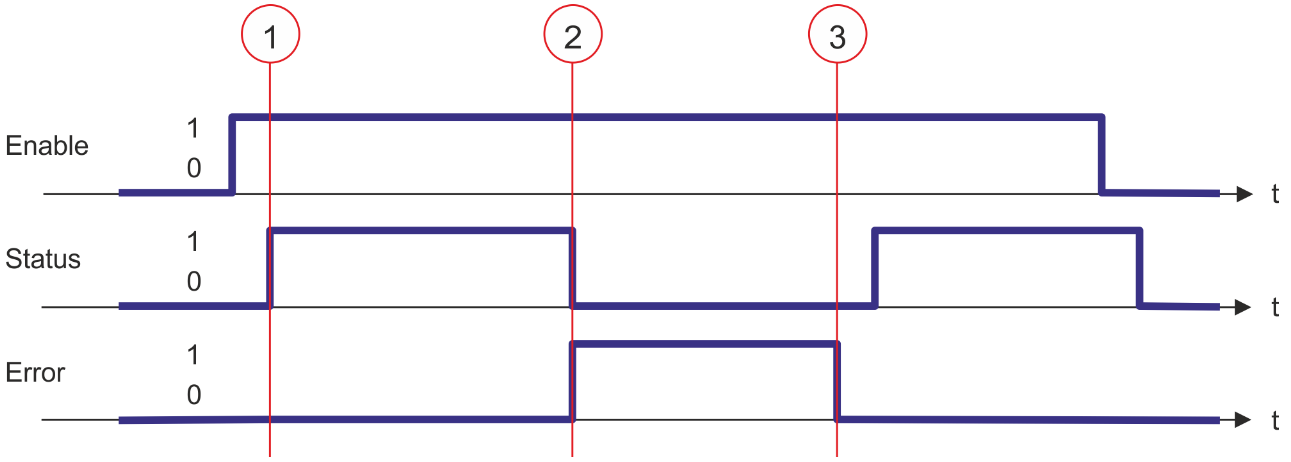
Enable axis

Call MC_Power with *Enable* = TRUE. If *Status* shows the value TRUE, the axis is enabled. In this status motion control jobs can be activated.

Disable axis

Call MC_Power with *Enable* = FALSE. If *Status* shows the value FALSE, the axis is disabled. When disabling the axis a possibly active motion job is cancelled and the axis is stopped.

Status diagram of the block parameters



(1) The axis is enabled with *Enable* = TRUE. At the time (1) it is enabled. Then motion control jobs can be activated.
(2) At the time (2) an error occurs, which causes the disabling of the axis. A possibly active motion job is cancelled and the axis is stopped.
(3) The error is eliminated and acknowledged at time (3). Thus *Enable* is further set, the axis is enabled again. Finally the axis is disabled with *Enable* = FALSE.

4.3.4 FB 801 - MC_Home - home axis

Description

With MC_Home an axis can be set to a reference point. This is used to match the axis coordinates to the real, physical drive position. The homing method and its parameters must be configured directly at the drive. For this use the Y_HomeInit_... blocks. To do this, use one of the Y_HomeInit _... blocks. As soon as *Done* of the Y_HomeInit _... block signals TRUE, FB 801 - MC_Home is to be called.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Homing – Edge 0-1: Homing is started
Position	INPUT	REAL	With a successful homing the current position of the axis is uniquely set to *Position*. *Position* is to be entered in the used user unit.
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information ☞ *Chap. 6 ‘ErrorID - Additional error information’ page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

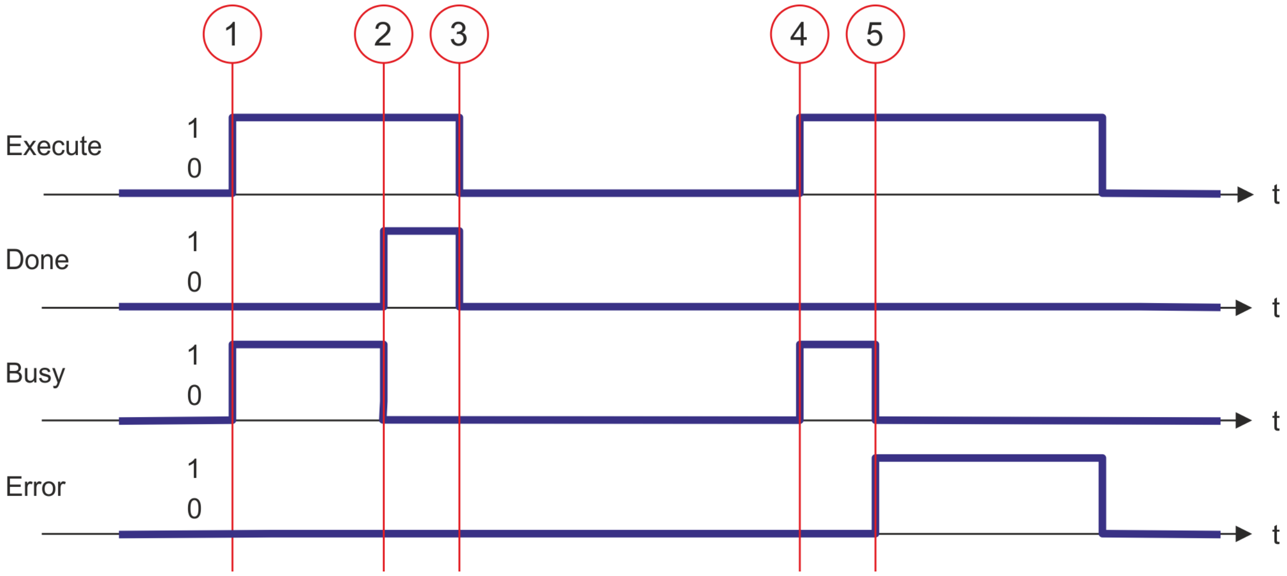
Start of the job only in the PLCopen-State *Standstill* possible. ☞ *Chap. 5.1 ‘PLCopen-States’ page 90*

Home axis

The homing is started with edge 0-1 at *Execute*. *Busy* is TRUE as soon as the homing is running. Once *Done* becomes TRUE, homing was successfully completed. The current position of the axis was set to the value of *Position*.

- *An active job continues to run even when Execute is set to FALSE.*
- *A running job can not be aborted by a move job (e.g. MC_MoveRelative).*

Status diagram of the block parameters



(1) The homing is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.
(2) At the time (2) the homing is completed. *Busy* has the value FALSE and *Done* the value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.
(4) At the time (4) with an edge 0-1 at *Execute* the homing is started again and *Busy* becomes TRUE.
(5) At the time (5) an error occurs during homing. *Busy* has the value FALSE and *ERROR* the value TRUE.

4.3.5 FB 802 - MC_Stop - stop axis

Description

With MC_Stop the axis is stopped. With the parameter *Deceleration*, the dynamic behavior can be determined during stopping.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Stop axis – Edge 0-1: Stopping of the axis is started
Deceleration	INPUT	REAL	Delay in stopping in [user units/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

- Start of the job in the PLCopen-States *Standstill*, *Homing*, *Discrete Motion* and *Continuous Motion* possible.
- MC_Stop switches the axis to the PLCopen-State *Stopping*. In *Stopping* no motion jobs can be started. As long as *Execute* is true, the axis remains in PLCopen-State *Stopping*. If *Execute* becomes FALSE, the axis switches to PLCopen-State*Standstill*. In *Standstill* motion tasks can be started.
- *Chap. 5.1 'PLCopen-States' page 90*

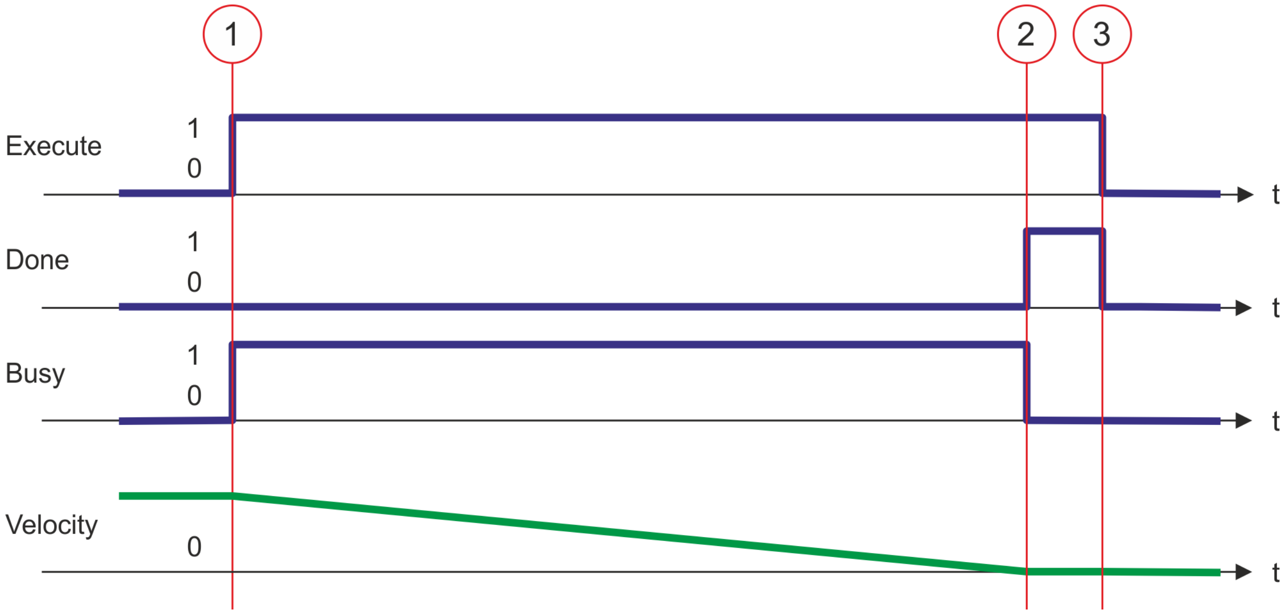
Stop axis

The stopping of the axis is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as the stopping of the axis is running. After the axis has been stopped and thus the speed has reached 0, *Busy* with FALSE and *Done* with TRUE is returned.

– *An active job continues until the axis stops even when Execute is set to FALSE.*

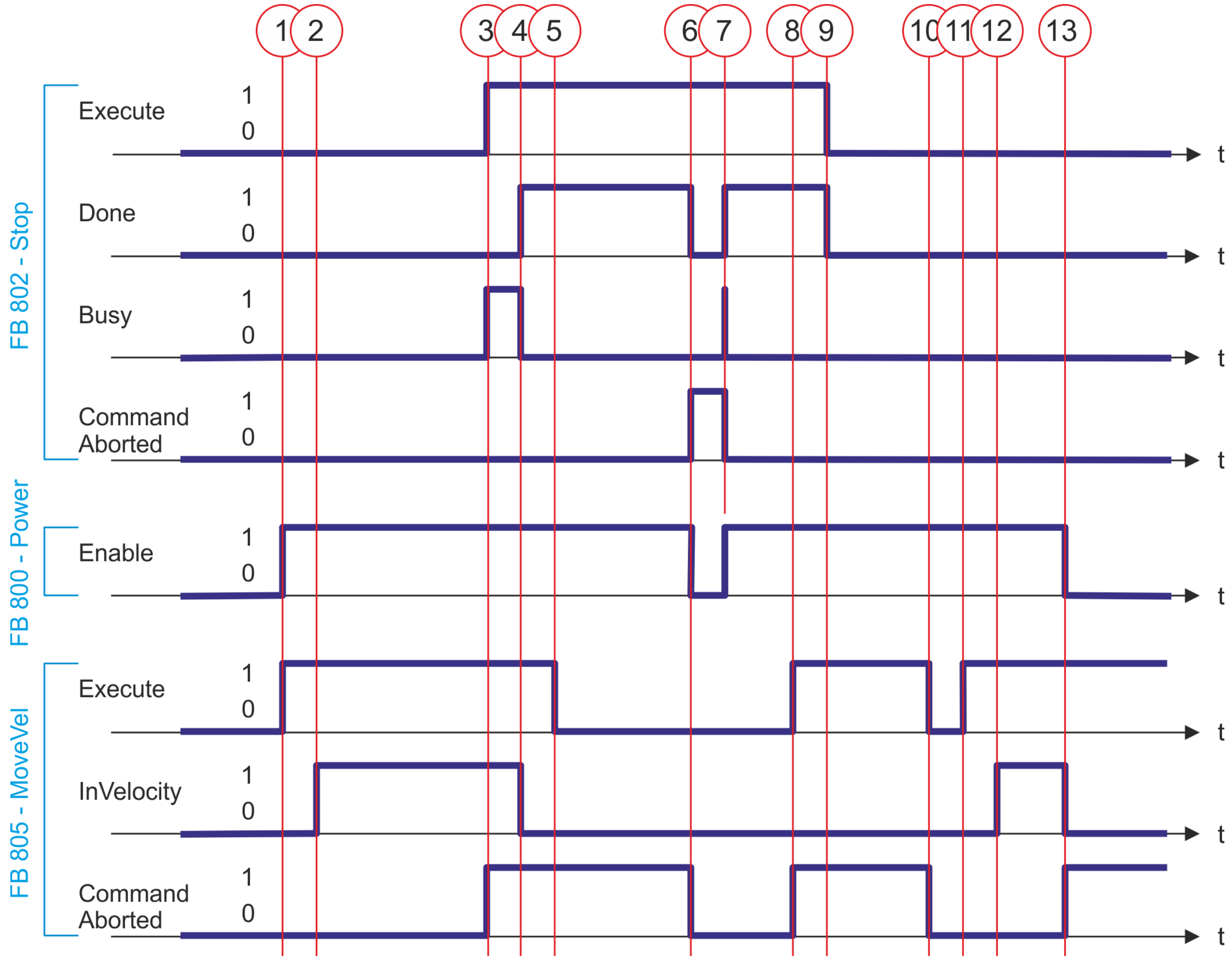
– *A running job can not be aborted by a move job (e.g. MC_MoveRelative).*

Status diagram of the block parameters



(1) Stopping of the axis is started with edge 0-1 at *Execute* and *Busy* becomes TRUE. The velocity of the axis is reduced to zero, regarding the parameter *Deceleration*.
(2) At time (2) stopping the axis is completed, the axis is stopped. *Busy* has the value FALSE and *Done* the value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

Behavior on return of axis release



(1) With edge 0-1 at *Enable* of FB 800 the axis is released. With edge 0-1 at *Enable* of FB 805 the movement of the axis is started.

(2) The axis reaches the velocity specification and *InVelocity* returns TRUE.

(3) With edge 0-1 at *Enable* of FB 802 the stopping of the axis is started and *Busy* returns TRUE. The axis is decelerated to standstill. *CommandAborted* of FB 805 returns TRUE.

(4) The stopping of the axis is completed, the axis is stopped. *Busy* returns FALSE and *Done* TRUE. Due to the axis is stopped, *InVelocity* of FB 805 returns FALSE.

(5) The FB 805 is terminated.

(6) *Enable* of FB 800 is set to FALSE and the axis is disabled. This sets *CommandAborted* of all the blocks to TRUE and *Done* of FB 802 to FALSE.

(7) With edge 0-1 at *Enable* of FB 800 the axis is released again. This causes *CommandAborted* of FB 802 to return FALSE and *Busy* a short time TRUE.

(8) With edge 0-1 at *Enable* of FB 805 the movement of the axis should be started. The active FB 802 prevents this and *CommandAborted* returns TRUE.

(9) The FB 802 is terminated. The movement of the axis can only be started by an edge 0-1 of *Execute* of FB 805.

(10) The FB 805 is terminated and *CommandAborted* returns FALSE.

(11) With edge 0-1 at *Enable* of FB 805 the movement of the axis is started.

(12) The axis reaches the specified velocity and *InVelocity* returns TRUE.

(13) *Enable* of FB 800 is set to FALSE and the axis is disabled. This sets *CommandAborted* of all the blocks to TRUE.

4.3.6 FB 803 - MC_Halt - holding axis

Description

With MC_Halt the axis is slowed down to standstill. With the parameter *Deceleration* the dynamic behavior can be determined during breaking.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Stop axis – Edge 0-1: Stopping of the axis is started
Deceleration	INPUT	REAL	Delay in breaking in [user units/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information ☞ *Chap. 6 ‘ErrorID - Additional error information’ page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

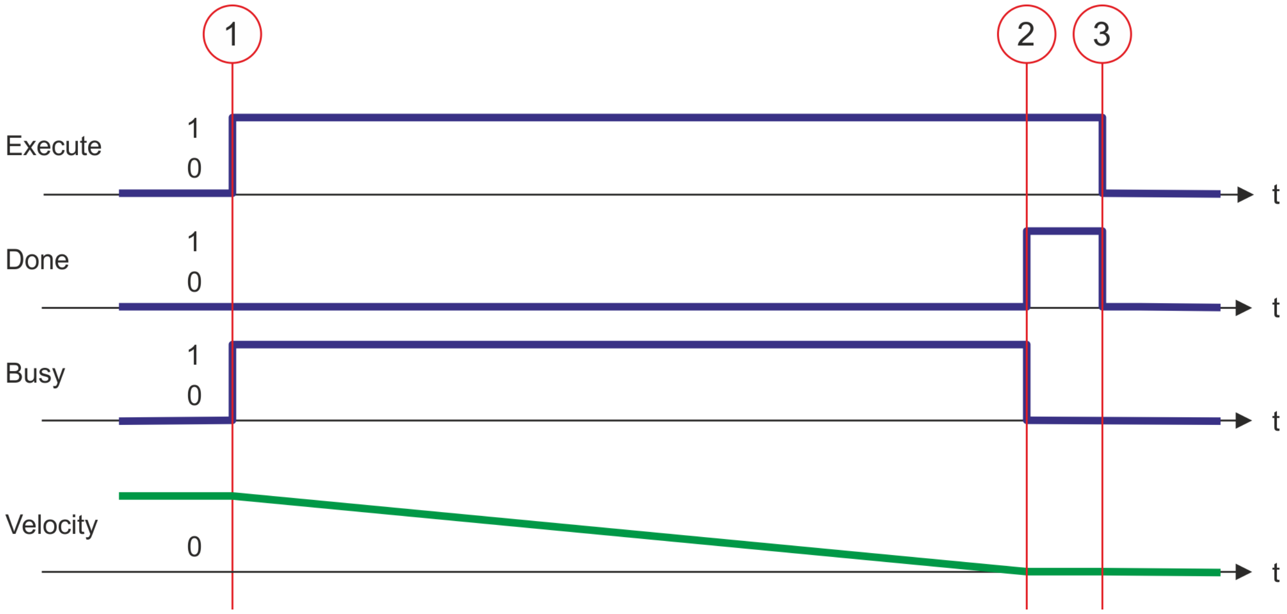
- Start of the job in the PLCopen-States *Discrete Motion* and *Continuous Motion* possible.
- MC_Halt switches the axis to the PLCopen-State *Discrete Motion*.
- ☞ *Chap. 5.1 ‘PLCopen-States’ page 90*

Slow down axis

The slow down of the axis is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as the slow down of the axis is running. After the axis has been slowed down and thus the speed has reached 0, *Busy* with FALSE and *Done* with TRUE is returned.

> – *An active halt job continues until the axis stops even when Execute is set to FALSE.*
>
> – *A running halt job can be aborted by a move job (e.g. MC_MoveRelative).*

Status diagram of the block parameters



(1) Breaking the axis is started with edge 0-1 at *Execute* and *Busy* becomes TRUE. The velocity of the axis is reduced to zero, regarding the parameter *Deceleration*.
(2) At time (2) slowing down the axis is completed, the axis is stopped. *Busy* has the value FALSE and *Done* the value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.7 FB 804 - MC_MoveRelative - move axis relative

Description

With MC_MoveRelative the axis is moved relative to the position in order to start a specified distance. With the parameters *Velocity*, *Acceleration* and *Deceleration* the dynamic behavior can be determined during the movement.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Move axis relative – Edge 0-1: The relative movement of the axis is started
Distance	INPUT	REAL	Relative distance in [user units]
Velocity	INPUT	REAL	Max. Velocity (needs not necessarily be reached) in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Deceleration	INPUT	REAL	Delay in breaking in [user units/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done; target position reached
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

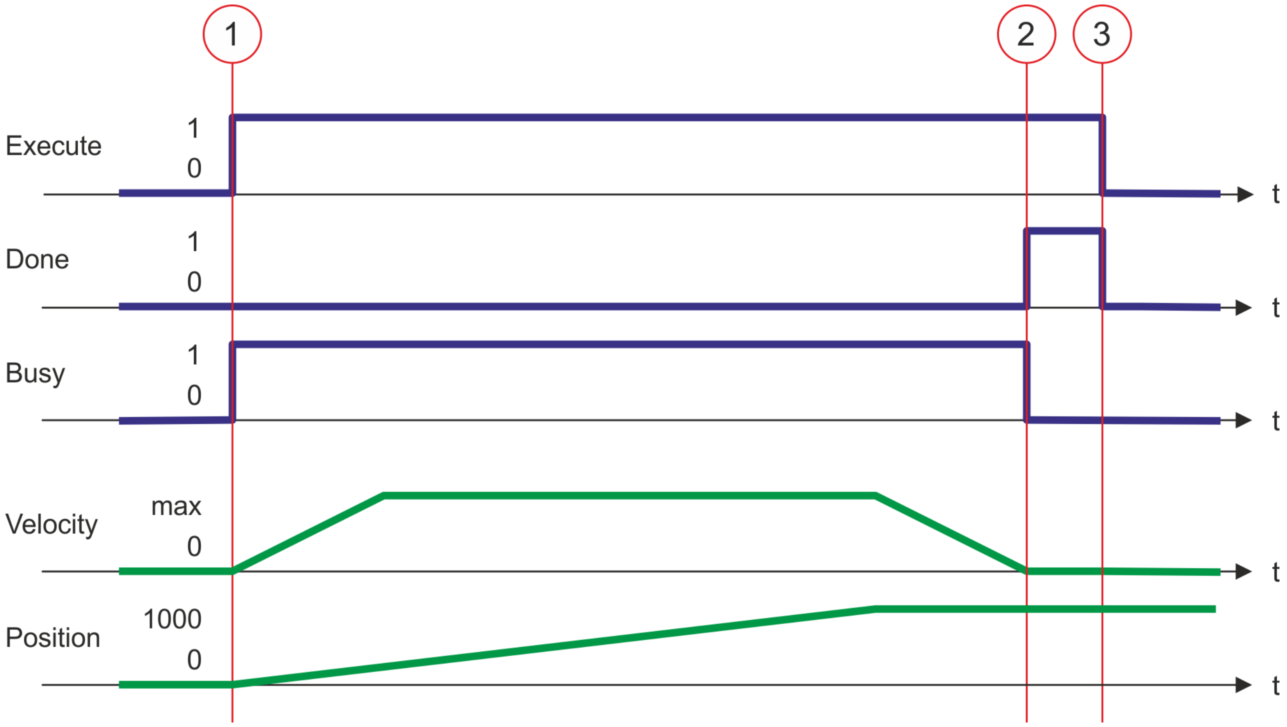
- Start of the job in the PLCopen-States *Standstill*, *Discrete Motion* and *Continuous Motion* possible.
- MC_MoveRelative switches the axis to the PLCopen-State *Discrete Motion*.
- *Chap. 5.1 'PLCopen-States' page 90*

Move axis relative

The movement of the axis is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as the movement of the axis is running. After the target position was reached, *Busy* with FALSE and *Done* with TRUE is returned. Then the velocity of the axis is 0.

- *An active job continues to move to target position even when Execute is set to FALSE.*
- *A running job can be aborted by a move job (e.g. MC_MoveAbsolute).*

Status diagram of the block parameters



(1) With MC_MoveRelative the axis is moved relative by a *Distance* = 1000.0 (start position at job start is 0.0). Moving the axis is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.

(2) At time (2) the axis was moved by the *Distance* = 1000.0, i.e. the target position was reached. *Busy* has the value FALSE and *Done* the value TRUE.

(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.8 FB 805 - MC_MoveVelocity - drive axis with constant velocity

Description

With MC_MoveVelocity the axis is driven with a constant velocity. With the parameters *Velocity*, *Acceleration* and *Deceleration* the dynamic behavior can be determined during the movement.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Drive axis with constant velocity – Edge 0-1: Drive axis with constant velocity is started
Velocity	INPUT	REAL	Velocity setting in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Deceleration	INPUT	REAL	Delay in breaking in [user units/s^2]
InVelocity	OUTPUT	BOOL	■ Velocity setting – TRUE: Velocity setting reached
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

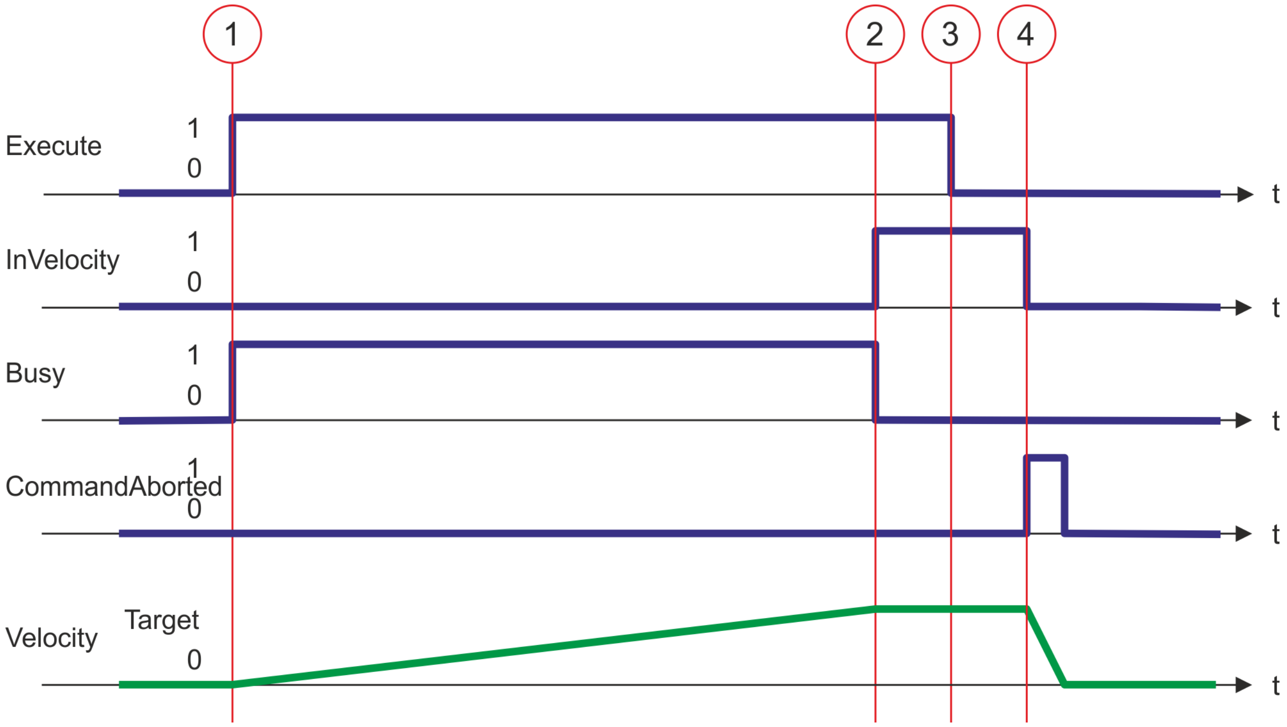
- Start of the job in the PLCopen-States *Standstill*, *Discrete Motion* and *Continuous Motion* possible.
- MC_MoveVelocity switches the axis to the PLCopen-State *Continuous Motion*.
- *Chap. 5.1 'PLCopen-States' page 90*

Drive axis with set velocity

The movement of the axis with set velocity is started with an edge 0-1 at *Execute*. *Busy* is TRUE and *InVelocity* FALSE as soon as the set velocity is not reached. If the set velocity is reached, *Busy* becomes FALSE and *InVelocity* TRUE. The axis is constant moved with this velocity.

- *An active job is continued, even when the set velocity is reached and even when Execute is set to FALSE.*
- *A running job can be aborted by a move job (e.g. MC_MoveAbsolute).*

Status diagram of the block parameters



(1) Moving the axis with set velocity is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.

(2) At time (2) the axis reaches the set velocity and *Busy* has the value FALSE and *InVelocity* the value TRUE.

(3) Resetting *Execute* to FALSE at time (3) does not influence the axis. The axis is further moved with constant set velocity and *InVelocity* is further TRUE.

(4) At the time (4) the MC_Velocity job is aborted by a MC_Stop job. The axis is decelerated to standstill.

4.3.9 FB 808 - MC_MoveAbsolute - move axis to absolute position

Description

With MC_MoveAbsolute the axis is moved to an absolute position. With the parameters *Velocity*, *Acceleration* and *Deceleration* the dynamic behavior can be determined during the movement.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Move the axis – Edge 0-1: The movement of the axis is started
Position	INPUT	REAL	Absolute position in [user units]
Velocity	INPUT	REAL	Maximum velocity (needs not necessarily be reached) in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Deceleration	INPUT	REAL	Delay in breaking in [user units/s^2]
Direction	INPUT	Byte	■ Direction – 0: Shortest way – 1: Positive direction – 2: Negative direction
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Target position was reached.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

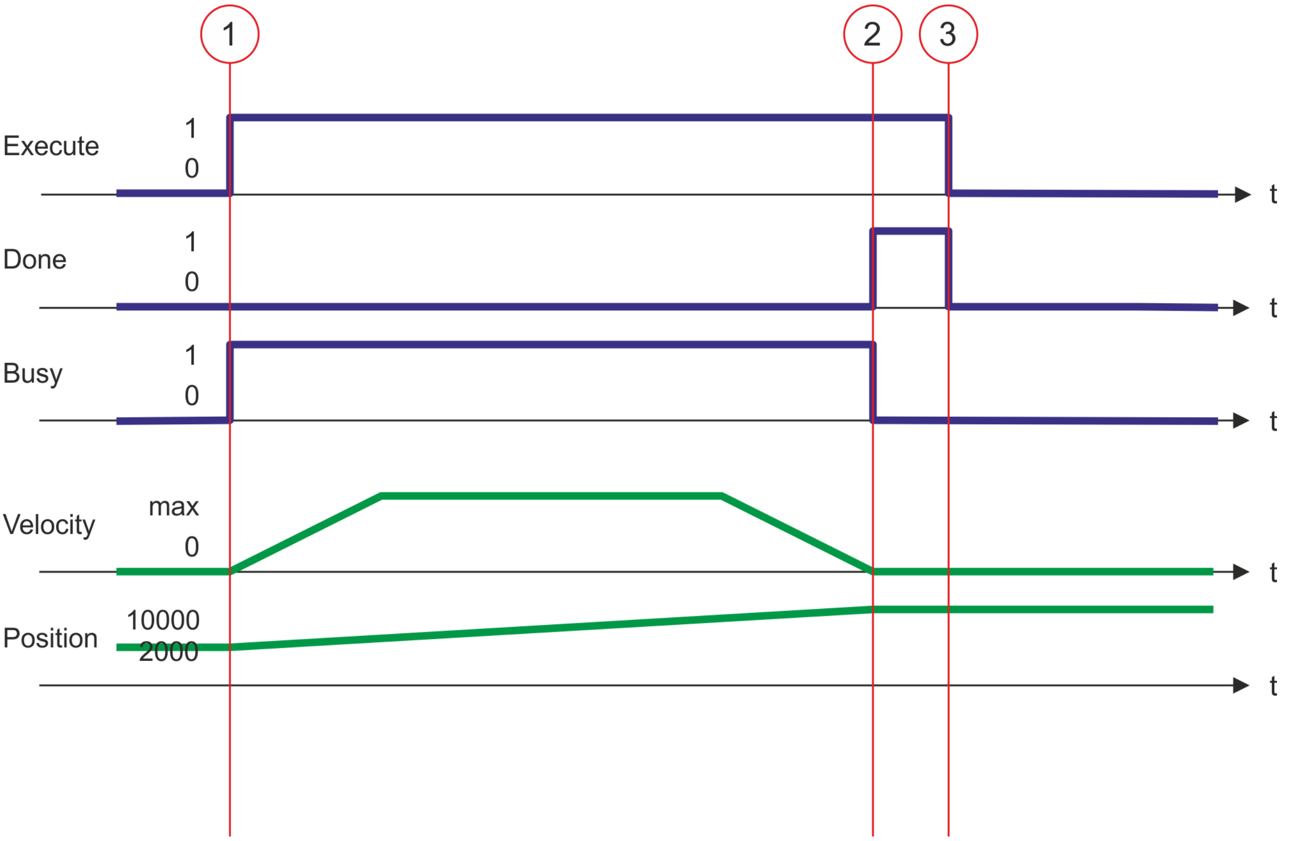
- Start of the job in the PLCopen-States *Standstill*, *Discrete Motion* and *Continuous Motion* possible.
- MC_MoveAbsolute switches the axis to the PLCopen-State *Discrete Motion*.
- *Chap. 5.1 'PLCopen-States' page 90*

Move axis absolute

The movement of the axis is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as the movement of the axis is running. After the target position was reached, *Busy* with FALSE and *Done* with TRUE is returned. Then the velocity of the axis is 0.

- *An active job continues to move to target position even when Execute is set to FALSE.*
- *A running job can be aborted by a move job (e.g. MC_MoveVelocity).*

Status diagram of the block parameters



(1) With MC_MoveAbsolute the axis is moved to the absolute position = 10000.0 (start position at job start is 2000.0). At time (1) moving the axis is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.

(2) At time (2) the axis has reached the target position. *Busy* has the value FALSE and *Done* the value TRUE.

(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.10 FB 811 - MC_Reset - reset axis

Description

With MC_Reset a reset (reinitialize) of the axis can be done. Here all the internal errors are reset.

> *Please note that as long as the drive has not yet been initialised, in particular during commissioning, the FB 800 - MC_Power can trigger error message 0x8103. Calling FB 849 - Y_Init this error is automatically reset. Resetting the error with FB 811 - MC_Reset is not possible.*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Reset axis – Edge 0-1: Axis reset is performed
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Reset was performed
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 ‘ErrorID - Additional error information’ page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

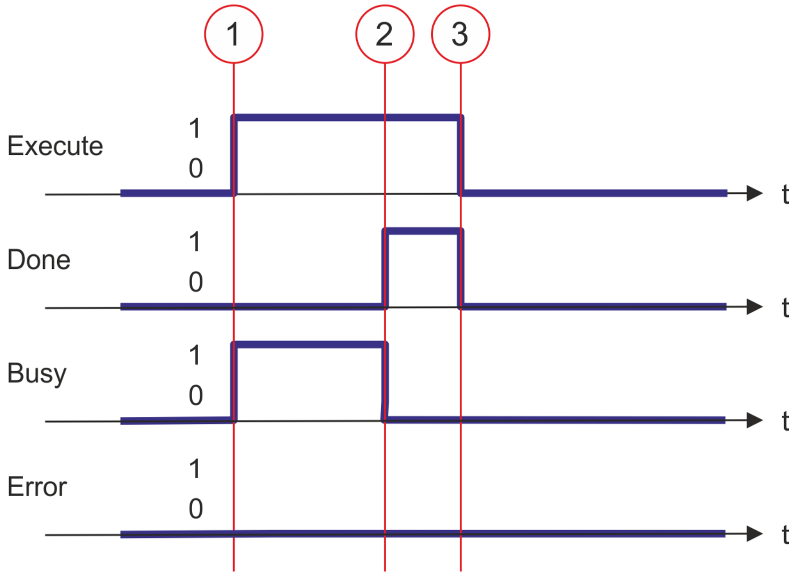
- Job start in PLCopen-State *ErrorStop*, *Disabled* and *Standstill* possible. Following messages can be reset:
  - Warnings
  - Errors
- MC_Reset switches the axis depending on MC_Power either to PLCopen-State *Standstill* (call MC_Power with *Enable* = TRUE) or *Disabled* (call MC_Power with *Enable* = FALSE).
- *Chap. 5.1 ‘PLCopen-States’ page 90*

Perform reset on axis

The reset of the axis is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as the reset of the axis is running. After axis has been reinitialized, *Busy* with FALSE and *Done* with TRUE is returned.

> *An active job continues until it is finished even when Execute is set to FALSE.*

Status diagram of the block parameters



(1) At time (1) the reset of the axis is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.
(2) At the time (2) the reset is successfully completed. *Busy* has the value FALSE and *Done* the value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.11 FB 812 - MC_ReadStatus - read status axis

Description

With MC_ReadStatus the PLCopen-State of the axis can be determined

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Status indication – TRUE: The status is permanently displayed at the outputs – FALSE: All the outputs are FALSE respectively 0
Valid	OUTPUT	BOOL	■ State is valid – TRUE: The shown state is valid
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information 🖙 *Chap. 6 ‘ErrorID - Additional error information’ page 94*
Disabled	OUTPUT	BOOL	■ Status axis: Disabled – TRUE: Axis is disabled, move job can not be activated
Standstill	OUTPUT	BOOL	■ Status move job – TRUE: No move job is active; a move job can be activated
Homing	OUTPUT	BOOL	■ Status axis: Homing – TRUE: Axis is just homing (MC_Homing is active)
DiscreteMotion	OUTPUT	BOOL	■ Status axis motion: Discrete – TRUE: Axis is moved by a discrete movement (MC_MoveRelative, MC_MoveAbsolute or MC_Halt is active)
ContinuousMotion	OUTPUT	BOOL	■ Status axis motion: Continuous – TRUE: Axis is moved by a continuous movement (MC_MoveVelocity is active)
SynchronizedMotion	OUTPUT	BOOL	This parameter is currently not used.
Stopping	OUTPUT	BOOL	■ Status axis: Stop – TRUE: Axis is stopped (MC_Stop is active)
ErrorStop	OUTPUT	BOOL	■ Axis errors – TRUE: An axis error has occurred, move job can not be activated
Axis	IN_OUT	STRUCT	Reference to the slave axis

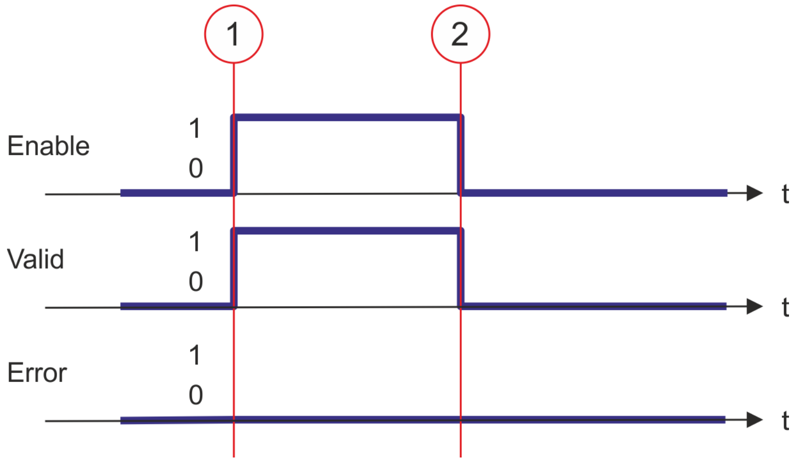
PLCopen-State

- Job start in each PLCopen-State possible.
- 🖙 *Chap. 5.1 ‘PLCopen-States’ page 90*

Determine the status of the axis

With *Enable* = TRUE the outputs represent the state of the axis according to the PLCopen-State diagram.

Status diagram of the block parameters



(1) At time (1) *Enable* is set to TRUE. So *Valid* gets TRUE and the outputs correspond to the status of the PLCopen-State.
(2) At time (2) *Enable* is set to FALSE. So all the outputs are set to FALSE respectively 0.

4.3.12 FB 813 - MC_ReadAxisError - read axis error

Description

With MC_ReadAxisError the current error of the axis is directly be read.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Status – TRUE: The FB continuously returns output data.
Valid	OUTPUT	BOOL	■ Status – TRUE: Valid output data are available in the FB.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
AxisErrorID	OUTPUT	WORD	Axis error ID; the read value is vendor-specifically encoded.
AxisWarningID	OUTPUT	WORD	Axis warning ID; the read value is vendor-specifically encoded.
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

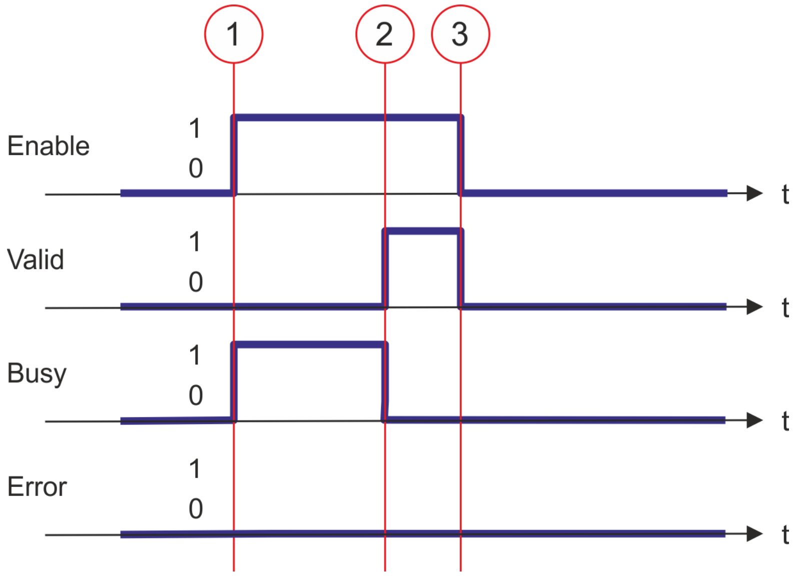
- Job start in each PLCopen-State possible.
- *Chap. 5.1 'PLCopen-States' page 90*

Read error of the axis

The reading of the error of the axis is started with an edge 0-1 at *Enable*. *Busy* is TRUE as soon as reading of the axis error is running. After the axis error was read, *Busy* with FALSE and *Valid* with TRUE is returned. The output *AxisErrorID* shows the current axis error.

An active job continues to run even when Enable is set to FALSE.

Status diagram of the block parameters



(1) At time (1) the reading of the axis error is started with edge 0-1 at *Enable* and *Busy* becomes TRUE.
(2) At the time (2) reading of the axis error is successfully completed. *Busy* has the value FALSE and *Valid* the value TRUE.
(3) At the time (3) the job is completed and *Enable* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.13 FB 816 - MC_ReadActualPosition - reading current axis position

Description

With MC_ReadActualPosition the current position of the axis is read.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Read axis position – TRUE: The position of the axis is continuously read – FALSE: All the outputs are FALSE respectively 0
Valid	OUTPUT	BOOL	■ Position valid – TRUE: The read position is valid
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 ‘ErrorID - Additional error information’ page 94*
Position	OUTPUT	REAL	Absolute position of the axis in [user units]
Axis	IN_OUT	STRUCT	Reference to the axis

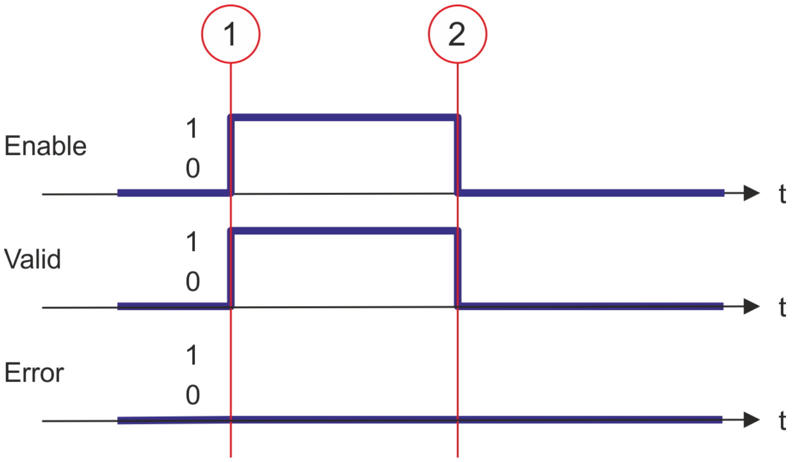
PLCopen-State

- Job start in each PLCopen-State possible.
- *Chap. 5.1 ‘PLCopen-States’ page 90*

Read axis position

The current axis position is determined and stored at *Position* with *Enable* set to TRUE.

Status diagram of the block parameters



(1) At time (1) *Enable* is set to TRUE. So *Valid* gets TRUE and output *Position* corresponds to the current axis position.
(2) At time (2) *Enable* is set to FALSE. So all the outputs are set to FALSE respectively 0.

4.3.14 FB 817 - MC_ReadActualVelocity - read axis velocity

Description

With MC_ReadActualVelocity the current velocity of the axis is read.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Read axis velocity – TRUE: The velocity of the axis is continuously read – FALSE: All the outputs are FALSE respectively 0
Valid	OUTPUT	BOOL	■ Velocity valid – TRUE: The read velocity is valid
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Velocity	OUTPUT	REAL	Actual velocity of the axis in [user units/s]
Axis	IN_OUT	STRUCT	Reference to the axis

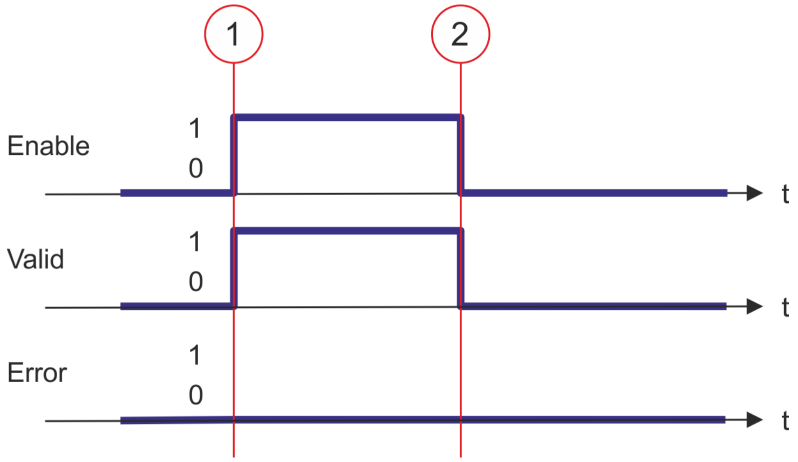
PLCopen-State

- Job start in each PLCopen-State possible.
- *Chap. 5.1 'PLCopen-States' page 90*

Read axis velocity

The current axis velocity is determined and stored at *Velocity* with *Enable* set to TRUE.

Status diagram of the block parameters



(1) At time (1) *Enable* is set to TRUE. So *Valid* gets TRUE and output *Velocity* corresponds to the current axis velocity.
(2) At time (2) *Enable* is set to FALSE. So all the outputs are set to FALSE respectively 0.

4.3.15 FB 818 - MC_ReadAxisInfo - read additional axis information

Description

With MC_ReadAxisInfo some additional information of the axis are shown.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Read additional information from axis – TRUE: The additional information of the axis are read – FALSE: All the outputs are FALSE respectively 0
Valid	OUTPUT	BOOL	■ Additional information valid – TRUE: The read additional information are valid
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information ↳ *Chap. 6 'ErrorID - Additional error information' page 94*
HomeAbsSwitch	OUTPUT	BOOL	Homing switch ■ TRUE: Homing switch is activated
LimitSwitchPos	OUTPUT	BOOL	Limit switch positive direction ■ TRUE: Limit switch positive direction is activated
LimitSwitchNeg	OUTPUT	BOOL	Limit switch negative direction (NOT bit of the drive) ■ TRUE: Limit switch negative direction is activated
Simulation	OUTPUT	BOOL	Parameter is currently not supported; always FALSE
CommunicationReady	OUTPUT	BOOL	■ Information axis: Data exchange – TRUE: Data exchange with axis is initialized; axis is ready for communication
ReadyForPowerOn	OUTPUT	BOOL	■ Information axis: Enable possible – TRUE: Enabling the axis is possible
PowerOn	OUTPUT	BOOL	■ Information axis: Enabled – TRUE: Enabling of the axis is carried out
IsHomed	OUTPUT	BOOL	■ Information axis: Homed – TRUE: The axis is homed
AxisWarning	OUTPUT	BOOL	■ Information axis: Error – TRUE: At least 1 error is reported from the axis
SwLimitSwitchPos	OUTPUT	BOOL	■ Software limit switch positive direction – TRUE: Software limit switch positive direction is activated.
SwLimitSwitchNeg	OUTPUT	BOOL	■ Software limit switch negative direction – TRUE: Software limit switch negative direction is activated.
TorqueLimit	OUTPUT	BOOL	■ Information axis: Torque limit – TRUE: Torque limit is activated.

Parameter	Declaration	Data type	Description
SafetyActive	OUTPUT	BOOL	■ Information axis: Safety state – TRUE: HWBB or safety function is activated.
Axis	IN_OUT	STRUCT	Reference to the axis

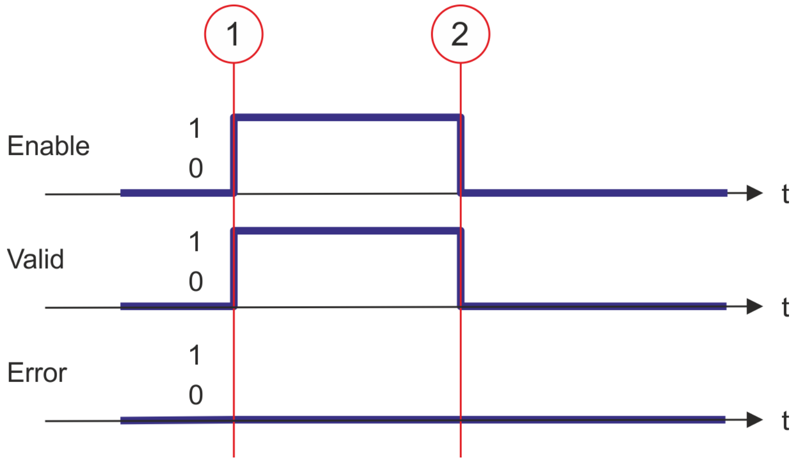
PLCopen-State

- Job start in each PLCopen-State possible.
- ⇨ *Chap. 5.1 'PLCopen-States' page 90*

Determine the status of the axis

The additional information of the axis are shown at the outputs with *Enable* set to TRUE.

Status diagram of the block parameters



(1) At time (1) *Enable* is set to TRUE. So *Valid* gets TRUE and the outputs show the additional information of the axis.
(2) At time (2) *Enable* is set to FALSE. So all the outputs are set to FALSE respectively 0.

4.3.16 FB 819 - MC_ReadMotionState - read status motion job

Description

With MC_ReadMotionState the current status of the motion job is shown.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Read motion state – TRUE: The status of the motion job is continuously read – FALSE: All the outputs are FALSE respectively 0
Source	INPUT	Byte	Only Source = 0 is supported; at the outputs the current status of the motion job is shown.
Valid	OUTPUT	BOOL	■ Status valid – TRUE: The read status of the motion job is valid
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
ConstantVelocity	OUTPUT	BOOL	■ Status motion job: Velocity – TRUE: Velocity is constant
Accelerating	OUTPUT	BOOL	Please note that this parameter is not supported when using inverter drives via EtherCAT! ■ Status motion job: Acceleration – TRUE: The axis is accelerated; the velocity of the axis is increasing
Decelerating	OUTPUT	BOOL	Please note that this parameter is not supported when using inverter drives via EtherCAT! ■ Status motion job: Braking process – TRUE: Axis is decelerated; the velocity of the axis is getting smaller
DirectionPositive	OUTPUT	BOOL	■ Status motion job: Position increasing – TRUE: The position of the axis is increasing
DirectionNegative	OUTPUT	BOOL	■ Status motion job: Position decreasing – TRUE: The position of the axis is decreasing
Axis	IN_OUT	STRUCT	Reference to the axis

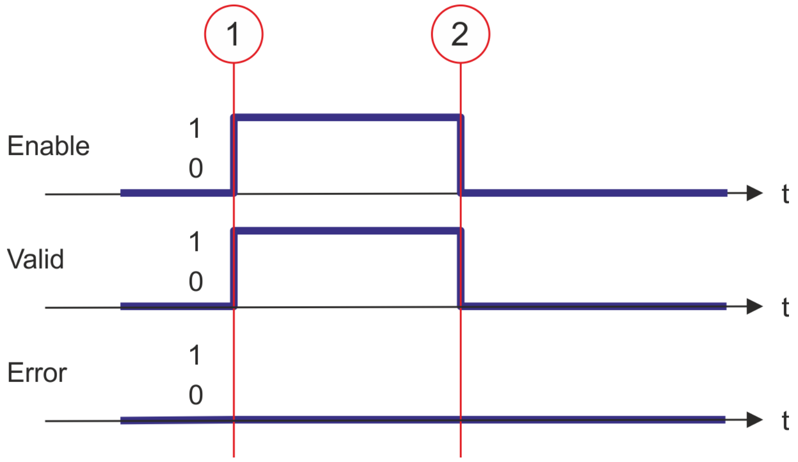
PLCopen-State

- Job start in each PLCopen-State possible.
- *Chap. 5.1 'PLCopen-States' page 90*

Read status of the motion job

With *Enable* = TRUE the outputs represent the status of the motion job of the axis.

Status diagram of the block parameters



(1) At time (1) *Enable* is set to TRUE. So *Valid* gets TRUE and the outputs correspond to the status of motion job.
(2) At time (2) *Enable* is set to FALSE. So all the outputs are set to FALSE respectively 0.

4.3.17 FB 823 - MC_TouchProbe - record axis position

Description

This function block is used to record an axis position at a trigger event. The trigger signal can be configured via the variable specified at the input *TriggerInput*. As trigger signal can serve e.g. a digital input or a encoder zero track.

Parameter

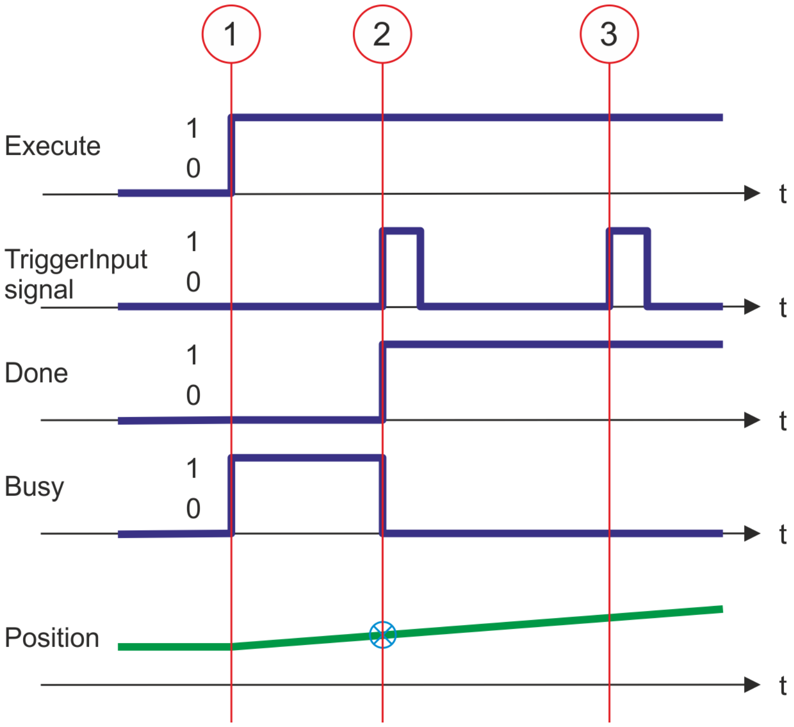
Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	The recording of the axis position is activated with edge 0-1 at *Execute*.
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. The axis position was recorded.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information ↳ *Chap. 6 'ErrorID - Additional error information' page 94*
RecordedPosition	OUTPUT	REAL	Recorded axis position where trigger event occurred in [user units].
Axis	IN_OUT	STRUCT	Reference to the axis.
TriggerInput	IN_OUT	STRUCT	Reference to the trigger input. Structure ■ .Probe – 01: TouchProbe register 1 – 02: TouchProbe register 2 ■ .TriggerSource – 00: Input – 01: Encoder zero pulse ■ .TriggerMode – 00: SingleTrigger ■ .Reserved

- *An active job continues to run until this is completed, even when Execute is set to FALSE. The detected axis position is the output at RecordedPosition for one cycle.* ↳ *Chap. 5.2 'Behavior of the inputs and outputs' page 92*
- *Thus the job can be executed, the communication to the axis must be OK and the PLCopen-State must be unequal Homing.*
- *A running job can be aborted with a new MC_TouchProbe job for the same axis.*
- *A running job can be aborted by MC_AbortTrigger.*
- *A running job can be aborted by MC_Home.*

Recording the axis position

The recording of the axis position is activated with edge 0-1 at *Execute*. *Busy* is TRUE as soon as the job is running. After processing the job, *Busy* with FALSE and *Done* with TRUE is returned. The recorded value can be found in *RecordedPosition*.

Status diagram of the block parameters



(1) The recording of the axis position is activated with edge 0-1 at *Execute* and *Busy* becomes TRUE.
(2) The axis position is recorded with edge 0-1 at *TriggerInput* and stored in *RecordedPosition*. The recording is completed. *Done* returns TRUE and *Busy* FALSE.
(3) Since the unique recording is completed, further trigger signals are ignored.

4.3.18 FB 824 - MC_AbortTrigger - abort recording axis position

Description

This block aborts the recording of the axis position, which was started via MC_TouchProbe.

Parameter

Parameter	Declaration	Data type	Description
Execute	IN	BOOL	The recording of the axis position is aborted with edge 0-1 at *Execute*.
Done	OUT	BOOL	■ Status – TRUE: Job successfully done. The recording of the axis position was aborted.
Busy	OUT	BOOL	■ Status – TRUE: Job is running.
Error	OUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis.
TriggerInput	IN_OUT	STRUCT	Reference to the trigger input. Structure ■ .Probe – 01: TouchProbe register 1 – 02: TouchProbe register 2 ■ .TriggerSource – 00: Input – 01: Encoder zero pulse ■ .TriggerMode – 00: SingleTrigger ■ .Reserved

Thus the job can be executed, the communication to the axis must be OK.

Abort the recording of the axis position

The recording of the axis position is aborted with edge 0-1 at *Execute*. *Busy* is TRUE as soon as the job is running. After processing the job, *Busy* with FALSE and *Done* with TRUE is returned.

4.3.19 FB 833 - Y_ReadParameter - read drive parameter

Description

With Y_ReadParameter the value of a parameter from the connected drive is read.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Read drive parameter data – Edge 0-1: The drive parameter data is reading.
Index	INPUT	WORD	Index of the drive parameter
Subindex	INPUT	BYTE	Subindex of the drive parameter
DIAG_BUF	INPUT	BOOL	■ Diagnostic buffer – TRUE: CPU diagnostic buffer entry (relevant for service personnel)
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was read
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 ‘ErrorID - Additional error information’ page 94*
ReadValue	OUTPUT	DWORD	Read Value
Axis	IN_OUT	STRUCT	Reference to the axis

PLCopen-State

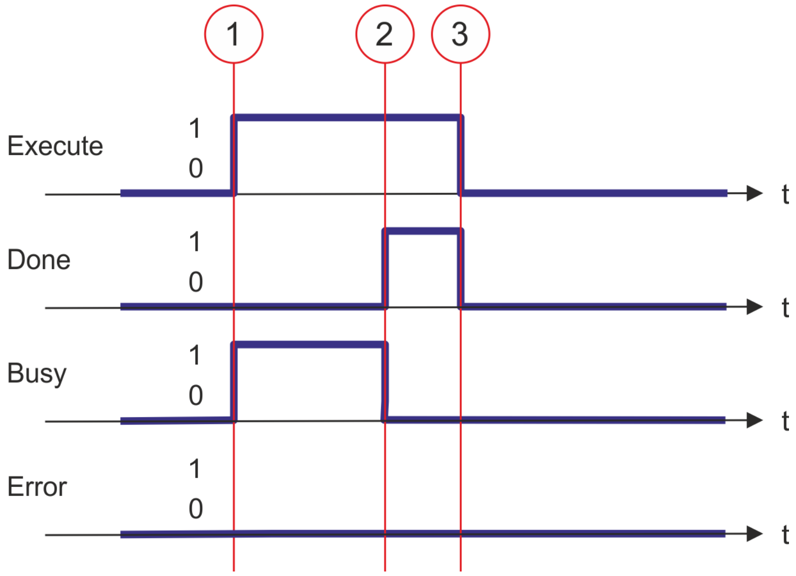
- Job start in each PLCopen-State possible.
- *Chap. 5.1 ‘PLCopen-States’ page 90*

Read drive parameter data

The reading of the parameter data is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as reading of parameter data is running. After the parameter data was read, *Busy* with FALSE and *Done* with TRUE is returned. The output *Value* shows the value of the parameter.

- *To access the drive parameters Pn000 ... Pn6FF, you have to add an offset of 0x9000 to the Index.*
  Example: Parameter Pn50A → Index 0x950A
- *Parameters with the following units are converted:*
  - *Unit "Pos.unit" → user unit*
  - *Unit "Vel.unit" → user unit/s*
  - *Unit "Vel.unit" → user unit/s²*
  - *Unit 0.1% Rated Motor Torque à 1.0% → 1.0% Rated Motor Torque*
  - *Unit (0.1% Rated Motor Torque)/s → (1.0% Rated Motor Torque)/s*
- *An active job continues to run even when Execute is set to FALSE.*

Status diagram of the block parameters



(1) At time (1) the reading of the parameter data is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.
(2) At the time (2) reading of the parameter data is successfully completed. *Busy* has the value FALSE and *Done* the value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.20 FB 834 - Y_WriteParameter - write drive parameter

Description

With Y_WriteParameter the value of the parameter is written to the connected drive.

Please do not change these parameters

When calling the Init block Y_SIG7PN_ServoInit, the following parameters are set. These should not be changed:

- *PnC00 ... PnC0F - Setpoint telegram: PZD 1 ... 16*
- *PnC10 ... PnC1F - Actual value telegram: PZD 1 ... 16*
- *PnC20 - Telegram selection*
- *PnB02 - Position user unit: Numerator*
- *PnB04 - Position user unit: Denominator*
- *PnB06 - Velocity user unit: Numerator*
- *PnB08 - Velocity user unit: Denominator*
- *PnB0A - Acceleration user unit: Numerator*
- *PnB42 - Position range limit (min.)*
- *PnB44 - Position range limit (max.)*
- *PnB48 - Software position limit (min.)*
- *PnB4A - Software position limit (max.)*
- *PnB0C - Acceleration user unit: Denominator*
- *PnB4C (607Fh): Max. profile velocity (Default: In reference to PnBF0 (2312h): Max. motor velocity)*
- *PnB7C (60C5h): Max. acceleration (Default: In reference to PnBF2 (2313h): Max. motor acceleration)*
- *PnB7E (60C6h): Max. deceleration (Default: In reference to PnBF2 (2313h): Max. motor acceleration)*
- *Pn205 - Multiturn limit*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Write drive parameter data – Edge 0-1: The drive parameter data is written.
Index	INPUT	WORD	Index of the drive parameter.
Subindex	INPUT	BYTE	Subindex of the drive parameter.
SetValue	INPUT	DINT	Reset the queue and initialize the buffer with *SetValue*.
DIAG_BUF	INPUT	BOOL	■ Diagnostic buffer – TRUE: CPU diagnostic buffer entry on (relevant for service personnel).
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was read.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.

Parameter	Declaration	Data type	Description
ErrorID	OUTPUT	WORD	Additional error information ↳ *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis.

PLCopen-State

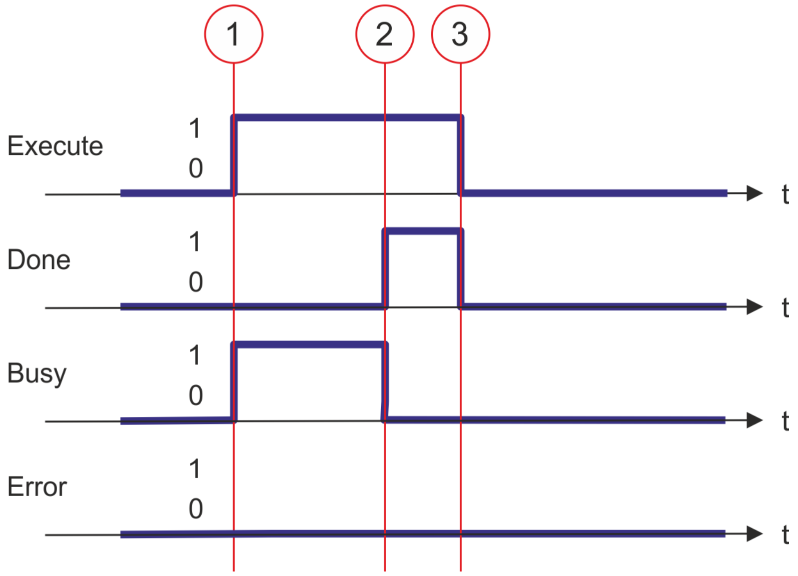
- Job start in each PLCopen-State possible.
- ↳ *Chap. 5.1 'PLCopen-States' page 90*

Write drive parameter data

The writing of the parameter data is started with an edge 0-1 at *Execute*. *Busy* is TRUE as soon as writing of parameter data is running. After the parameter data was written, *Busy* with FALSE and *Done* with TRUE is returned.

> – *To access the drive parameters Pn000 ... Pn6FF, you have to add an offset of 0x9000 to the Index.*
> *Example: Parameter Pn50A → Index 0x950A*
>
> – *An active job continues to run even when Execute is set to FALSE.*

Status diagram of the block parameters



(1) At time (1) the writing of the parameter data is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.
(2) At the time (2) writing of the parameter data is successfully completed. *Busy* has the value FALSE and *Done* the value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

4.3.21 FB 835 - Y_HomeInit_LimitSwitch - initialisation of homing on limit switch

Description This block initialises homing on limit switch.

Parameters

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Initialisation of the homing method – Edge 0-1: Values of the input parameter are accepted and the initialisation of the homing method is started.
Direction	INPUT	BOOL	■ Direction of homing – TRUE: on positive limit switch – FALSE: on negative limit switch
WithIndexPulse	INPUT	BOOL	■ Homing – TRUE: homing with index pulse – FALSE: homing without index pulse
VelocitySearchSwitch	INPUT	REAL	Velocity for search for the switch in [user units/s]
VelocitySearchZero	INPUT	REAL	Velocity for search for zero in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisation successfully done.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisation is active.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

Initialisation homing on limit switch

The values of the input parameters are accepted with an edge 0-1 at *Execute* and the initialisation of the homing method is started. As long as the initialisation is active, the output *Busy* is set to TRUE. If the initialisation has been completed successfully, the output *Done* is set to TRUE. If an error occurs during initialisation, the output *Error* is set to TRUE and an error number is output at the output *ErrorID*.

Initialisation of the homing method

1. Verify communication to the axis.
2. Check for permitted PLCopen-States.
3. Check the input values
   - Input *VelocitySearchSwitch* [user units]
     - 0.0 < *VelocitySearchSwitch* ≤ max. velocity
     - The max. velocity can be determined with FB 833 - Y_ReadParameter (*Index* = 2312h).
     - *Chap. 4.3.19 ‘FB 833 - Y_ReadParameter - read drive parameter’ page 76*
   - Input *VelocitySearchZero* [user units]
     - No index is used: 0.0
     - Index is used: 0.0 < VelocitySearchSwitch ≤ max. velocity
   - Input Acceleration [user units]
     - 0 < *Acceleration* ≤ max. acceleration
     - The max. acceleration can be determined with FB 833 - Y_ReadParameter (*Index* = 2313h).
     - *Chap. 4.3.19 ‘FB 833 - Y_ReadParameter - read drive parameter’ page 76*
4. Transfer of the drive parameters:
   - "Homing Method" in dependence of input "Direction"
     See table below!
   - "Homing Speed during search for switch" [Inc/s]
   - "Homing Speed during search for zero" [Inc/s]
   - "Homing Acceleration" [Inc/s^2]

Homing Method	Direction	WithIndexPulse
1	FALSE	TRUE
2	TRUE	TRUE
17	FALSE	FALSE
18	TRUE	FALSE

4.3.22 FB 836 - Y_HomeInit_HomeSwitch - initialisation of homing on home switch

Description This block initialises homing on home switch.

Parameters

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Initialisation of the homing method – Edge 0-1: Values of the input parameter are accepted and the initialisation of the homing method is started.
WithIndexPulse	INPUT	BOOL	■ Homing – TRUE: homing with index pulse – FALSE: homing without index pulse
SameDirIndexPulse	INPUT	BOOL	■ Direction for index pulse – TRUE: After detecting the home, search for index pulse without change of direction – FALSE: After detecting the home, search for index pulse with change of direction
OnRisingEdge	INPUT	BOOL	■ Edge of home switch – TRUE: Edge 0-1 – FALSE: Edge 1-0
VelocitySearchSwitch	INPUT	REAL	Velocity for search for the switch in [user units/s]
VelocitySearchZero	INPUT	REAL	Velocity for search for zero in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisation successfully done.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisation is active.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *↻ Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

Initialisation homing on home switch

The values of the input parameters are accepted with an edge 0-1 at *Execute* and the initialisation of the homing method is started. As long as the initialisation is active, the output *Busy* is set to TRUE. If the initialisation has been completed successfully, the output *Done* is set to TRUE. If an error occurs during initialisation, the output *Error* is set to TRUE and an error number is output at the output *ErrorID*.

Initialisation of the homing method

1. Verify communication to the axis.
2. Check for permitted PLCopen-States.
3. Check the input values
   - Input *VelocitySearchSwitch* [user units]
     – 0.0 < *VelocitySearchSwitch* ≤ max. velocity
     – The max. velocity can be determined with FB 833 - Y_ReadParameter (*Index* = 2312h).
     – *Chap. 4.3.19 ‘FB 833 - Y_ReadParameter - read drive parameter’ page 76*
   - Input *VelocitySearchZero* [user units]
     – No index is used: 0.0
     – Index is used: 0.0 < VelocitySearchSwitch ≤ max. velocity
   - Input Acceleration [user units]
     – 0 < *Acceleration* ≤ max. acceleration
     – The max. acceleration can be determined with FB 833 - Y_ReadParameter (*Index* = 2313h).
     – *Chap. 4.3.19 ‘FB 833 - Y_ReadParameter - read drive parameter’ page 76*
4. Transfer of the drive parameters:
   - "Homing Method" in dependence of input "Direction"
     See Table below!
   - "Homing Speed during search for switch" [Inc/s]
   - "Homing Speed during search for zero" [Inc/s]
   - "Homing Acceleration" [Inc/s^2]

Parameter

Homing Method	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
3	TRUE	TRUE	FALSE
4	TRUE	TRUE	TRUE
5	TRUE	FALSE	TRUE
6	TRUE	FALSE	FALSE
19	FALSE	TRUE	FALSE
20	FALSE	TRUE	TRUE
21	FALSE	FALSE	TRUE
22	FALSE	FALSE	FALSE

4.3.23 FB 837 - Y_HomeInit_ZeroPulse - initialisation of homing on zero pulse

Description

This block initialises homing on zero pulse.

Parameters

Parameter	Declaration	Data type	Description
Execute	IN	BOOL	■ Initialisation of the homing method – Edge 0-1: Values of the input parameter are accepted and the initialisation of the homing method is started.
Direction	IN	BOOL	■ Direction of homing – TRUE: Positive direction – FALSE: Negative direction
VelocitySearchZero	IN	REAL	Velocity for search for zero in [user units/s]
Acceleration	IN	REAL	*Acceleration* in [user units/s^2]
Done	OUT	BOOL	■ Status – TRUE: Initialisation successfully done.
Busy	OUT	BOOL	■ Status – TRUE: Initialisation is active.
Error	OUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

Initialisation homing on zero pulse

The values of the input parameters are accepted with an Edge 0-1 at *Execute* and the initialisation of the homing method is started. As long as the initialisation is active, the output *Busy* is set to TRUE. If the initialisation has been completed successfully, the output *Done* is set to TRUE. If an error occurs during initialisation, the output *Error* is set to TRUE and an error number is output at the output *ErrorID*.

Initialisation of the homing method

1. Verify communication to the axis.
2. Check for permitted PLCopen-States.
3. Check the input values
   - Input *VelocitySearchSwitch* [user units]
     - 0.0 < *VelocitySearchSwitch* ≤ max. velocity
     - The max. velocity can be determined with FB 833 - Y_ReadParameter (*Index* = 2312h).
     - *Chap. 4.3.19 'FB 833 - Y_ReadParameter - read drive parameter' page 76*
   - Input *VelocitySearchZero* [user units]
     - No index is used: 0.0
     - Index is used: 0.0 < VelocitySearchSwitch ≤ max. velocity
   - Input Acceleration [user units]
     - 0 < *Acceleration* ≤ max. acceleration
     - The max. acceleration can be determined with FB 833 - Y_ReadParameter (*Index* = 2313h).
     - *Chap. 4.3.19 'FB 833 - Y_ReadParameter - read drive parameter' page 76*

4. Transfer of the drive parameters:

- "Homing Method" in dependence of input "Direction". See table below!
- "Homing Speed during search for switch" [Inc/s]
- "Homing Speed during search for zero" [Inc/s]
- "Homing Acceleration" [Inc/s^2]

Homing Method	Direction
33	FALSE
34	TRUE

4.3.24 FB 838 - Y_HomeInit_SetPosition - initialisation of homing mode set position

Description

This block initialises homing on current position.

Parameters

Parameter	Declaration	Data type	Description
Execute	IN	BOOL	■ Initialisation of the homing method – Edge 0-1: Values of the input parameter are accepted and the initialisation of the homing method is started.
Done	OUT	BOOL	■ Status – TRUE: Initialisation successfully done.
Busy	OUT	BOOL	■ Status – TRUE: Initialisation is active.
Error	OUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis

Initialisation homing on home switch

The values of the input parameters are accepted with an edge 0-1 at *Execute* and the initialisation of the homing method is started. As long as the initialisation is active, the output *Busy* is set to TRUE. If the initialisation has been completed successfully, the output *Done* is set to TRUE. If an error occurs during initialisation, the output *Error* is set to TRUE and an error number is output at the output *ErrorID*.

Initialisation of the homing method

1. Verify communication to the axis.
2. Check for permitted PLCopen-States.
3. Transfer of the drive parameters:
   - "Homing Method" = 35

4.3.25 FB 839 - MC_TorqueControl - Move axis with constant torque

Description

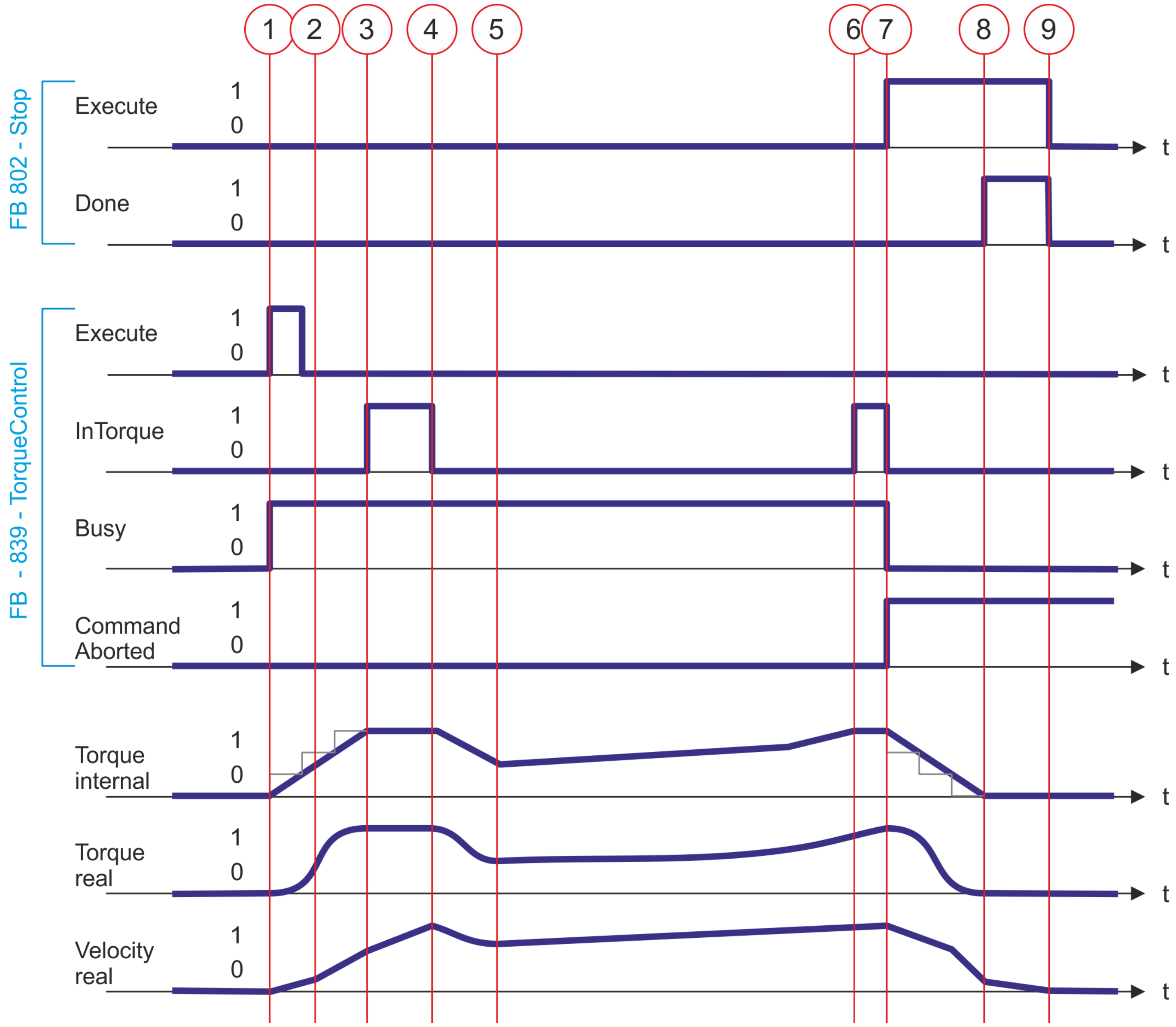
With MC_TorqueControl you can specify a maximum torque.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Start motion at edge 0-1.
Torque	INPUT	REAL	Value of the torque/force in [1.0% Rated Motor Torque] on motor side, regardless of the configured mechanics.
TorqueRamp	INPUT	REAL	Torque ramp per second [(1.0% Rated Motor Torque)/s].
Velocity	INPUT	REAL	Value of the maximum velocity on motor side, regardless of the configured mechanics. Please use the following units: ■ Rotary motor: [rpm] ■ Linear motor: [mm/s]
InTorque	OUTPUT	BOOL	■ Torque – TRUE: Commanded torque reached.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running - torque has not been reached.
Active	OUTPUT	BOOL	■ Status – TRUE: Block controls the axis.
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 6 'ErrorID - Additional error information' page 94*
Axis	IN_OUT	STRUCT	Reference to the axis.

Please note that this block uses acyclic resources of the CPU when checking and transmitting the parameters. Due to this, the movement does not start until the parameters have been transferred.

Status diagram and stop behavior



(1) With edge 0-1 at *Enable* of FB 839 the movement of the axis is started and *Busy* returns TRUE.
(2) With gradually increasing torque, the speed increases until one of the maximum values *Torque* or *Velocity* is reached.
(3) The axis reaches the specified torque and *InTorque* returns TRUE. The axis continues to accelerate until *Velocity* is reached.
(4) The axis reaches the preset *Velocity* and the torque is reduced. Here *InTorque* returns FALSE.
(5) Due to the drive resistance, the velocity decreases. This is counteracted by gradually increasing the torque.
(6) The axis reaches the specified torque and *InTorque* returns TRUE.
(7) With edge 0-1 at *Execute* of FB 802 the stopping of the axis is started. The axis is decelerated to standstill. *CommandAborted* of FB 839 returns TRUE, *InTorque* and *Busy* FALSE.
(8) The stopping of the axis is completed, the axis is stopped and *Done* of FB 802 returns TRUE.
(9) With edge 1-0 at *Execute* of FB 802 the stopping job is finished and *Done* of FB 802 returns FALSE.

4.3.26 FB 840 - MC_ReadActualTorque - read actual torque

Description

As long as *Enable* is TRUE, this block returns the value of the torque. As soon as output data are available, *Valid* becomes TRUE. If *Enable* is reset, *Valid* will also be reset, even if new output data are available.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Get the value of the parameter continuously while enabled.
Valid	OUTPUT	BOOL	■ A valid set of output values is available in the FB.
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information. *Chap. 6 'ErrorID - Additional error information' page 94*
Torque	OUTPUT	REAL	Actual torque/force of the axis in [1.0% Rated Motor Torque] on motor side, regardless of the configured mechanics.
Axis	IN_OUT	STRUCT	Reference to the axis.

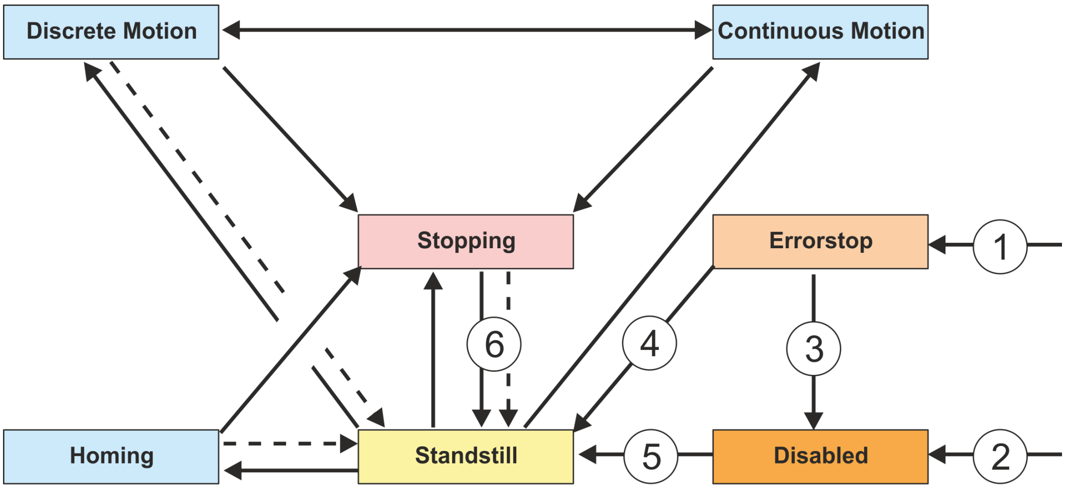
5 States and behavior of the outputs

5.1 PLCopen-States

State diagram

The *state diagram* shows all the states that an axis can assume. An axis is always in one of these states. Depending on the output state, a state change can take place automatically or via the blocks of the axis control. In principle, movement tasks are processed sequentially. You can use the following function blocks to query the state

- ☞ *Chap. 4.3.11 ‘FB 812 - MC_ReadStatus - read status axis’ page 63*



⇢ Return when done
(1) From each state: An error has occurred at the axis
(2) From each state: MC_Power.Enable = FALSE and there is no error on the axis
(3) MC_Reset and MC_Power.Status = FALSE
(4) MC_Reset and MC_Power.Status = TRUE and MC_Power.Enable = TRUE
(5) MC_Power.Enable = TRUE and MC_Power.Status = TRUE
(6) MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

There are the following states

- Disabled
  - Basic state of an axis.
  - Axis can not be moved by any function block.
- Error Stop
  - An error has occurred on the axis.
  - Axis is stopped and is blocked for further motion tasks.
  - Axis remains in this state until the error is solved and a RESET is triggered.
  - Errors on an axis are also reported via the corresponding function block.
  - Errors on a function block do not lead to this state
- Standstill
  - Ready for motion tasks
  - There is no error on the axis
  - There are no motion tasks active on the axis
  - Axis is power supplied
- Stopping
  - Axis is currently stopped:
    ☞ *Chap. 4.3.5 ‘FB 802 - MC_Stop - stop axis’ page 50*
  - The *Stopping* state is active as long as a Stop command is active (*Execute* = 1). Even if the axis is already stopped. Then the state automatically changes to *Standstill*.

- Homing
  - The axis is currently homing:
    ➪ *Chap. 4.3.4 'FB 801 - MC_Home - home axis' page 48*
  - As soon as the axis is homed, the state automatically changes to *Standstill*.
- Discrete Motion
  - The axis is currently executing a motion task:
    ➪ *Chap. 4.3.9 'FB 808 - MC_MoveAbsolute - move axis to absolute position' page 59*
    ➪ *Chap. 4.3.7 'FB 804 - MC_MoveRelative - move axis relative' page 55*
    ➪ *Chap. 4.3.6 'FB 803 - MC_Halt - holding axis' page 53*
  - As soon as the target of the movement task is reached, the state automatically changes to *Standstill*.
- Continuous Motion
  - The axis performs a permanent movement task:
    ➪ *Chap. 4.3.8 'FB 805 - MC_MoveVelocity - drive axis with constant velocity' page 57*

5.2 Behavior of the inputs and outputs

Exclusivity of the outputs

- The outputs *Busy*, *Done*, *Error* and *CommandAborted* exclude each other, so at a function block only one of these outputs can be TRUE at a time.
- As soon as the input *Execute* is TRUE, one of the outputs must be TRUE. Only one of the outputs *Active*, *Error*, *Done* and *CommandAborted* can be TRUE at one time.

Output status

- The outputs *Done*, *InVelocity*, *Error*, *ErrorID* and *CommandAborted* are reset with an edge 1-0 at the *Execute* input if the function block is not active (*Busy* = FALSE).
- The command execution is not affected by an edge 1-0 of *Execute*.
- If *Execute* is already reset during command execution, so it is guaranteed that one of the outputs is set at the end of the command for a PLC cycle. Only then the outputs are reset.

Input parameter

- The input parameters are taken with edge 0-1 at *Execute*.
- To change the parameters the command must be retriggered.
- If an input parameter is not passed to the function block, the last transferred value to this block remains valid.
- With the first call a sensible default value must be passed.

Position an distance

- The input *Position* designates an absolute position value.
- *Distance* designates a relative measure as distance between two positions.
- Both *Position* and *Distance* are preset in technical units e.g. [mm] or [°], in accordance to the scaling of the axis.

Parameter for the dynamic behavior

- The dynamic parameter for *Move* functions are preset in engineering units with second as the time base.

  If an axis is scaled in millimetres so the units are for *Velocity* [mm/s], *Acceleration* [mm/s^2], and *Deceleration* [mm/s^2].

Error handling

- All the function blocks have two fault outputs to indicate errors during command execution.
- *Error* indicates the error and *ErrorID* shows an additional error number.
- The outputs *Done* and *InVelocity* designate a successful command execution and are not set if *Error* becomes TRUE.

Error types

- Function block errors
  - Function block errors are errors that only concerns the function block and not the axis such as e.g. incorrect parameters.
  - Function block errors need not be explicitly reset , but will automatically reset when the input *Execute* is reset.
- Communication errors
  - Communication error such as e.g. the function block can not address the axis.
  - Communication errors often indicate an incorrect configuration or parametrization.
  - A reset is not possible, but the function block can be retriggered after the configuration has been corrected.
- Axis errors
  - Axis errors usually occur during the move such as e.g. position error.
  - An axis error must be reset by MC_Reset.

Behavior of the *Done* output

- The *Done* output is set, when a command was successfully executed.
- When operating with multiple function blocks at one axis and the current command is interrupted by another block, the *Done* output of the first block is not set.

Behavior of the *CommandAborted* output

- *CommandAborted* is set when a command is interrupted by another block.

Behavior of the *Busy* output

- The *Busy* output indicates that the function block is active.
- *Busy* is immediately set with edge 0-1 of *Execute* and will not be reset until the command was completed successfully or failed.
- As long as *Busy* is TRUE, the function block must be called cyclically to execute the command.

Behavior of the *Active* output

- If the motion of an axis is controlled by several function blocks, the *Active* output of each block indicates that the command is executed by the axis.

***Enable*-Input and *Valid* output**

- In contrast to *Execute* the *Enable* input causes that an action is permanently and continuously executed, as long as *Enable* is TRUE. MC_ReadStatus e.g. cyclically refreshes for example the status of an axis as long as *Enable* is TRUE.
- A function block with a *Enable* input indicates by the *Valid* output that the data of the outputs are valid. However, the data can constantly be updated during *Valid* is TRUE.

6 ErrorID - Additional error information

ErrorID	Description	Remark
0x0000	No Error	
0x8001	Invalid value at parameter *Position* (out of range).	
0x8002	Invalid value at parameter *Distance* (out of range).	
0x8003	Invalid value at parameter *Velocity*.	
0x8004	Invalid value at parameter *Acceleration*.	
0x8005	Invalid value at parameter *Deceleration*.	
0x8011	Invalid value at parameter *Source*.	
0x8012	Invalid value at parameter *Direction*.	
0x801D	Parameter communication with general error. The cause of the error is not described in detail.	
0x802F	No system resources available.	
0x8050	Rack: Wrong Device-ID / Manufacturer-ID.	MC_Power, Y_Init Y_ServoFunction
0x8051	Not supported firmware version.	Y_Init
0x8052	Configured slot is not a PN device.	MC_Power
0x8053	Configured rack not at place on slot.	MC_Power
0x8054	Device address (diagnosis address) is "0".	MC_Power
0x8057	Wrong device state / module not ready for operation. Rack not available and / or module not available.	MC_Power Y_Init Y_ServoFunction
0x8058	Device address of the module is not detectable.	
0x8059	Device connection lost.	Y_Init Y_ServoFunction
0x805A	Device not ready for operation, device not available. Expected type is not equal to actual type.	MC_Power
0x805B	Module: Wrong Device-ID / Manufacturer-ID.	MC_Power
0x805C	Wrong telegram configuration in the drive. Expected telegram number "999" is not present.	MC_Power Y_Init
0x805D	Wrong telegram configuration in the PLC. Configured telegram number is not "999".	MC_Power Y_Init
0x805E	Module is not ready for operation.	MC_Power
0x805F	Driver not ready.	Y_Init, Y_ServoFunction Y_ReadParameter, Y_WriteParameter
0x8060	Read SERVOPACK information failed.	Y_Init

ErrorID	Description	Remark
0x8061	Invalid configuration: Combination of inputs *Resolution*, *Gearbox-Factor* and *Traversing* is not supported. Internal *pos.unit* factor is out of range.	Y_Init
0x8062	Invalid configuration: Internal value for *Resolution*, *GearboxFactor* and *Traversing* is "0".	Y_Init
0x8063	Invalid configuration: Motor resolution too high.	Y_Init
0x8064	Unknown / not supported encoder/motor type.	Y_Init
0x8065	User unit setting failed.	Y_Init
0x8066	Invalid position limits configured.	Y_Init
0x8067	Write position limits failed.	Y_Init
0x8068	Read motor information failed.	Y_Init
0x8069	Set communication settings failed.	Y_Init
0x806A	Axis setting failed: Set endless axis information.	Y_Init
0x806B	Save PROFINET parameter failed.	Y_Init
0x806C	Software reset to activate communication and application settings failed.	Y_Init
0x806D	Axis setting failed: Alarm Multiturn-Limit disagreement is not available.	Y_Init
0x806E	Axis setting failed: Multiturn-Limit setting procedure.	Y_Init
0x806F	Axis setting failed: Baseblock state without alarm expected.	Y_Init
0x8070	Software reset to activate multiturn-limit settings failed.	Y_Init
0x8090	Specified logical address invalid.	MC_Power
0x8094	No subnet was configured with the specified SUBNET-ID.	MC_Power
0x8095	Illegal value for STATION parameter of hardware configuration.	MC_Power
0x8096	Illegal value for SLOT parameter of hardware configuration.	MC_Power
0x8097	Illegal value for SUBSLOT parameter of hardware configuration.	MC_Power
0x8099	The SLOT is not configured.	MC_Power
0x809A	The interface module address is not configured for the selected SLOT.	MC_Power
0x8101	No communication with axis possible.	
0x8102	Command is in current PLCopen-State not allowed.	
0x8103	Command is not supported by the axis.	
0x8104	Axis is not ready to switch on, possible reasons: ■ Communication to the axis is not ready. ■ Drive is not in status *'S2: Ready For Switching On'* → reset drive error possibly with MC_Reset. ■ Communication was interrupted, e.g. by CPU power cycle. Reset error with MC_Reset.	
0x8201	Command cannot be executed temporarily because of lack of internal resources (no free slot in command buffer / parameter buffer).	

ErrorID	Description	Remark
0x8203	Homing has no valid data for homing configuration because homing was not configured.	MC_Home
0x8204	Set homing configuration failed (Home *Method*, Home *Offset*, Home *Speed Switch*, Home *Speed Zero*, Home *Acceleration*).	MC_Home
0x820A	Write *TorqueRamp* failed.	MC_TorqueControl
0x820B	Write *Velocity* failed.	MC_TorqueControl
0x820C	Parameter Buffer: Internal error	
0x820D	Parameter Buffer: Timeout	
0x820E	Internal error: Programming error	
0x820F	Timeout: *DeviceCom* timer for read request / write request expired (900ms).	
0x8211	Timeout: *DeviceCom* timer that limits the max. processing time expired (25s).	
0x8212	Timeout: *DeviceDriver* timer for acyclic request expired (5s).	
0x8306	Communication errors at the master axis. Slave axis is stopped with fast stop.	
0x8316	FB called in more than one cyclical OB is not allowed.	
0x8317	FB called in not supported OB.	
0x8340	Invalid value at *TriggerInput*.Probe.	MC_TouchProbe, MC_AbortTrigger
0x8341	Invalid value at *TriggerInput*.Source.	MC_TouchProbe, MC_AbortTrigger
0x8342	Invalid value at *TriggerInput*.TriggerMode.	MC_TouchProbe, MC_AbortTrigger
0x8350	Invalid value at *VelocitySearchSwitch*.	Y_HomeInit...
0x8351	Invalid value at *VelocitySearchZero*.	Y_HomeInit...
0x8352	Invalid combination of inputs.	Y_HomeInit...
0x8363	Stop command is active. A New command is not allowed while a stop command is active (*Execute* = TRUE).	
0x8400	Unexpected Drive-State, Drive-State ≠ *Operation enabled*	MC_Power
0x8401	Unexpected Drive-State, Drive-State = *Quick stop active*	MC_Power
0x8402	Unexpected Drive-State, Drive-State = *Fault reaction active*	MC_Power
0x8403	Unexpected Drive-State, Drive-State = *Fault*	MC_Power
0x8410	Timeout while trying to reset the drive.	MC_Reset
0x851B	Invalid value at parameter *Torque*.	MC_TorqueControl
0x851C	Invalid value at parameter *TorqueRamp*.	MC_TorqueControl
0x851D	Invalid value at parameter *CmdType* respectively *Mode*.	Y_ServoFunction
0x8A00	Impermissible parameter number: Access to unavailable parameter.	Y_ReadParameter, Y_WriteParameter
0x8A01	Parameter value cannot be changed: Change access to a parameter value that cannot be changed.	Y_WriteParameter

ErrorID	Description	Remark
0x8A02	Low or high limit exceeded: Change access with value outside the value limits.	Y_WriteParameter
0x8A03	Invalid subindex: Access to unavailable subindex.	Y_ReadParameter, Y_WriteParameter
0x8A04	No array: Access with subindex to non-indexed parameter.	Y_ReadParameter, Y_WriteParameter
0x8A05	Incorrect data type: Change access with value that does not match the data type of the parameter.	Y_ReadParameter, Y_WriteParameter
0x8A06	Setting not permitted (may only be reset): Change access with value ≠ "0" where this is not permitted.	Y_WriteParameter
0x8A0B	No operation priority: Change access without rights to change parameters.	Y_ReadParameter, Y_WriteParameter
0x8A11	Request cannot be executed because of operating mode: Refer to Sigma-7 SERVOPACK user manual for details.	Y_ReadParameter, Y_WriteParameter
0x8A14	Value impermissible: Refer to Sigma-7 SERVOPACK user manual for details.	Y_WriteParameter
0x8A17	Illegal format: Write request: Illegal format or format of the parameter data which is not supported.	Y_ReadParameter, Y_WriteParameter

Appendix

Content

A History of changes

Rev.	Changes
19-16	The manual was created.
19-40	Include library - Chapter "Integration into Siemens TIA Portal" was added. Usage Sigma-7 PROFINET - Chapter "Usage in Siemens TIA Portal" was added.